

Extension of cross validation with confidence to determining number of communities in Stochastic Block Models

Jining Qin

Aug 2019

Department of Statistics & Data Science
Dietrich College of Humanities and Social Sciences
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Jing Lei
Alessandro Rinaldo
Larry Wasserman
Kehui Chen (University of Pittsburgh)

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

Abstract

Stochastic block model (SBM) and its variants constitute an important family of methods for modeling network data. There is a rich literature on methods for estimating the block labels and model parameters of stochastic block models, as well as study on the properties of such methods. Most of these studies would require the number of communities K as an input, making this an important problem. There are several methods proposed for this problem such as spectral methods, likelihood based methods, information criteria, Bayesian methods, and cross-validation. Cross-validation is a natural option for this problem since it is a widely used generic method for evaluating statistical methods, easy to be adapted for various scenarios. However, cross-validation is known to be inconsistent and prone to over-fitting unless using impractical split ratios. Cross-validation with confidence (CVC) has recently been proposed as a variation of cross-validation with better theoretical guarantees in conventional settings.

In this thesis we studied the properties of cross-validation with confidence for stochastic block models. Practically, we implemented different variations of this method by changing the train-test split scheme, approaches of obtaining the sampling distribution for the test statistic, and the loss function in cross-validation. We checked the performance of our method amongst these variations and against similar established methods in the literature. We checked its robustness under misspecification and different data generating processes. We also tested our algorithm by applying it to two widely used real-world data sets. In addition, through theoretical studies, we show that under certain assumptions, our method is guaranteed to eliminate under-fitting candidate models. We also further showed that CVC, unlike standard cross-validation, can consistently pick the optimal K by showing that the validated loss of the true model is not much worse than that of a slightly over-fitting model. Therefore, the candidate set output by a CVC procedure will contain the optimal model with guaranteed probability.

Acknowledgements

First and foremost I would like to thank my advisor Jing Lei. Through the years that we worked together, he has truly been a caring and supportive mentor. We started our collaboration on a quite different project, which didn't eventually pan out. Despite my struggles and procrastination over the years, despite the fact that I didn't quite have an appetite for an academic career, he has always been understanding and helpful. He would give me spot on advice on both minute details in my research problem and the guiding principles of conducting myself as a PhD student. Looking back, I couldn't wish for a better advisor. I very frequently feel guilty of wasting his precious time by asking him naive, foolish, and sometimes repetitive questions. His kindness and patience would touch me every time. Although from his perspective, I could only hope he wouldn't regret the decision of taking me as his student too much, just like how I feel about my decision to pursue a PhD.

I would like to thank my committee members Kehui Chen, Larry Wasserman, Alessandro Rinaldo, as well as other faculty members who provided me with generous help and valuable suggestions, including but not limited to Cosma Shalizi, David Krackhardt. I would like to also thank Professor Joel Greenhouse and Professor Ann Lee. 6 years ago, it was Joel's decision which brought me to the wonderful environment of CMU statistics and got me riveted to it. And 5 years ago, it was with Ann's help that I would continue my life here pursuing a PhD in statistics. I am constantly marvelled at the pleasant surprises life brings me. As a non-typical applicant to our statistics PhD program, I didn't get to where I am without excessive good fortune. The opportunity to spend 6 years in such a wonderful place is the piece of chocolate that I will be forever grateful for.

I would like to also thank fellow students from our department. I was really grateful for the strong support throughout my stay here. In the first couple of years, I can always find people to go to when I have questions about the course work. And in the later years, I could always find just the right collaborators when I get interested in various competitions or side projects. I definitely free-rided my way through a fair amount of them, and I'm constantly grateful to my teammates for tolerating me. Participating in IM-sports together with fellow students from our department, especially getting in some deep playoff runs, makes some of my best memories here.

I would like to thank my future colleagues, including and especially Bo and Xianchao, for putting their trust in me and giving me a chance to work with them. I realize what a luxury it is to work on my thesis without needing to worry about job searching in the final year in school. It was a great experience working amongst you guys during my internship and I hope we will be having more good time working together in the future.

My life in the past five years wouldn't have been nearly as pleasant if not for several people whom I am fortunate enough to become friends with. I am thankful for all the friends who spent time with me through happy and frustrating times. Some of these friends are right here in Pittsburgh and some of them barely met me in person during these years. Some of them are not here for the whole ride, yet I will always be grateful for the company I enjoyed. My sincere thanks to Jiaqi, Xiaofeng, Lei, Tippi, Xin, Chong. It is my distinguished privilege to witness the starting of two wonderful families: Fei, Mo, Nola; Cici, Gary and Adaline. I wish all the best for you and your babies.

Those who are familiar with me know that I am not the most proactive person. Luckily I have a lot of sources of inspiration in my life. At times when I need a little stimulus or motivation, it would always be easy to see someone out there chasing his/her dream, laying everything on the line. Some of my heroes are people I meet in person from time to time, while some of them maintain interactions with me mostly thanks to the Internet. Some of my heroes are people whose stories I only get to know about from reading their books or listening about their stories. I want to thank all those people nonetheless. Your pursuits, your struggles, and your enthusiasm remind me that there is always another level and I can always choose to live my life to the fullest.

I would like to also thank some people who likely wouldn't expect to be listed here. There was a time when I was sliding into the vicious spiral of unhealthy life habits and questionable work ethics. Their appearance in my life served as the critical inspiration to pull myself up and start thinking more about the present and the future. Their positive attitude to life and concentration on what really matters make me want to lead my life the same way and obtain the same kind of joy out of life. To some extent, I believe many people's life is determined by the number and timing of their run-ins with such people who shed light on how they should make decisions about their life. I will always be thankful for the shining lights I am fortunate enough to encounter.

Last but not least, I want to thank my parents for being supportive most of the time throughout my excessively long career as a student. Ever since I was a little child, they provided all they could for me to get well educated. I wouldn't be who I am today without their influences. I want to thank both of them. I hope I have made you proud at some points in the past, and I hope I can continue to do so in the future.

Contents

1	Introduction	1
1.1	Prevalence of network data and overview of network study	1
1.2	Stochastic Block Model and Its Variants	2
1.3	Summary of Our Contribution	4
1.4	Notation table	6
2	Background and Algorithm Description	7
2.1	Background and Limitation of Classical Cross-validation	8
2.2	Train-test splitting in Network Settings	10
2.3	Estimation of Stochastic Block Models	14
2.4	Cross-validation with Confidence	18
2.5	V-fold Cross-validation with Confidence in the Context of Stochastic Block Models	21
2.6	Using Parametric Bootstrap instead of Gaussian Multiplier Bootstrap	25
3	Simulation Studies and Application on Data Sets	29
3.1	Studies through simulated data	29
3.1.1	Simulation of adjacency matrices from Stochastic Block Models	29
3.1.2	Illustration of the Hypothesis Testing Step	31
3.1.3	Effect of the hypothesis testing step: comparison with BIC and NCV	32
3.1.4	Sensitivity to mis-specification	39
3.1.5	One case of unidentifiability	42
3.1.6	Sensitivity to different community interaction matrix setting	48
3.1.7	Changing up the train-test splitting: comparison with other CV methods	58
3.1.8	Swapping out Gaussian Multiplier Bootstrap: comparison with para- metric bootstrap	61
3.1.9	Note on Computation Speed	64
3.2	Application on real-world data sets	67
3.2.1	Political Blog Data Set	67
3.2.2	Political Books Data Set	72
4	Theoretical Results	75
4.1	Under-fitting case	78
4.2	Over-fitting case	82

5	Conclusions and Future Work	85
A	Proofs for results in 4.1	87
A.1	When A_s is relatively large, and A_s^c grows sufficiently fast.	88
A.2	When A_s is relatively large, and A_s^c grows very slowly.	93
A.3	When A_s is not large enough.	94
B	Proofs for results in 4.2	97
B.1	When all the estimates are moderate from the over-fitting model	98
B.2	When some estimates are extreme from the over-fitting model	101
	Bibliography	105

List of Figures

1.1	Three examples of network data set. Left plot: a gene co-expression network, [RCPHLR ⁺ 16]. Middle plot: Marriage connection network among notable families in 15th Century Florence, originally [PA93], re-introduced in [Jac10]. Right plot: Payment network between Bitcoin wallet addresses controlled by major Bitcoin exchanges, [GS18].	1
1.2	Illustration of a network data set generated from a stochastic block model with $K = 4$. The interaction between each two communities clearly have different frequencies.	3
2.1	Illustration of random edge sampling. Here we show an adjacency matrix of size 15×15 split into 3 folds. Left plot shows the fold indices without being made symmetric. Right plot shows the fold indices after being made symmetric.	11
2.2	Illustration of block-wise node-pair splitting.	12
2.3	Comparison between edge sampling ([LLZ16]), block-wise node-pair splitting ([CL17]) and LatinCV ([DJ16]). Plot from [Dab16].	12
2.4	Illustration of latinCV edge sampling. Here we show an adjacency matrix of size 15×15 split into 3 folds. Left plot shows the fold indices without being made symmetric. Right plot shows the fold indices after being made symmetric.	13
2.5	Left: block-wise node-pair splitting [CL17]. Middle: LatinCV [DJ16]. Right: random edge sampling [LLZ16].	13
3.1	P-values when testing whether the candidate K is rejected by CVC. Data sets are generated to be networks with 1200 nodes with true $K = 4, 6$) and different density ($r \in \{0.025, 0.05, 0.1, 0.15, 0.2\}$).	31
3.2	Results for simulation. The plots show the proportions of 200 simulated datasets for which K is correctly estimated. The datasets are generated from SBMs with $K = 2, 3, 4$, sparsity levels $r \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$, and various levels of community imbalance. n_1 represents the size of the first community with other community sizes being equal. Total number of nodes of 1000.	38

3.3	The same results, but using Network Cross-Validation proposed in [CL17]. The datasets are generated from SBMs with $K = 2, 3, 4$, sparsity levels $r \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$, and various levels of community imbalance. n_1 represents the size of the first community with other community sizes being equal. Total number of nodes of 1000. The top plot uses the likelihood function as the objective while the bottom plot uses squared error loss as the objective function.	38
3.4	Left plot shows the variances explained by each principal component of the political blogs adjacency matrix. Right plot shows the projections of all political blogs onto first two principal components of their adjacency matrix.	67
3.5	P-values for different candidate K values using 3-fold CVC assuming standard stochastic block model.	68
3.6	P-values for different candidate K values using 3-fold CVC assuming degree-corrected stochastic block model.	69
3.7	P-values for different candidate K values using 5-fold and 10-fold CVC assuming degree-corrected stochastic block model. Here we are using the extra squared error loss function.	70
3.8	P-values for different candidate K values using 5-fold and 10-fold CVC assuming degree-corrected stochastic block model. Here we are using the negative likelihood loss function.	70
3.9	Left plot shows the variances explained by each principal component of the political blogs adjacency matrix. Right plot shows the projections of all political blogs onto first two principal components of their adjacency matrix.	72

List of Tables

1.1	Notations used in later part of this thesis	6
2.1	Example of a test set using LatinCV. Here we are splitting all node pair into 3 fold in a network of 6 nodes.	27
3.1	Percentage of under-fitting, correct and over-fitting model selection results for NCV and CVC methods, under different network density parameter and true community numbers. Data sets in this table are generated using the standard Stochastic Block Model. NCV tends to select under-fitting models when we are dealing with more true communities, while CVC performs much better. In low-density network data sets, CVC tends to select under-fitting models.	33
3.2	Percentage of under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) model selection results for NCV and CVC methods, under different network density parameter and true community numbers. Data sets in this table are generated using the standard Stochastic Block Model. NCV tends to select under-fitting models when we are dealing with more true communities, while CVC performs much better. In low-density network data sets, CVC tends to select under-fitting models.	34
3.3	Percentage of under-fitting, correct and over-fitting model selection results for NCV and CVC methods, under different network density parameter and true community numbers. Data sets in this table are generated using the standard Stochastic Block Model. We can see CVC outperforms NCV in many scenarios, especially when we have a high network density and high number of communities (bottom right corner of the table). In these cases NCV tends to select over-fitting models, while CVC selects the correct model the majority of times. Although we observe some cases where CVC is underperforming compared to NCV, for example, in the top right corner of the table, when we have relatively high number of communities with low network densities. Cross-validation with confidence tends to select under-fitting models in this case.	36

3.4	Percentage of under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) model selection results for NCV and CVC methods, under different network density parameter and true community numbers. Data sets are generated using the standard Stochastic Block Model. We can see CVC outperforms NCV in many scenarios, especially when we have a high network density and high number of communities (bottom right corner of the table). In these cases NCV tends to select over-fitting models, while CVC selects the correct model the majority of times. Although we observe in the top right corner of the table, when we have relatively high number of communities with low network densities, NCV outperforms CVC. Cross-validation with confidence tends to select under-fitting models in this case.	37
3.5	Under-fitting, correct, and over-fitting percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.2, 1).	40
3.6	Under-fitting, correct, and over-fitting percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.5, 1).	40
3.7	Under-fitting, correct, and over-fitting percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.75, 1).	40
3.8	Under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.2, 1).	41
3.9	Under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.5, 1).	41
3.10	Under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.75, 1).	41
3.11	Proportion of each candidate K being selected using CVC method with squared error loss using $d = 2$, under various network density levels. The column for $\hat{K} = 2$ is highlighted since it is the correct number of communities assuming standard stochastic block model (according to Equation 3.2).	43
3.12	Proportion of each candidate K being selected using CVC method with squared error loss and weighting each column of the singular vector matrix U by the square-root of the corresponding singular value d_i , under various network density levels. The column for $\hat{K} = 2$ is highlighted since it is the correct number of communities assuming standard stochastic block model (according to Equation 3.2).	43
3.13	Proportion of each candidate K being selected using CVC method with different loss functions, under various network density levels. The column for $\hat{K} = 1$ is highlighted since it is the correct number of communities assuming degree-corrected stochastic block model (according to Equation 3.2).	44
3.14	Proportion of each candidate K being selected using NCV method with different loss functions, under various network density levels. The column for $\hat{K} = 1$ is highlighted since it is the correct number of communities assuming degree-corrected stochastic block model (according to Equation 3.2).	44

3.15	Frequency for each model to be selected using Algorithm 9. The data sets here are generated using prototype community interaction matrix as defined in Equation 3.2.	45
3.16	Proportion of each candidate K being selected using CVC method with squared error loss, under various network density levels. The column for $\hat{K} = 3$ is highlighted since it is the correct number of communities assuming standard stochastic block model (according to Equation 3.3).	45
3.17	Proportion of each candidate K being selected using CVC method with squared error loss, under various network density levels. The column for $\hat{K} = 3$ is highlighted since it is the correct number of communities assuming standard stochastic block model (according to Equation 3.3).	46
3.18	Proportion of each candidate K being selected using CVC method with different loss functions, under various network density levels. The column for $\hat{K} = 2$ is highlighted since it is the correct number of communities assuming degree-corrected stochastic block model (according to Equation 3.3).	46
3.19	Frequency for each model to be selected using Algorithm 9. The data sets here are generated using prototype community interaction matrix as defined in Equation 3.3.	46
3.20	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 2$. Each community is equally sized. The entire network has 300 nodes.	48
3.21	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 1$. Each community is equally sized. The entire network has 300 nodes.	49
3.22	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{2}$. Each community is equally sized. The entire network has 300 nodes.	49
3.23	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{4}$. Each community is equally sized. The entire network has 300 nodes.	49
3.24	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 2$. Each community is equally sized. The entire network has 600 nodes.	50
3.25	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 1$. Each community is equally sized. The entire network has 600 nodes.	50

3.26	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{2}$. Each community is equally sized. The entire network has 600 nodes.	50
3.27	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{4}$. Each community is equally sized. The entire network has 600 nodes.	51
3.28	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 2$. Each community is equally sized. The entire network has 300 nodes.	51
3.29	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 1$. Each community is equally sized. The entire network has 300 nodes.	52
3.30	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{2}$. Each community is equally sized. The entire network has 300 nodes.	52
3.31	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{4}$. Each community is equally sized. The entire network has 300 nodes.	52
3.32	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 2$. Each community is equally sized. The entire network has 600 nodes.	53
3.33	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 1$. Each community is equally sized. The entire network has 600 nodes.	53
3.34	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{2}$. Each community is equally sized. The entire network has 600 nodes.	53
3.35	Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{4}$. Each community is equally sized. The entire network has 600 nodes.	54
3.36	Under-fitting, correct, and over-fitting model selection percentages for running Algorithm 8 on network data sets generated on deterministic and non-diagonal-dominant community interaction matrices.	54

3.37	Under-fitting (red), correct(green), and over-fitting(blue) model selection percentages for running Algorithm 8 on network data sets generated on deterministic and non-diagonal-dominant community interaction matrices.	54
3.38	Under-fitting, correct, and over-fitting model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.4$. .	55
3.39	Under-fitting, correct, and over-fitting model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.6$. .	56
3.40	Under-fitting, correct, and over-fitting model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.8$. .	56
3.41	Under-fitting (red), correct (green), and over-fitting (blue) model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.4$	56
3.42	Under-fitting (red), correct (green), and over-fitting (blue) model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.6$	56
3.43	Under-fitting (red), correct (green), and over-fitting (blue) model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.8$	57
3.44	Percentage of under-fitting, correct and over-fitting model selection results for CVC methods with random edge sampling ('random CV' in the table) and LatinCV, under different network density parameter and true community numbers.	59
3.45	Under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) percentage of model selection results for CVC methods with random edge sampling ('random CV' in the table) and LatinCV, under different network density parameter and true community numbers.	60
3.46	Percentage of under-fitting, correct and over-fitting model selection results for CVC methods with parametric bootstrap, under different network density parameter and true community numbers. Data sets in this table are generated using the standard Stochastic Block Model.	62
3.47	Percentage of under-fitting(light red on the left of green), correct (green), over-fitting (blue on right of green), and none (gray) model selection results for CVC methods with parametric bootstrap. Data sets generated using the standard Stochastic Block Model.	63
3.48	Example of a test set using LatinCV. Here we are splitting all node pair into 3 fold in a network of 6 nodes.	64

3.49	Average running time on a network with 600 nodes with 3 equally sized communities and bootstrap sample size $B = 200$. For all the tasks we are using 32-core computer with 64G memory, each core with 2.6GHz speed. Note that for the first five tasks, parallelism doesn't really help with running time, so a 4-core computer each with 3.6GHz speed (numbers in parentheses) would actually be faster for the same tasks. Each average running time estimate comes from average of 100 runs. Note that for the parametric bootstrap tasks, we heavily optimized some slower R functions by re-writing them in C++ with the help of Rcpp package and parallelism. So this is not really a fair comparison. In fact, block-wise node pair splitting with Gaussian multiplier bootstrap is much faster if we implement all of algorithms verbatim.	66
3.50	Confusion matrix between manual labels (columns) and estimated community labels assuming a degree-corrected stochastic model with $K = 2$	68
3.51	Frequency table for selected K values using 3-fold CVC on the political book data set, with extra squared error loss function.	69
3.52	Confusion matrix between book labels (columns) and estimated communities (rows) assuming DC-SBM with $K = 2$	72
3.53	Most frequent model selection results selecting in both standard stochastic block models and degree-corrected stochastic block models, using spectral weighting of the singular vector matrix. We select the most parsimonious model within the retained K 's in each category. We omitted the results that appear fewer than 5 times out of 200 runs.	73

List of Algorithms

1	V-fold classical cross validation for prediction problem	8
2	V-fold train-test splitting via edge sampling	10
3	V-fold train-test splitting via block-wise node-pair splitting	11
4	V-fold train-test splitting via LatinCV	12
5	Model estimation for stochastic block models using rectangular spectral clustering, using block-wise node-pair splitting	14
6	Model estimation for stochastic block models using rectangular spectral clustering, using LatinCV or random edge sampling, with matrix completion	15
7	V-fold cross validation with confidence for prediction problem	19
8	V-fold block-wise node-pair splitting cross validation with confidence	21
9	V-fold block-wise node-pair splitting cross validation with confidence, selecting among both SBMs and DC-SBMs	23
10	V-fold block-wise node-pair splitting cross validation with confidence, via parametric bootstrap	25
11	Random prototype community interaction matrix generation	55

Chapter 1

Introduction

1.1 Prevalence of network data and overview of network study

We live in an increasingly interconnected world. Many important phenomena occur in the form of connections between various networks: people form social networks through their interaction, businesses form a network of payment and supply through flow of money and goods, genes form a network through their co-expression activities.

These networks usually carry a great amount of information about the group they occur in. Careful study of the network data would lead to useful insight into the topics: examination of the payment network can expose trade relations as well as money laundering activities, study of social network can lead to identification of important and influential social groups, exploration of gene co-expression network can help us understand the mechanism of diseases such as Autism.

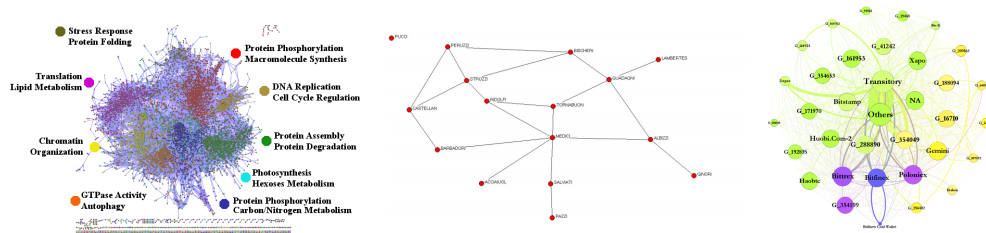


Figure 1.1: Three examples of network data set. Left plot: a gene co-expression network, [RCPHLR⁺16]. Middle plot: Marriage connection network among notable families in 15th Century Florence, originally [PA93], re-introduced in [Jac10]. Right plot: Payment network between Bitcoin wallet addresses controlled by major Bitcoin exchanges, [GS18].

After examining a lot of specific cases of networks, people start to realize that the abstract connection patterns in these networks also have some interesting feature and are worth in-depth exploration on their own. This leads to the study of networks. This field started from descriptive summary of network and the nodes within them, such as degree distribution, degree

centrality. With the development of probability and statistics, models for depicting the random generation of network data sets emerge, which assume the networks we observe are one-time realizations of such data generating processes. With the spectacular growth of the Internet, we are able to collect data and observe the evolution of networks such as Facebook and Tor with precision never imagined before. The study of dynamic networks thus emerged to model how networks evolve over time. General overviews of this field can be found in [Lin10], [New10]. [GZF⁺10] is a more concise survey. While [EK⁺10] and [Jac10] are among the interesting special treatments of this topic.

As statisticians, we are interested in the network data not only because they are useful in practice, but also because their modeling leads to some neat models as well as interesting estimation and inference problems. Statistical models for modeling network data usually considers the edges in the network to be random and try to depict the generating process of them using features of the nodes, etc. Depending on whether we take one snapshot of the network or at least several snapshots at different time points, we will be using either static or dynamic network modeling techniques. Most notable and well-studied models belong to the static network modeling subset. For example, Erdos-Reyni model assumes that the entire network has a uniform edge probability and edges form independently. Latent Space Model as introduced in [HRH02] would assign a position for each node in a feature space, and the edge probability between two nodes would be inversely proportional to their distance in the space. Our focus will be on Stochastic Block Models, which stress the structural equivalence within the network.

1.2 Stochastic Block Model and Its Variants

Stochastic block models refer to a family of statistical models for network data. It takes into account the structural equivalence within a network: each node is assigned a community label, then the edge probability between any two nodes would depend only on the interaction pattern between the communities they belong to. In a network of n different nodes. A symmetric binary adjacency matrix $\left(\mathbf{A}_{ij}\right)_{n \times n}$ is used to record whether an edge exists between two nodes: if an edge exists between node i and node j , then $A_{ij} = 1$, otherwise $A_{ij} = 0$. Each node i has a community label $g_i \in \{1, 2, \dots, K\}$, assigning it to one of K communities. And the edge between any two nodes depends on the community interaction matrix $\mathbf{B}_{K \times K}$:

$$A_{ij} \sim \text{Bernoulli}(B_{g_i g_j})$$

Figure 1.2 is a simple illustration of a network data set generated from a stochastic block models with four true communities. After sorting the nodes properly, we can actually obtain a lot of information about the community interaction matrix B by looking at the interaction frequency between different communities in the figure.

The earliest suggestions of stochastic block models can be found in [FW81] [HLL83]. It refers to the model of networks where each node is assigned a discrete block label and the edge probability between two nodes is determined solely by the blocks they belong to. More refined version of the SBM includes the degree-corrected SBM (DCSBM) proposed in [KN11], where the nodes within the same block are allowed more flexibility in terms of degree heterogeneity.

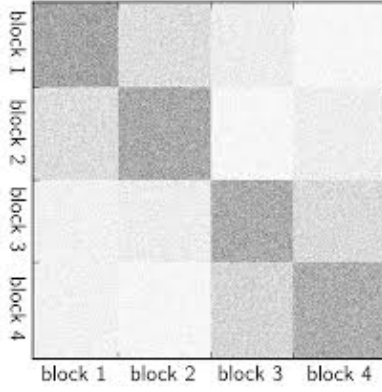


Figure 1.2: Illustration of a network data set generated from a stochastic block model with $K = 4$. The interaction between each two communities clearly have different frequencies.

Specifically, each node is assigned a degree correction parameter $\theta_i, i = 1, \dots, n$. And the edge between any two nodes would instead be determined by:

$$A_{ij} \sim \text{Bernoulli}(\theta_i \theta_j B_{g_i g_j})$$

Here each node is assigned its own popularity score, and it can have its own degree distribution despite its community membership. The regular Stochastic Block Model can be seen as the special case where all θ_i 's are constant.

Also, in mixed membership stochastic block models([ABFX08]), each node is allowed to be assigned to several different blocks. SBM and its variants are widely used in modeling of networks in social science and biology, see [BC09], [DPR08].

There is a rich literature for the estimation of block memberships and interaction probabilities in SBMs. The methods come in several different flavors. [New06], [ZLZ11] approaches it by optimizing a discrete objective function based on the edges in the network. [Vuo89], [BC09], and [ACBL13] use likelihood-related methods. [YSJ⁺14] further showed how to select between degree-corrected SBMs and the regular SBMs using likelihood-related methods. [DPR08], [LBA12] solves the estimation problem using variational methods. [RCY11], [FST⁺13], [KMM⁺13], and [Jin15] treats it as a spectral clustering problem. [DKMZ11] proposes a belief propagation algorithm, borrowing perspectives from statistical physics. There are also many studies exploring the theoretical properties of these methods. [BC09], [RCY11], [CDP12], [BCCZ13], [LR15], [ABH16] are all examples of such work, studying the asymptotic properties of various methods mentioned above. However, most of the proposed methods, as well as studies into their properties, would require the true number of communities in the network K as an input.

Determining the number of communities in SBM has attracted considerable interest in both statistics and machine learning community. Many methodological studies would propose a corresponding approach to determine number of communities. Several approaches are recursive, for example, [GN02], [NG03b], [NG03a] proposed the idea of recursively splitting up blocks till some stopping criterion and largely rekindled the community's interest in community recovery using stochastic block models. [ZLZ11] and [BS16] also proposed methods of the same flavor.

[LL15] gives fast method for estimating K using the spectral properties of some graph operators. [Lei16] develops a goodness-of-fit test for stochastic block models based on the largest singular value of the residual matrix. The test is shown to be powerful against models with finer structures and thus leads to a sequential testing for estimating number of communities for SBMs. [WB17] motivates their method by deriving the log-likelihood ratio statistic’s limiting distribution in case of under-fitting and convergence rate in case of over-fitting. [HW08], [MMFH13] and [CL15] proposed to tackle the problem from Bayesian viewpoint.

[Hof08] proposes cross-validation for model selection in this context. Yet the train-test split is based on node pairs, which leads to cumbersome computation and unnecessary complication since the test set node memberships would need to be determined again. [CL17], on the other hand, examines properties of the network cross-validation method based on block-wise node-pair splitting. The model is estimated using the rectangular fitting set containing the interaction between training set nodes and all nodes. The community memberships are determined simultaneously in the estimation stage and information in the data set gets fully utilized. [LLZ16] proposes edge sampling, which is to select node pairs at random, then use a low-rank matrix completion to get the training set on this subset of node-pairs. And then evaluate the loss function on the held-out node pairs.

Cross validation is widely used for model evaluation and selection in statistics and machine learning. However, traditional cross validation doesn’t take into account the randomness in the test set and thus usually selects over-fitting models over the most appropriate ones. [Sha93] and [Zha93] pointed out that traditional cross validation cannot achieve consistency unless the testing set dominates the training set in size, which neither leave-one-out CV nor k-fold cross validation satisfies. In [Lei17], the author developed a new, statistically principled approach to select optimal model by combining cross validation and Gaussian multiplier bootstrap approach. In the original paper, the method is developed and validated on relatively straightforward settings such as linear regression. In principle, CVC applies to a wide range of model selection problems where there is a well-defined loss function yet the estimation of the loss function is complicated by randomness in the validation set.

1.3 Summary of Our Contribution

In this work we will try to extend the CVC method to the problem of selecting number of communities in Stochastic Block Models and Degree Corrected Stochastic Block Models. We will define the cross validation with confidence in this setting and discuss the methodology choices we make in the process. In network settings, the data splitting step is no-longer straightforward and we are facing several options, each with its pros and cons. We are also faced with the choice of two loss functions, as well as different options for conducting the bootstrap step in order to obtain the sampling distribution for the test statistic. We will then explore its effectiveness by testing the method on simulated data as well as real data sets. We will compare our method against existing methods in the literature on simulated and real-world data sets. We will also examine the performance of different versions of our method through simulation. We will then examine the property of CVC for SBM from a theoretical standpoint. We will prove its validity by showing that it is guaranteed with high probability to output a set of candidate models that

contains the true model under certain mild assumptions. And with a simple model selection principle, we are guaranteed to arrive at the true model with guaranteed probability.

This thesis is organized as follows: chapter 2 will cover the background, setup of the problem, and then define the CVC method in Stochastic Block Model setting, as well as the variants and existing methods in the literature we will be looking into. we will discuss in depth the choices we are faced with that would lead to different versions of the algorithm. In chapter 3, we will first demonstrate the validity of CVC method using simulated data. Then we will compare the performance of its different versions among themselves as well as against other methods under representative settings. We will also explore the limit of its effectiveness under less desirable settings. We then apply these methods on real-world data sets. In chapter 4 we discuss the theoretical properties of CVC for SBM. In particular, we show that the under-fitting models (i.e. SBMs with fewer number of communities than optimal) are guaranteed to be rejected by the CVC procedure, and the over-fitting models won't be able to eliminate the true model in the hypothesis testing step. The two conclusions combined would imply the consistency of CVC method for SBMs. And in chapter 5, we would have some discussion and about the implication of our work and list some potential future directions.

1.4 Notation table

Here we list the notations we will be using later in this thesis for reference.

Notation	Meaning
A_{ij}	Edge between node i and node j , $A_{ij} = 1$ if there is an edge, 0 otherwise.
$n, n^{(tr)}, n^{(te)}$	Number of nodes in the entire network, training set, testing set, respectively.
g_i	The true community label of node i .
$\hat{g}_i^{(K, tr)}$	Estimated community label of node i using model with K communities. using the training set data only.
K^*	True number of communities in a network.
\mathbb{K}	Set of candidate K values.
B_{kl}^*	Real edge probability between community k and community l .
$\hat{B}_{kl}^{(K, tr)}$	Estimated edge probability between estimated community k and l using the training set and model with K communities.
$I_{k, k'}$	$\{(i, j) : g_i = k, g_j = k'\}$. The node pairs where first node comes from true community k and second node comes from true community k' .
$\hat{I}_{k, k'}$	$\{(i, j) : \hat{g}_i = k, \hat{g}_j = k'\}$. The node pairs where first node has estimated community label k and second node has estimated community label k' .
ρ_n	Network density parameter. True edge probabilities are proportional to ρ_n .
\mathbb{N}_v	Nodes in the v th fold.
$\tau_{k, k', l, l'}$	$\hat{I}_{k, k'} \cap I_{l, l'}$, the node pairs (i, j) where $i \in \hat{I}_k \cap I_l, j \in \hat{I}_{k'} \cap I_{l'}$
$\hat{P}_{k, k', l, l'}$	the average of A_{ij} over $\tau_{k, k', l, l'}$
$\Omega(\cdot)$	Asymptotic lower bound: $g(n) = \Omega(f(n))$ means $\exists M > 0, k > 0, s.t. \forall n > M, g(n) > k \cdot f(n)$
$O_P(\cdot)$	Stochastic boundedness notation: if $x_n = O_P(a_n)$ $\forall \epsilon > 0, \exists M > 0, N > 0, s.t. P(\frac{x_n}{a_n} > M) < \epsilon, \forall n > N$

Table 1.1: Notations used in later part of this thesis

Chapter 2

Background and Algorithm Description

In this thesis, the method we want to propose and explore is the cross-validation with confidence on network data for the purpose of selecting number of communities in stochastic block models. Given a network data set in the form of symmetric binary adjacency matrix $(A)_{n \times n}$ (or equivalently, an adjacency list) and a list of candidate K values \mathbb{K} , we want to output the number of true communities K in this network in the sense that the model estimates given using a K -community model would be most desirable.

This algorithm has a multi-step process. Some of these steps can actually be altered slightly and it would end up as a variant of our method. To perform cross-validation, we first need to split the data set into training and testing set, which is not a trivial decision in network setting. Then we can estimate the community labels of all nodes on the training set, the method for which would differ based on the models we are using (for example, SBM vs DC-SBM) and specific preferences of statisticians. We would then evaluate a loss function for the estimated model using the test set, which involves a choice between different loss functions (we will explore two of them). Then among different candidate K s in \mathbb{K} , we will compare their performance according to their respective test set losses. Straightforward V -fold cross-validation would lead to a simple comparison over the one train-test split without considering the randomness in train-test splitting and the test set data points. While cross-validation with confidence would take it into account and use a formal hypothesis testing step and only choose one K value over another if it is significantly better. In order to obtain the sampling distribution of the test statistic in the hypothesis testing step, we can either use Gaussian multiplier bootstrap, as proposed in [Lei17], or use a parametric bootstrap scheme by generating multiple network data sets under the null hypothesis. In the following section, we will discuss in depth the decisions we make each step and the variations we will get when we use slightly different methods in some of these steps.

We are aware of the many routes we can take in constructing the CVC method under network settings. We will try to examine as many of these variants as possible in Chapter 3, where we explore their performance using simulation and data application, and mostly focus on one case in Chapter 4, where we try to establish theoretical guarantees for the method.

2.1 Background and Limitation of Classical Cross-validation

The most general statistical problem usually comes in the following form, given a set of data points generated from an unknown distribution

$$Z_1, Z_2, \dots, Z_n \sim \mathbb{P}$$

we would want to find a model \hat{P} based on the information in the data point Z_i 's such that it resembles true data generating process \mathbb{P} . We usually characterize how well the model fits the data set using some loss function:

$$L(\hat{P}, \mathbb{P})$$

The loss function is often a positive valued function. It is smaller when the model fits the data set well. However, since \mathbb{P} is unknown, we can only come up with an estimate using the data points $\mathbf{Z} = (Z_1, \dots, Z_n)$ generated from \mathbb{P} .

$$\hat{L}(\hat{P}, \mathbf{Z})$$

When we have a set of several candidate models $\{\hat{P}_{m_1}, \dots, \hat{P}_{m_k}\}$, then we are faced with the model selection problem, where we need to select the model most appropriate for the data set. The straightforward and somewhat naive way of solving this problem is simplistically choosing the model that would give the smallest estimated loss function value. i.e. pick the model

$$\arg \min_{m_i} \hat{L}(\hat{P}_{m_i}, \mathbf{Z})$$

The problem with this approach is over-fitting. When we have a set of flexible models, straightforward loss function minimization would lead to the model that fits not only the signal but also the noise in the data set. The model would have great performance in sample, yet generalizes very poorly outside of the given data set.

Cross-validation is an ingenious approach to the model selection problem while addressing the over-fitting problem. The classical cross-validation widely used in various fields in statistics, largely due to it being very generic and adaptable to many circumstances. It is based on the notion that the model shouldn't have 'seen' the data points it will be later evaluated on during the training phase, so that it wouldn't be over-fitting to the data set.

Below we list describe the classical cross-validation algorithm for a prediction problem. In other words, we have many data points (x_i, y_i) and we want to find a model \hat{f} such that for a new value x , we can predict y with good enough accuracy. In other words, we want to minimize the loss function $L(\hat{f}(x), y)$ for new (x, y) pairs.

Algorithm 1 V-fold classical cross validation for prediction problem

Data: Data points $\mathbb{D} = \{(x_i, y_i), i = 1, 2, \dots, n\}$. A loss function we want to minimize for predictions $L(\hat{f}(x), y)$. A set of candidate models to select from $\{m_1, m_2, \dots, m_k\}$.

Result: The model that will likely yield the lowest out-of-sample prediction loss.

1. Randomly split the data set into V equal-sized subsets $\{\mathbb{D}_v : 1 \leq v \leq V\}$.
2. For each $1 \leq v \leq V$, and each $m \in \{m_1, m_2, \dots, m_k\}$:

- (a) Estimate model $\hat{f}_m^{(D^{-v})}(\cdot)$ for model m using all the data points except in the v th fold: $\mathbb{D}^{(-v)} = \{(x_i, y_i), i \notin \mathbb{D}_v\}$.
- (b) Evaluate the loss function of $\hat{f}_m^{(D^{-v})}$ using the data points in $\mathbb{D}^{(v)}$:

$$\sum_{(x_i, y_i) \in \mathbb{D}^{(v)}} L(y_i, \hat{f}_m^{(D^{-v})}(x_i))$$

- 3. For each $m \in \{m_1, m_2, \dots, m_k\}$, calculate the aggregate cross-validation loss:

$$\mathbb{L}(m_i) = \sum_{v \in \{1, \dots, V\}} \sum_{(x_i, y_i) \in \mathbb{D}^{(v)}} L(y_i, \hat{f}_m^{(D^{-v})}(x_i))$$

- 4. Return the model with the lowest aggregate cross-validation loss.

Here we are using simple prediction problem just for illustration. In fact, classical cross-validation can be adapted for a lot of scenarios as long as we have an explicit loss function, and the model can be trained with a subset of the available data set.

There are, however, limitations of this method. Results in [Sha93] and [Zha93] show that classical cross-validation cannot achieve model selection consistency in its typical use case. In essence, we are evaluating the performance of models based on their aggregate error measure on one instance of train-test splitting, in which case the worse model could look superior just due to luck. Especially when we are dealing with a series of nested models, classical cross-validation would still lead us to an over-fitting model.

These limitations of classical cross-validation leads to the proposal of cross-validation with confidence in [Lei17], which we will discuss in more detail in 2.4.

2.2 Train-test splitting in Network Settings

Under simpler settings such as the prediction problems shown above, the train-test splitting is almost a trivial process: given a train-test ratio, we randomly select a certain portion of all data points to be in the training set and the rest becomes the test set, when we need to do V -fold cross-validation, then we randomly split all data points into V equally sized portions and each portion would rotate as the test set each time.

However, under network settings, the train-test splitting is also no longer straightforward. The data set is (V, E) , the combination vertices and edges in the network. It is also acceptable to think of the data set as (V, A) , the combination of all vertices and the adjacency matrix, which records the existence of edges between any two node pairs.

A somewhat naive way of doing train-test splitting in this case would be to directly split all node-pairs into a train set and a test set base on a given train-test ratio randomly, as proposed in [Hof08]. For the purpose of model selection, we would still estimate different models on the train set and then compare their prediction performance on the test set. The problem is that it is possible that node-pairs would not be equally represented within different folds, and it is possible that we cannot give prediction for certain nodes pairs at all if one or both of the nodes don't show up in the training set. Also, this method would lead to cumbersome computation problems. [LLZ16] also proposed using this method, but added a matrix completion step after selecting the random subset of edges. In this way, the (completed) adjacency matrix provided to the estimation algorithm will not be porous, making it easier for the later steps. This approach can be described as Algorithm 2. Note here we have an optional step 4 for symmetricizing the fold index matrix. Note that in our model estimation step, we will first conduct eigen-decomposition of the training set. Making sure the input matrix is symmetric will guarantee that the eigen-vectors all have real entries.

Algorithm 2 V -fold train-test splitting via edge sampling

Data: Number of nodes n . Fold number V .

Result: A $n \times n$ matrix \mathbf{I} indicating the fold number of each entry in the $n \times n$ adjacency matrix. $\mathbf{I}_{ij} \in \{1, \dots, V\}$.

1. Randomly shuffle the vector $(1, 2, \dots, n^2)$. Let \vec{a} denote the resulting vector.
2. Calculate $\vec{a'}$ using \vec{a} using the following equation:

$$\vec{a'}_i = \left(\vec{a}_i \mod V \right) + 1, \quad i = 1, 2, \dots, n^2$$

3. Align $\vec{a'}$ into a $n \times n$ matrix:

$$\mathbf{I} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ a_{n+1} & a_{n+2} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n(n-1)+1} & a_{n(n-1)+2} & \cdots & a_{n^2} \end{bmatrix}$$

4. **(Optional)** Set the lower-triangle of \mathbf{I} to the transpose of the upper-triangle of \mathbf{I} . This step would make the resulting index matrix symmetric, but might make the fold index distribution deviate a little from a even split between different folds.

5. Return \mathbf{I} as the final result.

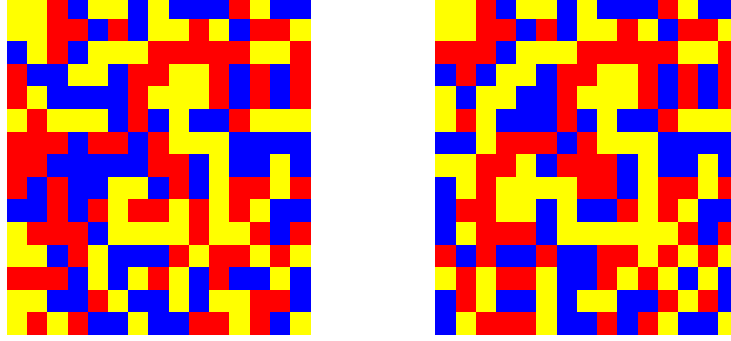


Figure 2.1: Illustration of random edge sampling. Here we show an adjacency matrix of size 15×15 split into 3 folds. Left plot shows the fold indices without being made symmetric. Right plot shows the fold indices after being made symmetric.

We can also view nodes in the network as subjects in train-test splitting. For example, in [CL17], the authors proposed block-wise node-pair splitting. In this case all nodes are split into V folds. Only the interactions between test set nodes (i.e. if both nodes in the node pair belongs to the v th fold) are considered the test set, while all other node pairs (we will henceforth refer to as the rectangular set, as the red region shown in Figure 2.2) are considered the test set. We summarize the block-wise node-pair splitting in Algorithm 3.

Algorithm 3 V -fold train-test splitting via block-wise node-pair splitting

Data: Number of nodes n . Fold number V .

Result: A $n \times n$ matrix \mathbf{I} indicating the fold number of each entry in the $n \times n$ adjacency matrix. $\mathbf{I}_{ij} \in \{1, \dots, V\}$. Entries in each row will always have the same value.

1. Randomly shuffle the vector $(1, 2, \dots, n)$. Let \vec{a} denote the resulting vector.
2. Align \vec{a} into a $n \times n$ matrix by assigning all columns of \mathbf{I} to be \vec{a} :

$$\mathbf{I} = \begin{pmatrix} \vec{a}, \vec{a}, \dots, \vec{a} \end{pmatrix}$$

3. Return \mathbf{I} as the final result.
-

[Dab16] proposed Latin-CV, a well-calibrated method to get around this problem. Still viewing node-pairs as the subjects to be split, the authors made sure to guarantee that in the node-pair splitting, all nodes-pairs will be equally represented in each fold, and all node pairs will have a chance to be in the test set. The comparison is illustrated in Figure 2.3. In the latinCV splitting, the training set is no longer rectangular and the test set is no longer square. Yet all node-pairs have equal representation in the training and testing set, unlike using block-wise node-pair splitting. We describe LatinCV in Algorithm 4.

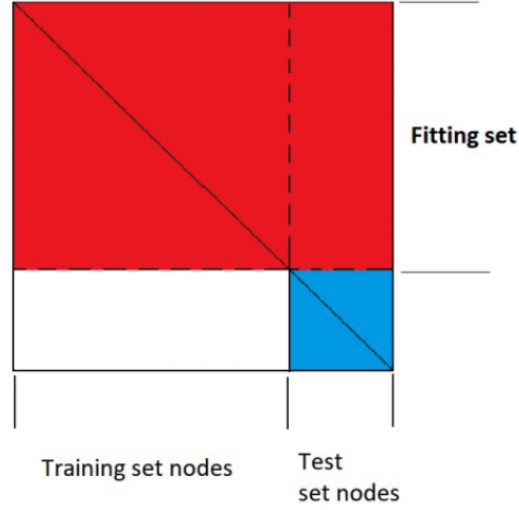


Figure 2.2: Illustration of block-wise node-pair splitting.

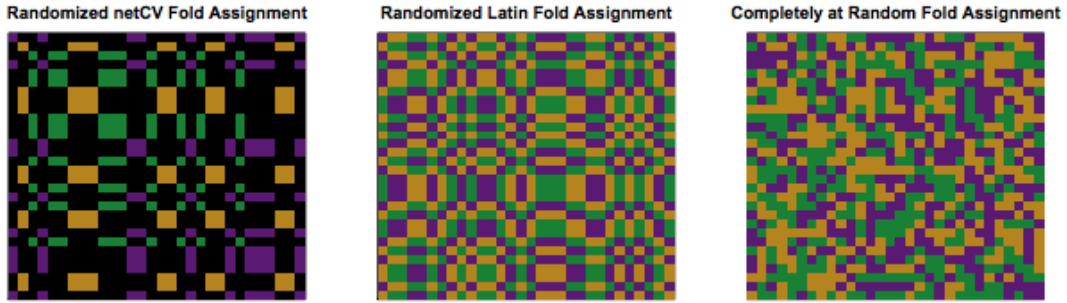


Figure 2.3: Comparison between edge sampling ([LLZ16]), block-wise node-pair splitting ([CL17]) and LatinCV ([DJ16]). Plot from [Dab16].

Algorithm 4 V-fold train-test splitting via LatinCV

Data: Number of nodes n . Fold number V .

Result: A $n \times n$ matrix \mathbf{I} indicating the fold number of each entry in the $n \times n$ adjacency matrix. $\mathbf{I}_{ij} \in \{1, \dots, V\}$. Fold index $\{1, \dots, V\}$ will be (at least approximately) evenly distributed in each row and column.

1. Randomly shuffle the vector $(1, 2, \dots, n)$ twice. Let \vec{a}, \vec{b} denote the resulting vectors.
2. Assign the entries in \mathbf{I} using the following equation:

$$\mathbf{I}_{ij} = \left((\vec{a}_i + \vec{b}_j) \mod V \right) + 1$$

3. **(Optional)** Set the lower-triangle of \mathbf{I} to the transpose of the upper-triangle of \mathbf{I} . This step would make the resulting index matrix symmetric, but might make the fold index distribution deviate a little from a even split between different folds.

4. Return \mathbf{I} as the final result.

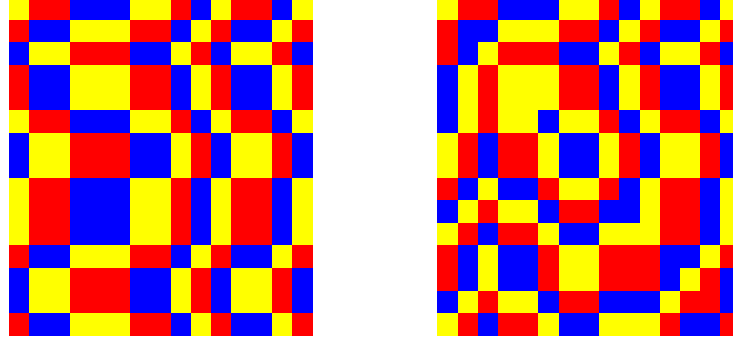


Figure 2.4: Illustration of latinCV edge sampling. Here we show an adjacency matrix of size 15×15 split into 3 folds. Left plot shows the fold indices without being made symmetric. Right plot shows the fold indices after being made symmetric.

Other proposed methods include [LLZ16]. The authors proposed to sample node-pairs at random, and then conduct a low-rank matrix completion over the selected sample and use the completed matrix as the training set. After training the model, it is evaluated on the held-out node pairs. This can be seen as an improvement upon the more straightforward approach in [Hof08].

In Figure 2.5, we demonstrate the train-test splitting for one fold under the three splitting schemes we discuss here. Again, we use the adjacency matrix for a network with 15 nodes and split the data set into three folds.

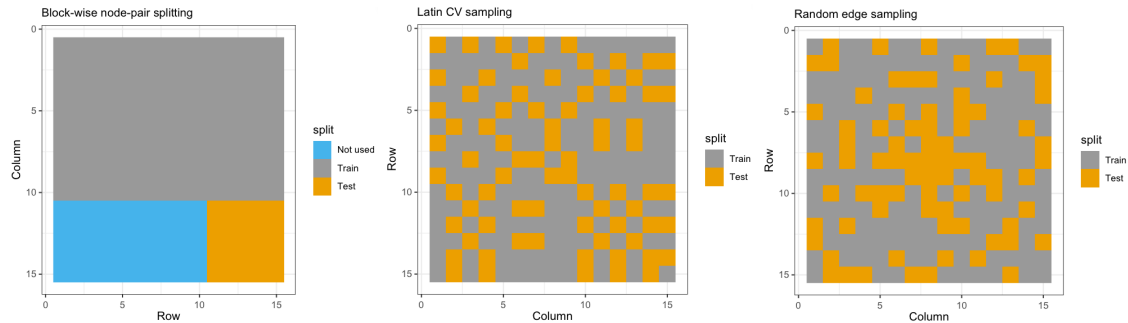


Figure 2.5: Left: block-wise node-pair splitting [CL17]. Middle: LatinCV [DJ16]. Right: random edge sampling [LLZ16].

2.3 Estimation of Stochastic Block Models

Throughout this work, we will need to conduct estimation for stochastic block models as well as degree-corrected stochastic block models given the (partial) adjacency matrix \mathbf{A} of the given network. Spectral clustering [VL07] is one of the mostly widely used method in community detection. It usually starts with eigen-value decomposition of the adjacency matrix \mathbf{A} or the graph Laplacian. Then the community labels will be assigned by running a clustering algorithm such as K-means on rows of the matrix formed by a handful of leading eigen-vectors.

However, spectral clustering cannot be directly applied to our case, especially when we use Algorithm 3 for the train-test splitting, since it would leave us with a rectangular matrix for model estimation. Therefore we use the following rectangular spectral clustering method to estimate the community labels for stochastic block models.

The following is an algorithm for estimating stochastic block models given a rectangular (including square) training adjacency matrix. When we are using block-wise node-pair splitting, this is straightforward since the training set is rectangular, as is shown in Figure 2.5. When we are using LatinCV or random edge sampling, neither the training set nor the testing set is in a well-aligned shape. When using a specific train-test split, we extract the edge values of all node-pairs in the test set, and then 'block out' these node-pairs by setting the corresponding edge values to 0 in the original adjacency matrix. We can then optionally conduct a matrix completion step to this blocked-out version of adjacency matrix. Afterwards, we can feed the square adjacency matrix to the following algorithm.

Algorithm 5 Model estimation for stochastic block models using rectangular spectral clustering, using block-wise node-pair splitting

Data: A rectangular network adjacency matrix \mathbf{A} , number of communities K , number of spectral components d (default choice is $d = K$).

Result: Community labels $\hat{\mathbf{g}} \in \{1, \dots, K\}$. Community interaction matrix $\hat{\mathbf{B}}_{K \times K}$.

1. Conduct singular value decomposition on \mathbf{A} :

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$$

2. Let $\tilde{\mathbf{U}}$ be the $n \times d$ matrix consisting of the first d columns of the singular vectors matrix \mathbf{U} .
3. Run K-means algorithm on row vectors of $\tilde{\mathbf{U}}$ (use multiple starts for stability of result). Set the clustering results to be $\hat{\mathbf{g}}$.
4. Estimate the community interaction matrix $\hat{\mathbf{B}}$ using the plug-in estimator, i.e. by taking the average interaction frequency over the corresponding set in the rectangular training set:

$$\hat{B}_{k,k}^{(K,tr)} = \frac{1}{\binom{n_k^{(K,tr)}}{2} + n_k^{(K,tr)} \cdot n_k^{(K,te)}} \sum_{\substack{(i,j) \in \hat{I}_k^{(K,tr)} \times \hat{I}_k^{(K,tr)} \\ i < j}} A_{ij}$$

$$\hat{B}_{k,k'}^{(K,tr)} = \frac{1}{n_k^{(K,tr)} n_{k'}^{(K,tr)} + n_k^{(K,tr)} \cdot n_{k'}^{(K,te)} + n_{k'}^{(K,tr)} \cdot n_k^{(K,te)}} \sum_{\substack{(i,j) \in \left(\hat{I}_k^{(K,tr)} \times \hat{I}_{k'}^{(K,tr)} \right) \\ \cup \left(\hat{I}_k^{(K,te)} \times \hat{I}_{k'}^{(K,te)} \right) \\ \cup \left(\hat{I}_{k'}^{(K,te)} \times \hat{I}_k^{(K,te)} \right)}} A_{ij}$$

(If we are using LatinCV or random edge sampling, we need to adjust for the down-sampling and use $\frac{V}{V-1}\hat{B}$ as the estimate for community interaction matrix, after making sure all values are within $[0, 1]$.)

5. **(Optional)** Estimate the community interaction matrix $\hat{\mathbf{B}}^{(test)}$ using the community labels and the test set only.

$$\hat{\mathbf{B}}_{k,k}^{(test)} = \frac{1}{\binom{n_k^{(K,te)}}{2}} \sum_{\substack{(i,j) \in \hat{I}_k^{(K,te)} \times \hat{I}_k^{(K,te)} \\ i < j}} A_{ij}$$

$$\hat{\mathbf{B}}_{k,k'}^{(test)} = \frac{1}{n_k^{(K,te)} n_{k'}^{(K,te)}} \sum_{(i,j) \in \hat{I}_k^{(K,te)} \times \hat{I}_{k'}^{(K,te)}} A_{ij}$$

Note that step 5 in the above algorithm is optional. We will only need it when we obtain the sampling distribution of the test statistic via parametric bootstrap. In that case, we will need to generate new network data sets using the existing data set as the null. We will discuss the reasoning of this in more detail later in section 2.6.

When we are estimating the model after getting the training set obtained via either random edge sampling or LatinCV, the training set will be a square adjacency matrix with holes. Certain entries will be assigned to the test set and therefore held out, their values set to 0. We can view this as a training set with missing edge values. One way to handle the situation is to first have a matrix completion step to reconstruct the training network (as proposed in [LLZ16]). We would then obtain the community labels of all nodes in the network by running rectangular spectral clustering on the completed training set. The completed training set, however, will lead to very biased estimates of the community interaction matrix \mathbf{B} . Therefore, when we estimate the community interaction matrix, we will use the edge values in the original training set \mathbf{A} and adjust for the down-sampling from holding out roughly $\frac{1}{V}$ of its existing edges. The version of model estimation algorithm with matrix completion is described in Algorithm 6. In Chapter 3, we will specify whether the algorithm we are using includes the matrix completion step by stating whether the algorithm is with or without matrix completion.

Algorithm 6 Model estimation for stochastic block models using rectangular spectral clustering, using LatinCV or random edge sampling, with matrix completion

Data: A network adjacency matrix \mathbf{A} with certain cells being 'blocked out' (i.e. set to 0), number of communities K , number of spectral components d (default choice is $d = K$).

Result: Community labels $\hat{\mathbf{g}} \in \{1, \dots, K\}$. Community interaction matrix $\hat{B}_{K \times K}$.

1. Conduct low-rank matrix completion on matrix \mathbf{A} and obtain $\hat{\mathbf{A}}$. Retain only the singular vectors up to the second largest singular value gap for \mathbf{A} . Save both $\hat{\mathbf{A}}$ and \mathbf{A} .
2. Conduct singular value decomposition on $\hat{\mathbf{A}}$:

$$\hat{\mathbf{A}} = \mathbf{U}\Sigma\mathbf{V}^T$$

3. Let $\tilde{\mathbf{U}}$ be the $n \times d$ matrix consisting of the first d columns of the singular vectors matrix \mathbf{U} .

4. Run K-means algorithm on row vectors of $\tilde{\mathbf{U}}$ (use multiple starts for stability of result). Set the clustering results to be \hat{g} .
5. Estimate the community interaction matrix $\hat{\mathbf{B}}$ using the plug-in estimator, i.e. by taking the average interaction frequency over the corresponding set in the original adjacency matrix \mathbf{A} and adjust for the down-sampling:

$$\hat{B}_{k,k}^{(K,tr)} = \frac{V}{V-1} \cdot \frac{1}{\binom{n_k^{(K,tr)}}{2} + n_k^{(K,tr)} \cdot n_k^{(K,te)}} \sum_{\substack{(i,j) \in \hat{I}_k^{(K,tr)} \times \hat{I}_k^{(K,te)} \\ i < j}} A_{ij}$$

$$\hat{B}_{k,k'}^{(K,tr)} = \frac{V}{V-1} \cdot \frac{1}{n_k^{(K,tr)} n_{k'}^{(K,tr)} + n_k^{(K,tr)} \cdot n_{k'}^{(K,te)} + n_{k'}^{(K,tr)} \cdot n_k^{(K,te)}} \sum_{\substack{(i,j) \in \left(\hat{I}_k^{(K,tr)} \times \hat{I}_{k'}^{(K,tr)} \right) \\ \cup \left(\hat{I}_k^{(K,te)} \times \hat{I}_{k'}^{(K,te)} \right) \\ \cup \left(\hat{I}_{k'}^{(K,te)} \times \hat{I}_k^{(K,te)} \right)}} A_{ij}$$

6. **(Optional)** Estimate the community interaction matrix $\hat{\mathbf{B}}^{(test)}$ using the community labels and the test set only.

$$\hat{\mathbf{B}}_{k,k}^{(test)} = \frac{1}{\binom{n_k^{(K,te)}}{2}} \sum_{\substack{(i,j) \in \hat{I}_k^{(K,te)} \times \hat{I}_k^{(K,te)} \\ i < j}} A_{ij}$$

$$\hat{\mathbf{B}}_{k,k'}^{(test)} = \frac{1}{n_k^{(K,te)} n_{k'}^{(K,te)}} \sum_{(i,j) \in \hat{I}_k^{(K,te)} \times \hat{I}_{k'}^{(K,te)}} A_{ij}$$

We also need to note that when we are using splitting methods other than the block-wise node-pair splitting, the estimation of community interaction matrix $\hat{B}_{k,k'}^{(K,tr)}$ becomes a little tricky. When we are using random edge sampling, for example, the training set will still be a square matrix, though the entries assigned to the training set will be blocked out, i.e. set to 0. In this case we will optionally do a low-rank matrix completion, using the eigen-values up to the second largest eigen gap, and achieve community detection by running spectral clustering on this matrix. Then after we obtain the community labels, we will need to account for the fact that our interaction probabilities will be underestimated and add a correction term to the estimates. In other words, when we are using LatinCV or random edge sampling, we will use the following community interaction matrix estimator:

$$\hat{\mathbf{B}}_{k,k}^{(test)} = \frac{V}{V-1} \cdot \frac{1}{\binom{n_k^{(K,te)}}{2}} \sum_{\substack{(i,j) \in \hat{I}_k^{(K,te)} \times \hat{I}_k^{(K,te)} \\ i < j}} A_{ij}$$

$$\hat{\mathbf{B}}_{k,k'}^{(test)} = \frac{V}{V-1} \cdot \frac{1}{n_k^{(K,te)} n_{k'}^{(K,te)}} \sum_{(i,j) \in \hat{I}_k^{(K,te)} \times \hat{I}_{k'}^{(K,te)}} A_{ij}$$

Also, when we are working instead with degree-corrected stochastic block models, only need to use the $\tilde{\mathbf{U}}$ obtained by scaling each row of \mathbf{U} to unit norm, and we will get the spherical spectral clustering algorithm. Both algorithms are as mentioned in [CL17].

In the input of this algorithm we see a tuning parameter d . When conducting the rectangular spectral clustering, we take only the first d columns of the singular vectors in matrix U . Selection of value d basically requires us to know the number of meaningful rank in the adjacency matrix \mathbf{A} , which is very close to selection of K . Barring situations where we have extra knowledge about the rank of the adjacency matrix, we likely cannot know this for sure. Two options we have are as follows.

1. Using $d = K$ as the default choice. Since we assume the community has K different communities, the community interaction matrix \mathbf{B} will be a $K \times K$ matrix. Therefore the adjacency matrix should be rank- K , beyond which the variation should be random noise.
2. Weighting each column of the singular vector matrix using the square root of singular values. In other words, use

$$\tilde{\mathbf{U}} = \mathbf{U} \cdot \Sigma^{\frac{1}{2}}$$

instead of \mathbf{U} in step 3. Since the singular values indicate the amount of variation explained by the corresponding rank, the dimensions with relatively low information content will be given lower weight and thus influence the clustering output to a much lesser degree.

2.4 Cross-validation with Confidence

The shortcomings of classical cross-validation include the fact that it doesn't take into account the randomness in test sets while evaluating models. Since the evaluation of models is usually done over one fold assignment (as in Algorithm 1), it is possible that an inferior model can appear better just due to luck. Also, an over-fitting model might also be selected by the procedure since it can have similar performance compared to the true model, and in one random fold assignment, its cross-validation loss could be slightly smaller.

Cross-validation with confidence takes this into account and constructs a hypothesis testing framework. Given a set of candidate models, each model is compared against every other model. For each candidate model, if the hypothesis that none of the other models is significantly better than the current model gets rejected, then it is abandoned, otherwise it is kept in the output confidence set. Eventually, we have a set of models that are not outperformed by any other model significantly. Since the randomness in test set is now addressed, the output set of models would contain the true model with a guaranteed probability. Also, since any under-fitting model will be rejected by the hypothesis test when compared with the optimal model, while any over-fitting model wouldn't look significantly superior compared to the optimal model, the output set will likely contain the optimal model and some over-fitting models. We need only to pick the most parsimonious model out of the output set when we want to select one model using CVC.

Similar to the style of [Lei17], we have the following notations. Given a series of candidate model estimates $\{\hat{f}_m : m \in M\}$ obtained from training data D_{tr} . We evaluate a fitted model \hat{f} by $Q(\hat{f}) = \mathbb{E}(L(\hat{f}(X), Y) | \hat{f})$, where $L(\hat{f}(X), Y)$ denotes a typical loss function. For each $m \in M$, we want to conduct the following hypothesis test:

$$H_{0,m} : Q(\hat{f}_m) \leq Q(\hat{f}_{m'}), \forall m' \neq m$$

vs

$$H_{1,m} : Q(\hat{f}_m) > Q(\hat{f}_{m'}), \text{ for some } m' \neq m$$

Hypothesis $H_{0,m}$ means that the fitted model \hat{f}_m has the best predictive risk among all fitted models, and $H_{1,m}$ means that there exists another fitted model whose predictive risk is strictly less than \hat{f}_m . Given a chosen $\alpha \in (0, 1)$, if we have p-value $\hat{P}_{cv,m}$ for each $m \in M$, then the confidence set

$$\mathbb{A}_{cv} = \{\hat{f}_m : m \in M, \hat{P}_{cv,m} > \alpha\}$$

Fix an $m \in M$, and define random vector $\xi_m = (\xi_{K,K'} : j \neq m)$ as

$$\xi_{K,K'} = L(\hat{f}_m(X), Y) - L(\hat{f}_j(X), Y)$$

Let $\mu_{K,K'} = \mathbb{E}(\xi_{K,K'} | \hat{f}_m, \hat{f}_j)$, then the hypothesis testing problem can be simplified as

$$H_{0,m} : \max_{j \neq m} \mu_{K,K'} \leq 0, \text{ vs } H_{1,m} : \max_{j \neq m} \mu_{K,K'} > 0$$

We evaluate a fitted model \hat{f} by $Q(\hat{f}) = \mathbb{E}(L(\hat{f}(X), Y)|\hat{f})$. The V-fold version of the hypothesis testing problem becomes:

$$H_{0,m} : \frac{1}{V} \sum_{v=1}^V Q(\hat{f}_m^{(v)}) \leq \frac{1}{V} \sum_{v=1}^V Q(\hat{f}_{m'}^{(v)}), \forall m' \neq m$$

$$H_{1,m} : \frac{1}{V} \sum_{v=1}^V Q(\hat{f}_m^{(v)}) > \frac{1}{V} \sum_{v=1}^V Q(\hat{f}_{m'}^{(v)}), \text{ for some } m' \neq m$$

The challenge in this hypothesis testing framework mostly lies in the fact that there might be a large number of correlated candidate models, making the minimum testing problem high-dimensional and correlated. Also, $\xi_{K,K'}$'s and $\xi_{K,K''}$'s might be on a different scale. Fortunately, Gaussian multiplier bootstrap proposed in [CCK⁺13] provides us with the appropriate tool to handle these challenges and makes it possible to define a consistent test statistic. The following algorithm from [Lei17] describes the procedure in more details:

Algorithm 7 V-fold cross validation with confidence for prediction problem

Data: Data points $\mathbb{D} = \{(x_i, y_i), i = 1, 2, \dots, n\}$, let $\mathbb{N} = \{1, \dots, n\}$ be the index set corresponding to each data point. A loss function we want to minimize for predictions $L(y, \hat{y})$. A set of candidate models to select from $\{m_1, m_2, \dots, m_k\}$. A p-value threshold α_n .

Result: A confidence set \mathbb{A}_{cv} that will include the true model with guaranteed probability.

1. Randomly split the dataset into V equal-sized subsets $\{\mathbb{D}_v : 1 \leq v \leq V\}$, each fold would correspond to a subset of the index set \mathbb{N}_v using one of Algorithm 2, 3, and 4. Optionally, conduct low-rank matrix completion before estimating the stochastic block model.
2. For each $m \in \{m_1, m_2, \dots, m_k\}$
 - 2.1. For all $j \in \{m_1, m_2, \dots, m_k\} \setminus \{m\}$, for $v \in \{1, \dots, V\}$,

$$\xi_{K,K'}^{(i)} = L(\hat{f}_m^{(v)}(x_i), y_i) - L(\hat{f}_j^{(v)}(x_i), y_i), i = 1, 2, \dots, n$$

- 2.2. $\hat{\mu}_{K,K'}^{(v)} = \frac{V}{n} \sum_{i \in \mathbb{N}_v} \xi_{K,K'}^{(i)}$, for all $j \neq m, 1 \leq i \leq n$. These $\hat{\mu}_{K,K'}^{(v)}$'s are the estimated group mean effect.

- 2.3. $\tilde{\xi}_{K,K'}^{(i)} = \xi_{K,K'}^{(i)} - \hat{\mu}_{K,K'}^{(v)}$, for all $j \neq m, 1 \leq i \leq n$. These $\tilde{\xi}_{K,K'}^{(i)}$'s are the group-wise centered difference of cross-validated predictive loss.

- 2.4. Let $\hat{\mu}_{K,K'}$ and $\hat{\sigma}_{K,K'}$ be the overall mean effect and the sample standard deviation of $\{\tilde{\xi}_{K,K'}^{(i)} : 1 \leq i \leq n\}$.

- 2.5. Let $T = \max_{j \neq m} \sqrt{n} \frac{\hat{\mu}_{K,K'}}{\hat{\sigma}_{K,K'}}$.

- 2.6. For $b = 1, \dots, B$:

- i. Generate i.i.d standard normal random variables $\zeta_i, 1 \leq i \leq n$.
- ii. Let

$$T_b^* = \max_{j \neq m} \frac{1}{\sqrt{n}} \sum_{v=1}^V \sum_{i \in \mathbb{N}_v} \frac{\xi_{K,K'}^{(i)} - \hat{\mu}_{K,K'}^{(v)}}{\hat{\sigma}_{K,K'}} \zeta_i$$

- 2.7. $\hat{p}_{cv,m} = \frac{1}{B} \sum_{b=1}^B \mathbb{I}(T_b^* > T)$. If $\hat{p}_{cv,m} < \alpha_n$, then at least one other model $j \neq m$ is significantly better than m in the sense that $\xi_{K,K'}$ has a positive expectation. Then m is eliminated from the candidate set.
3. Return the confidence set of all models that were not eliminated: $\mathbb{A}_{cv} = \{m : m \in \{m_1, \dots, m_k\}, \hat{P}_{cv,m} > \alpha_n\}$.

[Lei17] showed that under certain assumptions, when we select $\alpha_n \in (\frac{1}{n}, 1)$, the true model will be included in the confidence set with probability at least $1 - \alpha_n + o(1)$ and the probability that an under-fitting model will be included is $o(1)$. Therefore, we can pick the most parsimonious model in the confidence set and it will very likely be the true model we are looking for.

2.5 V-fold Cross-validation with Confidence in the Context of Stochastic Block Models

In this thesis we want to extend cross-validation with confidence to selecting number of communities in stochastic block models. The adaptation is mostly straightforward, though it becomes tricky and requires some decision on our part. For example, we need to make a choice between the train-test splitting methods under network context as discussed in section 2.2. Also, as we will later see in the theory section, we may want to change the normalization of the loss difference slightly for convenience of our later theoretical study into the properties of the method.

Given a model and a series of candidate models that we want to compare it to, we want to evaluate the difference loss function for each node of the network between any two models. When we are dealing with a node in fold $v \in \{1, 2, \dots, V\}$, we use the information of nodes in the other $V - 1$ folds (including their interaction with the i th fold) for estimating community labels and interaction probabilities. With the estimated model, we can then make prediction for nodes in the i th fold and evaluate the loss function on nodes in the fold for the corresponding model.

In the context of networks, here we define the calculation of p-value $\hat{P}_{ss,m}$ by block-wise node-pair splitting. For a network represented by an $n \times n$ symmetric binary adjacency matrix A , we randomly split the nodes into V equal-sized subsets $\{\mathbb{N}_v : 1 \leq v \leq V\}$, and split the adjacency matrix correspondingly into $V \times V$ equal sized blocks:

$$A = \{A^{(uv)} : 1 \leq u, v \leq V\}$$

$$A^{(uv)} = \{(i, j), i \in \mathbb{N}_u, j \in \mathbb{N}_v\}$$

$A^{(uv)}$ is the sub-matrix of A with rows in \mathbb{N}_u and columns in \mathbb{N}_v .

We describe the cross-validation with confidence method in selecting number of communities in stochastic block models using block-wise node-pair splitting and Gaussian multiplier bootstrap in the following algorithm:

Algorithm 8 V-fold block-wise node-pair splitting cross validation with confidence

Data: $n \times n$ symmetric binary adjacency matrix A , candidate set of possible values of community number $\{K : K \in \mathbb{K}\}$, loss function l , cross validation fold number V , P-value threshold $\alpha_n \in (0, 1)$, bootstrap sample size B .

Result: Confidence set \mathbb{A}_{cv}

1. Randomly split the nodes into V equal-sized subsets $\{\mathbb{N}_v : 1 \leq v \leq V\}$, and split the adjacency matrix correspondingly into $V \times V$ equal sized blocks:

$$A = (\hat{A}^{(uv)} : 1 \leq u, v \leq V)$$

2. For each $K \in \mathbb{K}$, and each $1 \leq v \leq V$:

- (a) Using Algorithm 5, estimate model parameters $(\hat{\mathbf{g}}^{(K,-v)}, \hat{\mathbf{B}}^{(K,-v)})$ using community number K using the rectangular sub matrix obtained by removing the rows of A in subset \mathbb{N}_v :

$$\tilde{A}^{(-v)} = \{A_{ij} : i \notin \mathbb{N}_v\}$$

With the same approach, we can estimate the model parameters corresponding to other values in \mathbb{K} , for comparison purposes in later steps.

(b) For all $K' \in \mathbb{K} \setminus \{K\}$, and $i, j \in \bar{\mathbb{N}}_v$

$$\xi_{K,K'}^{(i,j)} = L(A_{ij}, \hat{B}_{ij}^{(K,-v)}) - L(A_{i,jij}, \hat{B}_{ij}^{(K',-v)})$$

where $\hat{B}_{ij}^{(K,-v)}$ is the edge probability estimate for node pair (i, j) in the test set $\bar{\mathbb{N}}_v \times \bar{\mathbb{N}}_v$ using number of communities K and $\tilde{A}^{(-v)}$ as the training set.

(c) Let $\hat{\mu}_{K,K'}^{(v)}$ and $\hat{M}_{K,K'}^{(v)}$ be the sample mean and uncentered second moment of $\{\xi_{K,K'}^{(ij)} : i, j \in \bar{\mathbb{N}}_v\}$.

(d) Let

$$T = \max_{j \neq m} \sqrt{\frac{n^2}{V^2}} \cdot \frac{\hat{\mu}_{K,K'}}{\hat{M}_{K,K'}} \quad (2.1)$$

(e) For $b = 1, \dots, B$, generate iid standard Normal variables $\zeta_{ij}, i, j \in \bar{\mathbb{N}}_v$, let

$$T_b^* = \max_{K' \neq K} \sqrt{\frac{n^2}{V^2}} \sum_{v \in \{1, \dots, V\}} \sum_{i, j \in \bar{\mathbb{N}}_v} \frac{\xi_{K,K'}^{(i,j)} - \hat{\mu}_{K,K'}}{\hat{M}_{K,K'}} \zeta_{ij}$$

(f) $\hat{P}_{cv,m} = \frac{1}{B} \sum_{b=1}^B \mathbf{1}(T_b^* > T)$.

3. Return $\mathbb{A}_{cv} = \{m : m \in M, \hat{P}_{cv,m} > \alpha_n\}$

As mentioned in Chapter 1, there are some variants of the standard stochastic block models which are more flexible and appropriate for networks with more heterogeneity among its individual nodes. For example, degree-corrected stochastic block models allow nodes in the same community to have different degree-correction parameters, such that each node's connection pattern would not only depend on its community affiliation but also its own level of activity in the network. Mixed-membership stochastic block models allow each node to have membership in several communities with certain weights. These models add more flexibility to the networks they can depict and fit real-world data sets better than the standard stochastic block model.

When we observe Algorithm 8 closely, it is easy to realize that it isn't necessarily limited to the standard stochastic block models. In fact, the only step that is specific to the stochastic block model is the estimation step in step 2(a). For regular stochastic block models, we usually estimate the community labels using spectral clustering of the rectangular adjacency matrix, then obtain the edge probability by calculating the empirical edge frequency between communities. Therefore, Algorithm 8 can be very easily extended to degree-corrected stochastic block models (DC-SBM). We just need to notice that in estimating degree-corrected stochastic block models people usually use spherical projection before conducting spectral clustering for degree corrected models to accommodate effect of the individual degree correction parameters.

Also, we can include both standard stochastic block models and degree-corrected stochastic block models as candidate models, and select the confidence set as models not rejected in the hypothesis testing step. One caveat we have for this method is that since there will possibly be candidate K 's left from both categories, there might not be a clear-cut most parsimonious

model in the confidence set. There are possible cases to make a comparison between them (for example, since degree-corrected stochastic block models are more flexible, a standard stochastic block model with the same number of communities should be seen as more parsimonious than its degree-corrected counterpart). We describe this algorithm that allows us to select among both standard stochastic block models and degree-corrected stochastic block models in Algorithm 9.

Algorithm 9 V-fold block-wise node-pair splitting cross validation with confidence, selecting among both SBMs and DC-SBMs

Data: $n \times n$ symmetric binary adjacency matrix A , candidate set of possible values of community number $\{K : K \in \mathbb{K}\}$, loss function l , cross validation fold number V , P-value threshold $\alpha_n \in (0, 1)$, bootstrap sample size B .

Result: Confidence set \mathbb{A}_{cv}

1. Randomly split the nodes into V equal-sized subsets $\{\mathbb{N}_v : 1 \leq v \leq V\}$, and split the adjacency matrix correspondingly into $V \times V$ equal sized blocks:

$$A = (\hat{A}^{(uv)} : 1 \leq u, v \leq V)$$

2. For each $K \in \mathbb{K}$, model type in $\{\text{SBM}, \text{DC-SBM}\}$, and each $1 \leq v \leq V$:
 - (a) Using Algorithm 5 or its extension to degree-corrected stochastic block models, estimate model parameters $(\hat{\mathbf{g}}^{(K,-v)}, \hat{\mathbf{B}}^{(K,-v)})$ using community number K using the rectangular sub matrix obtained by removing the rows of A in subset \mathbb{N}_v :

$$\tilde{A}^{(-v)} = \{A_{ij} : i \notin \mathbb{N}_v\}$$

With the same approach, we can estimate the model parameters corresponding to other values in \mathbb{K} , for comparison purposes in later steps.

- (b) For all $K' \in \mathbb{K} \setminus \{K\}$ with the same model type, and all $K' \in \mathbb{K}$ with the other model type, and $i, j \in \bar{\mathbb{N}}_v$

$$\xi_{K,K'}^{(i,j)} = L(A_{ij}, \hat{B}_{ij}^{(K,-v)}) - L(A_{i,jij}, \hat{B}_{ij}^{(K',-v)})$$

where $\hat{B}_{ij}^{(K,-v)}$ is the edge probability estimate for node pair (i, j) in the test set $\mathbb{N}_v \times \mathbb{N}_v$ using number of communities K and $\tilde{A}^{(-v)}$ as the training set.

- (c) Let $\hat{\mu}_{K,K'}^{(v)}$ and $\hat{M}_{K,K'}^{(v)}$ be the sample mean and uncentered second moment of $\{\xi_{K,K'}^{(ij)} : i, j \in \bar{\mathbb{N}}_v\}$.
 - (d) Let

$$T = \max_{j \neq m} \sqrt{\frac{n^2}{V^2}} \cdot \frac{\hat{\mu}_{K,K'}}{\hat{M}_{K,K'}} \quad (2.2)$$

- (e) For $b = 1, \dots, B$, generate iid standard Normal variables $\zeta_{ij}, i, j \in \bar{\mathbb{N}}_v$, let

$$T_b^* = \max_{K' \neq K} \sqrt{\frac{n^2}{V^2}} \sum_{v \in \{1, \dots, V\}} \sum_{i, j \in \bar{\mathbb{N}}_v} \frac{\xi_{K,K'}^{(i,j)} - \hat{\mu}_{K,K'}}{\hat{M}_{K,K'}} \zeta_{ij}$$

$$(f) \hat{P}_{cv,m} = \frac{1}{B} \sum_{b=1}^B \mathbf{1}(T_b^* > T).$$

3. Return $\mathbb{A}_{cv} = \{m : m \in M, \hat{P}_{cv,m} > \alpha_n\}$

We will implement and explore the properties of these extended algorithms in Chapter 3.

In this algorithm we use the uncentered second moment of $\xi_{K,K'}$'s to normalize their mean, which is slightly different in how the normalization is done in Section 2.4. This is mostly out of convenience for our later work in Chapter 4. Remember that the purpose of the normalization is to bring all the ξ 's back to the same scale, using the uncentered second moment should still be able to achieve the goal reasonably well, and we will check its influence on the practical effectiveness to the algorithm by comparing different versions using both the standard deviation and uncentered second moment for scaling in Section 3.

Let $\hat{I}_{k,k'} = \{(i, j) : (\hat{g}_i, \hat{g}_j) = (k, k'), i > j\}$ denote the pairs (i, j) where i is assigned to community k and j is assigned to community k' . The extra squared error loss on cell (i, j) between using community number K_1, K_2 is denoted as:

$$\xi_{i,j}^{(K_1, K_2)} = (A_{ij} - \hat{B}_{\hat{g}_i^{(K_1)}, \hat{g}_j^{(K_1)}}^{(K_1, tr)})^2 - (A_{ij} - \hat{B}_{\hat{g}_i^{(K_2)}, \hat{g}_j^{(K_2)}}^{(K_2, tr)})^2 \quad (2.3)$$

The negative likelihood extra loss on (i, j) would be:

$$\xi_{i,j}^{(K_1, K_2)} = \left(-A_{ij} \log \hat{B}_{\hat{g}_i^{(K_1)}, \hat{g}_j^{(K_1)}}^{(K_1, tr)} - (1 - A_{ij}) \log(1 - \hat{B}_{\hat{g}_i^{(K_1)}, \hat{g}_j^{(K_1)}}^{(K_1, tr)}) \right) \quad (2.4)$$

$$- \left(-A_{ij} \log \hat{B}_{\hat{g}_i^{(K_2)}, \hat{g}_j^{(K_2)}}^{(K_2, tr)} - (1 - A_{ij}) \log(1 - \hat{B}_{\hat{g}_i^{(K_2)}, \hat{g}_j^{(K_2)}}^{(K_2, tr)}) \right) \quad (2.5)$$

Both loss functions are very often used in the network literature. In our work, we will explore both loss functions in the simulation study and concentrate on the squared error loss in our theoretical studies.

In Chapter 4 we will try to show similar results about the guaranteed probabilities for inclusion of different models in \mathbb{A}_{cv} . More specifically, we will show that an under-fitting model is guaranteed (with probability $\rightarrow 1$) to be eliminated in the hypothesis testing procedure, while the true model will be included in \mathbb{A}_{cv} when we select $\alpha_n \in (\frac{1}{n}, 1)$. Therefore, if we pick the most parsimonious model in \mathbb{A}_{cv} (or simply, the smallest K value), we will likely have the true K value for the network.

Note on the regularity conditions for Gaussian multiplier bootstrap: In [CCK⁺13], there are certain regularity conditions required for the Gaussian multiplier bootstrap results to be valid, i.e. for the reference distribution to be a reasonable approximation to the true sampling distribution of the test statistic. In other words, we want the maximum of a normalized sample mean of the extra prediction loss to have a similar distribution as the maximum of a Gaussian random vector with the same covariance structure. In Algorithm 8, we are using the normalized difference in prediction losses and therefore the tail conditions will not be guaranteed for each normalized ξ 's. Therefore we will not use the Gaussian comparison to control the type I error probability. Instead, we will use concentration inequalities to control the probability of type I errors, as will be seen in our work in Chapter 4.

2.6 Using Parametric Bootstrap instead of Gaussian Multiplier Bootstrap

When we observe the problem close enough, we would find that in algorithm 7, we are using Gaussian Multiplier Bootstrap largely because we cannot obtain the sampling distribution of ξ 's in a generative way. However, in our context, it is in fact possible to obtain the sampling distribution of the test statistic under the null by generating new network data sets with certain parameters. we can use the results of this approach to check the correctness of Gaussian Multiplier Bootstrap.

In other words, in Algorithm 8, after calculating the test statistic using Equation 2.6, we can obtain the sampling distribution of the test statistic by directly generating network data sets under the null hypothesis. Recall that the null hypothesis is that the current community number K is not significantly inferior to other candidate community values, meaning that it could possibly be the true number of communities. We will generate a series of adjacency matrices using the model parameters estimated for the current K : $(\hat{\mathbf{g}}^{(K,-v)}, \hat{\mathbf{B}}^{(K,-v)})$. Let $\mathbf{A}^{(b)}$, $b = 1, \dots, B$ denote these adjacency matrices.

After obtaining such adjacency matrices under the null hypothesis, we can similarly split $\mathbf{A}^{(b)}$ into V folds and calculate the cross validation loss of corresponding to a model of K communities. We would then be able to calculate ξ 's, and thus get a sample of the test statistic T_b^* under the null hypothesis. We can then calculate the P-value by comparing our realization of the test statistic to this sample. We describe this procedure in more detail in the following algorithm.

Algorithm 10 V-fold block-wise node-pair splitting cross validation with confidence, via parametric bootstrap

Data: $n \times n$ symmetric binary adjacency matrix \mathbf{A} , candidate set of possible values of community number $\{K : K \in \mathbb{K}\}$, loss function l , cross validation fold number V , P-value threshold $\alpha_n \in (0, 1)$, bootstrap sample size B .

Result: Confidence set $\hat{\mathbb{A}}_{cv}$

1. Randomly split the nodes into V equal-sized subsets $\{\mathbb{N}_v : 1 \leq v \leq V\}$, and split the adjacency matrix correspondingly into $V \times V$ equal sized blocks:

$$\mathbf{A} = (\hat{A}^{(uv)} : 1 \leq u, v \leq V)$$

2. For each $K \in \mathbb{K}$, and each $1 \leq v \leq V$:
 - (a) Using Algorithm 5, estimate model parameters $(\hat{\mathbf{g}}^{(K,-v)}, \hat{\mathbf{B}}^{(K,-v)})$ using community number K using the rectangular sub matrix obtained by removing the rows of \mathbf{A} in subset \mathbb{N}_v :

$$\tilde{\mathbf{A}}^{(-v)} = \{A_{ij} : i \notin \mathbb{N}_v\}$$

Also estimate the community interaction matrix using only the test set $\hat{\mathbf{B}}^{(K,-v,test)}$. With the same approach, we can estimate the model parameters corresponding to other values in \mathbb{K} , for comparison purposes in later steps.

- (b) For all $K' \in \mathbb{K} \setminus \{K\}$, and $i, j \in \bar{\mathbb{N}}_v$

$$\xi_{K,K'}^{(i,j)} = L(A_{ij}, \hat{B}_{ij}^{(K,-v)}) - L(A_{i,jij}, \hat{B}_{ij}^{(K',-v)})$$

where $\hat{B}_{ij}^{(K,-v)}$ is the edge probability estimate for node pair (i, j) in the test set $\mathbb{N}_v \times \mathbb{N}_v$ using number of communities K and $\tilde{A}^{(-v)}$ as the training set.

- (c) Let $\hat{\mu}_{K,K'}^{(v)}$ and $\hat{M}_{K,K'}^{(v)}$ be the sample mean and uncentered second moment of $\{\xi_{K,K'}^{(ij)} : i, j \in \bar{\mathbb{N}}_v\}$.

3. Let

$$T_K = \max_{j \neq m} \sqrt{\frac{n^2}{V^2}} \cdot \frac{\hat{\mu}_{K,K'}}{\hat{M}_{K,K'}} \quad (2.6)$$

This is the test statistic for the null hypothesis that no alternative model is significantly outperforming the model with K .

4. Using the estimated community labels $\hat{\mathbf{g}}^{(K,-v)}$ and estimated community interaction matrix over the test set $\hat{\mathbf{B}}^{(K,-v,test)}$, we can generate test sets with the given K value, and obtain new loss functions for each candidate K , and thus get bootstrap samples for test statistic $T_K^b, b = 1, 2, \dots, B$.
5. $\hat{P}_{cv,m} = \frac{1}{B} \sum_{b=1}^B \mathbf{1}(T_K^b > T_K)$.
6. Return $\mathbb{A}_{cv} = \{m : m \in M, \hat{P}_{cv,m} > \alpha_n\}$

Note that here the sampling distribution is the distribution of the test statistic assuming that $K^* = K$. However, this is hard to achieve since we cannot evenly go through all the cases of \mathbf{B} and \mathbf{g} given $K^* = K$. The compromise we are making is to assume that the true K is the current value, and the true community interaction matrix is the community interaction matrix estimated using only the test set $\hat{\mathbf{B}}^{(K,-v,test)}$.

Readers might notice that we are generating bootstrap samples of the test set using the community interaction matrix estimated on the original test set $\hat{\mathbf{B}}^{(K,-v,test)}$, instead of the community interaction matrix estimated over the training set. We are going this extra step because when calculating the test statistic, we need to obtain the cross-validation loss using the model estimates (obtained from the training set) and the observations over the (bootstrap sample of) test set. If we are generating the bootstrap samples of test set using the model estimates obtained from the training set, then the cross-validation loss would merely be the unavoidable random noise between the ground truth and realizations of Bernoulli random variables. However, the cross-validation loss should also include the generalization loss, i.e. the discrepancy between the model estimated obtained using the training set and the observations from the test set, since the estimation algorithm has never 'seen' the test set. If we omit this important source of the cross-validation loss, we will grossly underestimate the cross-validation loss under the null hypothesis and thus only get a very biased hypothesis test. In fact, we will likely reject all candidate K 's, since the problematic sampling distribution of the test statistic will be entirely smaller than the actual realization of the test statistic.

When we are working with block-wise node-pair splitting, for each bootstrap iteration, we expand the community labels $\hat{\mathbf{g}}^{(K,-v)}$ into a full matrix form $\mathbf{G}_{n \times K}^{(K,-v)}$ such that

$$\mathbf{G}_{i,j}^{(K,-v)} = \text{liff.} \hat{g}_i^{(K,-v)} = j$$

where $i \in \{1, \dots, \binom{n^{(te)}}{2}\}, j \in \{1, \dots, K\}$.

Combine it with the estimated community interaction matrix over the test set $\hat{\mathbf{B}}^{(K,-v,test)}$ to obtain an edge probability prediction over the test set:

$$\hat{\mathbf{P}}^{(K,-v,test)} = \mathbf{G}^{(K,-v)} \cdot \hat{\mathbf{B}}^{(K,-v,test)} \cdot \left(\mathbf{G}^{(K,-v)}\right)^T$$

This matrix is of size $n^{(te)} \times n^{(te)}$. For each cell in its upper triangle, we generate a Bernoulli random variable using the corresponding value in the edge probability prediction. After copying over the upper triangle over to the lower triangle, we arrive at a symmetric adjacency matrix the same size as our test set under the null hypothesis. And we can calculate its test statistic value and use it as a bootstrap sample.

When we are working with random edge sampling, or LatinCV sampling, the test set won't be in a neat square shape. Instead, it will be a list of row and column coordinates such as the one shown in Table 2.1. We will loop through each row of the table and get the edge probability prediction by calculating

$$\hat{\mathbf{B}}_{\hat{g}_i^{(tr,K)}, \hat{g}_j^{(tr,K)}}^{(K,-v,test)}$$

	Row index	Column index
1	1	2
2	2	3
3	1	5
4	3	5
5	2	6

Table 2.1: Example of a test set using LatinCV. Here we are splitting all node pair into 3 fold in a network of 6 nodes.

Another note we want to add is about the computational speed. When we are working with Gaussian multiplier bootstrap, we only need to calculate the cross-validation error matrix once. Then we can calculate the test statistic for each candidate K and obtain the bootstrap samples of these test statistics by operating on the cross-validation loss matrix. We can benefit from the optimizations for matrix operations since these calculations are mostly vectorized. When we are working with parametric bootstrap, we need to generate a new bootstrap sample of the test set each time. Then we calculate the entire cross-validation error matrix using this bootstrap sample of the test set and the model estimates, and then calculate the test statistic for this bootstrap sample. To get each sample of the test statistic, we need to go through the process of simulating new test sets, obtaining cross-validation error matrix, and calculating test statistic. This is a longer cycle compared to the Gaussian multiplier bootstrap alternative and thus will be computationally more expensive.

Chapter 3

Simulation Studies and Application on Data Sets

In this chapter we will examine the effectiveness and properties of our method and compare it with similar alternatives in the literature, by implementing them and testing them on simulated and real-world data sets.

3.1 Studies through simulated data

3.1.1 Simulation of adjacency matrices from Stochastic Block Models

Given a set of parameters of a network, we can generate an instance of its adjacency matrices according to the Stochastic Block Model settings. The parameters we will specify characterize the connectivity matrix B between communities, the size of each community, the edge probability between different nodes.

To be specific, our simulation algorithm takes in the following arguments: the true community label vector g with K different values (typically ranging from 1 to K), the value q that characterizes how diagonally dominant the connectivity matrix is, the density parameter ρ . Note that here we can have a subroutine for generating g , for example, we can simply generate the following vector that contains equal number of members for each community:

$$\mathbf{g} = (1, \dots, 1, 2, \dots, 2, \dots, K, \dots, K)$$

Then we turn this community label vector into a matrix:

$$\mathbf{G}_{n \times K} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \dots & \dots & 0 \\ 1 & \ddots & \dots & \dots & 0 \\ 0 & 1 & \dots & \dots & 0 \\ \vdots & \vdots & \dots & \ddots & \vdots \\ 0 & 1 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 1 \end{bmatrix}$$

We define the following B_0 as the prototype of the connectivity matrix:

$$B_0 = \begin{bmatrix} 1 & \frac{1}{q+1} & \dots & \frac{1}{q+1} \\ \frac{1}{q+1} & 1 & \dots & \frac{1}{q+1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{q+1} & \frac{1}{q+1} & \dots & 1 \end{bmatrix}$$

This formation of B is not the only scheme we will be testing, although we will use it in many cases for its simplicity when B is not the central part of parameters we want to test our method for. In general, the community interaction matrix can be any symmetric matrix that doesn't have any identical rows and has all of its values fall within $(0, 1)$. We will use more general settings for generating B in section 3.1.9.

The larger q is, the better separation there is in the network in the sense that nodes are way more likely to be connected with other members within the community rather than with members from other communities.

The true community connectivity matrix is then defined as:

$$B = \rho \cdot B_0$$

And the edge probability matrix is as follows:

$$P = \mathbf{G} \cdot B \cdot \mathbf{G}^T = \begin{bmatrix} 1 & B_{g_1, g_2} & B_{g_1, g_3} & \dots & B_{g_1, g_n} \\ B_{g_2, g_1} & 1 & B_{g_2, g_2} & \dots & B_{g_2, g_n} \\ B_{g_3, g_1} & B_{g_3, g_2} & 1 & \dots & B_{g_3, g_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{g_n, g_1} & B_{g_n, g_2} & B_{g_n, g_3} & \dots & 1 \end{bmatrix} \quad (3.1)$$

Then the adjacency matrix A is generated from the corresponding Bernoulli distributions:

$$A_{ij} \sim \text{Bernoulli}(P_{ij})$$

For generating data sets from degree-corrected stochastic block models, we would only need to add one more step. Define the degree correction parameter vector θ , where

$$\theta_i \sim \text{Uniform}(\text{lo}, \text{hi}) \quad , \quad 0 < \text{lo} < \text{hi} \leq 1$$

And simply substitute \mathbf{G} with $\text{diag}(\theta) \cdot \mathbf{G}$ in Equation 3.1, where $\text{diag}(\theta)$ is the diagonal matrix whose diagonal line is θ_i .

3.1.2 Illustration of the Hypothesis Testing Step

In [Lei17], it is shown that cross-validation with confidence can select the true model with guaranteed probability. This is due to the fact that the hypothesis testing framework can reject the under-fitting models as under-performing, yet the over-fitting model won't outperform the true model significantly. Therefore if we select the most parsimonious model among the models not rejected, we would very likely end up with the true model. Ideally we would hope after being extended to stochastic block model context, cross-validation with confidence would still retain this desirable property.

We set up the simulations as following. In each simulation, we generate a network of 1200 nodes with K ($K \in \{2, 3, \dots, 7\}$) true communities with equal size and various network density levels ρ . We then conduct the hypothesis test for each candidate $K \in \{1, 2, \dots, 10\}$ and record the P-value given by the Gaussian multiplier bootstrap test. Figure 3.1 shows the results from these simulations. We can see that the P-value would typically be very low for under-fitting models, meaning that they would likely be rejected. While it would have a sharp rise at the true K and then go down again. When the true number of communities is small, even the over-fitting models will be rejected in the test. Yet when the true number of communities is larger, some over-fitting models will remain after the test, although the true model would also get retained. For example, in Figure 3.1, when $K = 4$ and the network density is 0.025, the confidence set $\mathbb{A}_{cv} = \{4\}$ only contains the true K value, while when the density is 0.3, we can see the confidence set becomes $\{4, 5, 6\}$, where two over-fitting models are retained. Although as long as we pick the most parsimonious model (i.e. the smallest number of communities) in the confidence set, we will still get back the true K value 4. Similar observations can be made about the case when true K value is 6.

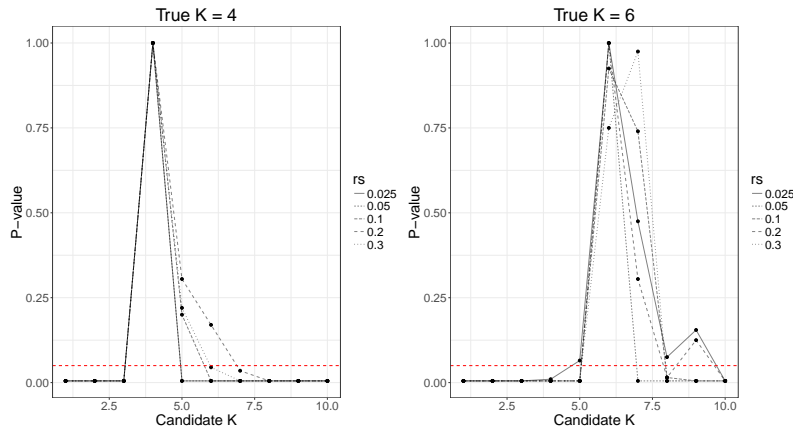


Figure 3.1: P-values when testing whether the candidate K is rejected by CVC. Data sets are generated to be networks with 1200 nodes with true $K = 4, 6$) and different density ($r \in \{0.025, 0.05, 0.1, 0.15, 0.2\}$).

Theses are mostly illustrations of how Cross-validation with Confidence would work under stochastic block model context. In the coming section, we will conduct more thorough experiments and in particular compare it with other established methods in the literature.

3.1.3 Effect of the hypothesis testing step: comparison with BIC and NCV

The first comparison we make is under the setting where all communities have the equal size, we simulate adjacency matrices according to the Stochastic Block Models with different true number of communities, and different network densities. Then we conduct model selection on them using the Bayesian Information Criterion (as proposed in [HRT07], which we will call BIC from here on), two versions of Network Cross Validation and four versions of Cross-validation with Confidence (remember we have two scaling options and two loss functions). Under each community number/network density combination, we repeat the experiment 200 times and compare the percentage of selecting under-fitting, correct, and over-fitting models by each method. The results for standard Stochastic Block Models are summarized numerically in Table 3.1 and visually in Table 3.2. In the table, rows represent different network density parameters r and columns represent true number of communities K^* . The tuples in the table represents percentage of under-fitting, correct, over-fitting model selection results. For example, $(0.2, 0.5, 0.3)$ means 20% of the time the method chooses an under-fitting model, 50% of the time it chooses the true model, and 30% of the time it chooses an over-fitting model.

we also make the same comparison between Network Cross Validation and Cross-validation with Confidence for degree-corrected stochastic block models, the results are shown numerically in Table 3.3 and visually in Table 3.4.

In general we see that our method yields at least comparable or better performance compared to NCV, while it yields comparable results as BIC (as proposed in [HRT07]). The advantage of CVC is especially clear when we are in the nice parameters settings, where the network density is high and number of true communities relatively high. In these settings, NCV tends to select under-fitting models, while CVC would make mostly correct model selections. CVC doesn't perform very well when we are dealing with low-density networks, especially when there are many true communities. The intuition behind this is likely due to the signal being very weak in the data set. Since there is no clear separation between true communities, the under-fitting model can get away with merging some true communities.

Another comparison we make is in the case where the community are not evenly sized. We consider the situations where the true number of communities $K^* = 2, 3, 4$ and set the community-wise edge probability matrix $B = rB_0$, where B_0 has diagonal values 3 and off-diagonal values 1. We used sparsity levels of $r \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$. In other words, for a network with $n = 1000$ nodes, the smallest expected degree would be between 12 and 400. We use the size of the smallest community n_1 as a parameter to control community size imbalance. We assume the other $K - 1$ communities have the same size. When $K = 4$, $n_1 = 250$ means that all communities are equally sized, while if $n_1 = 100$, then the other three communities will each contain 300 nodes, making the network pretty imbalanced. For each true K value, we pick several combinations of edge sparsity and community size imbalance level, generate 200 datasets, and check the performance of CVC. The results are shown in Figure 3.2.

For convenience of comparison we also attach Figure 3.3 results from [CL17]. We can see that our method is an improvement in some cases, in the sense that the correct model is selected with higher chance. For example, when $K = 2$ and $K = 4$, the correct selection percentage by our method is higher than that given by both versions of NCV.

	2	3	4	5	6	7
0.025	BIC	(0,1,0)	(0,1,0)	(0,19,0.81,0)	(1,0,0)	(1,0,0)
	NCV: ls	(0,1,0)	(0,1,0)	(0.13,0.765,0.105)	(0.975,0.02,0.005)	(1,0,0)
	NCV: likelihood	(0,1,0)	(0,1,0)	(0.085,0.825,0.09)	(0.98,0.01,0.01)	(1,0,0)
	CVC: ls	(0,1,0)	(0.005,0.995,0)	(0.735,0.265,0)	(1,0,0)	(1,0,0)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0.71,0.29,0)	(0.995,0.005,0)	(1,0,0)
0.05	BIC	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)	(0.93,0.07,0)
	NCV: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0.02,0.98,0)	(1,0,0)
	NCV: likelihood	(0,1,0)	(0,0.995,0.005)	(0,0.995,0.005)	(0.01,0.99,0)	(1,0,0)
	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0.285,0.715,0)	(0.975,0.025,0)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0.22,0.775,0.005)	(0.985,0.015,0)
0.1	BIC	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	NCV: ls	(0,1,0)	(0,1,0)	(0,0.97,0.03)	(0,0.96,0.04)	(1,0,0)
	NCV: likelihood	(0,1,0)	(0,0.995,0.005)	(0,0.965,0.035)	(0,0.955,0.045)	(1,0,0)
	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.995,0.005)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.995,0.005)	(0,1,0)
0.15	BIC	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	NCV: ls	(0,1,0)	(0,0.985,0.015)	(0,0.97,0.03)	(0,0.945,0.055)	(1,0,0)
	NCV: likelihood	(0,1,0)	(0,0.995,0.005)	(0,0.98,0.02)	(0,0.955,0.045)	(1,0,0)
	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.955,0.045)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.995,0.005)	(0,0.925,0.075)
0.2	BIC	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	NCV: ls	(0,1,0)	(0,0.995,0.005)	(0,0.95,0.05)	(0,0.925,0.075)	(1,0,0)
	NCV: likelihood	(0,1,0)	(0,0.99,0.01)	(0,0.975,0.025)	(0,0.935,0.065)	(1,0,0)
	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.87,0.13)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.855,0.145)

Table 3.1: Percentage of under-fitting, correct and over-fitting model selection results for NCV and CVC methods, under different network density parameter and true community numbers. Data sets in this table are generated using the standard Stochastic Block Model. NCV tends to select under-fitting models when we are dealing with more true communities, while CVC performs much better. In low-density network data sets, CVC tends to select under-fitting models.







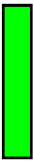
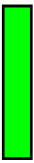
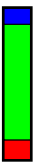



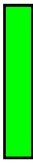
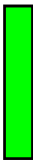
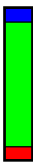



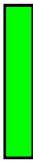
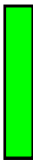










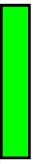
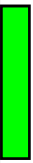
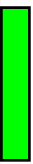
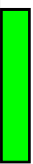
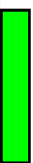

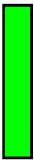
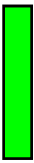
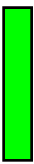
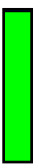
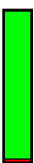

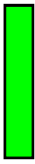
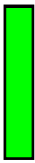
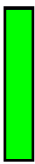
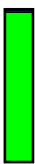
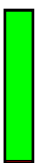

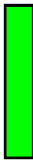
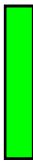
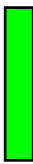
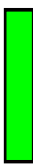
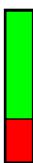













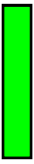
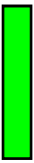
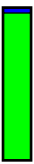
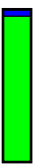
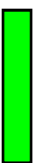

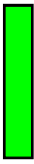
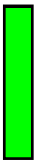
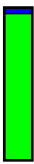
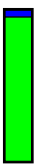
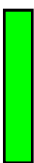

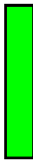
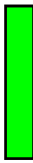
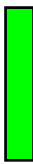
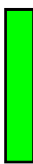
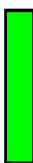
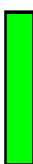












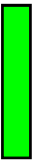
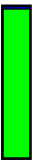
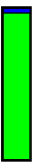
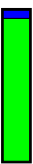
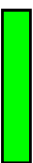

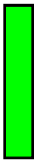
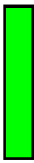
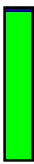
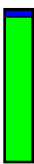
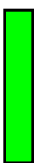



















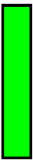
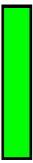
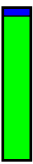
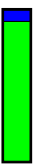
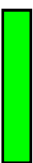

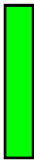
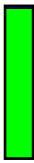
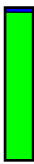
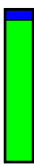
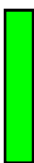













		2	3	4	5	6	7
0.025	BIC						
	NCV: ls						
	NCV: likelihood						
	CVC: ls						
	CVC: likelihood						
0.05	BIC						
	NCV: ls						
	NCV: likelihood						
	CVC: ls						
	CVC: likelihood						
0.1	BIC						
	NCV: ls						
	NCV: likelihood						
	CVC: ls						
	CVC: likelihood						
0.15	BIC						
	NCV: ls						
	NCV: likelihood						
	CVC: ls						
	CVC: likelihood						
0.2	BIC						
	NCV: ls						
	NCV: likelihood						
	CVC: ls						
	CVC: likelihood						

Table 3.2: Percentage of under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) model selection results for NCV and CVC methods, under different network density parameter and true community numbers. Data sets in this table are generated using the standard Stochastic Block Model. NCV tends to select under-fitting models when we are dealing with more true communities, while CVC performs much better. In low-density network data sets, CVC tends to select under-fitting models.

		2	3	4	5
0.05	BIC	(0,1,0)	(1,0,0)	(1,0,0)	(1,0,0)
	NCV: ls	(0.005,0.995,0)	(0.23,0.2,0.57)	(0.76,0.08,0.16)	(0.95,0.025,0.025)
	NCV: likelihood	(0,1,0)	(0.16,0.145,0.695)	(0.43,0.055,0.515)	(0.625,0.19,0.185)
	CVC: ls	(0.305,0.695,0)	(1,0,0)	(1,0,0)	(1,0,0)
	CVC: likelihood	(0.06,0.94,0)	(0.93,0.07,0)	(1,0,0)	(1,0,0)
0.1	BIC	(0,1,0)	(0,1,0)	(0.09,0.91,0)	(0.995,0.005,0)
	NCV: ls	(0,1,0)	(0.0,99,0.01)	(0.32,0.68)	(0.37,0.035,0.595)
	NCV: likelihood	(0,1,0)	(0,1,0)	(0.01,0.75,0.24)	(0.59,0.085,0.325)
	CVC: ls	(0,1,0)	(0.01,0.99,0)	(0.585,0.415,0)	(0.975,0.02,0.005)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0.22,0.78,0)	(0.995,0.005,0)
0.15	BIC	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	NCV: ls	(0,1,0)	(0.0,99,0.01)	(0.0,895,0.105)	(0.0,58,0.42)
	NCV: likelihood	(0,1,0)	(0,1,0)	(0.0,965,0.035)	(0.0,76,0.24)
	CVC: ls	(0,1,0)	(0,1,0)	(0.0,99,0.01)	(0.025,0.875,0.1)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0.0,985,0.015)
0.2	BIC	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	NCV: ls	(0,1,0)	(0,1,0)	(0.0,935,0.065)	(0.0,695,0.305)
	NCV: likelihood	(0,1,0)	(0,1,0)	(0.0,96,0.04)	(0.0,845,0.155)
	CVC: ls	(0,1,0)	(0,1,0)	(0.0,99,0.01)	(0.0,895,0.105)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0.0,965,0.035)
0.3	BIC	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	NCV: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0.0,805,0.195)
	NCV: likelihood	(0,1,0)	(0,1,0)	(0.0,97,0.03)	(0.0,715,0.285)
	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0.0,985,0.015)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0.0,915,0.085)

Table 3.3: Percentage of under-fitting, correct and over-fitting model selection results for NCV and CVC methods, under different network density parameter and true community numbers. Data sets in this table are generated using the standard Stochastic Block Model. We can see CVC outperforms NCV in many scenarios, especially when we have a high network density and high number of communities (bottom right corner of the table). In these cases NCV tends to select over-fitting models, while CVC selects the correct model the majority of times. Although we observe some cases where CVC is underperforming compared to NCV, for example, in the top right corner of the table, when we have relatively high number of communities with low network densities. Cross-validation with confidence tends to select under-fitting models in this case.

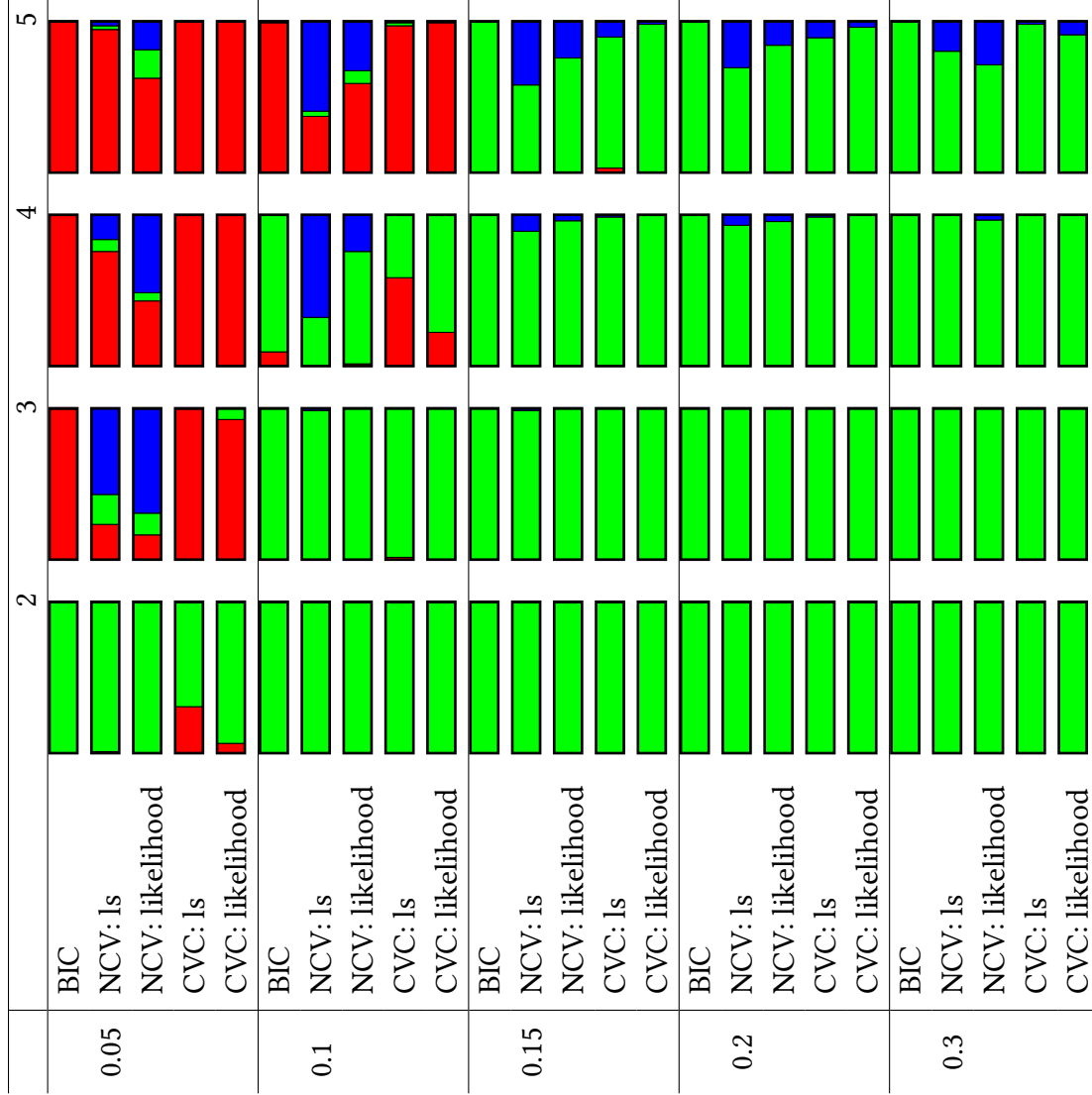


Table 3.4: Percentage of under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) model selection results for NCV and CVC methods, under different network density parameter and true community numbers. Data sets are generated using the standard Stochastic Block Model. We can see CVC outperforms NCV in many scenarios, especially when we have a high network density and high number of communities (bottom right corner of the table). In these cases NCV tends to select over-fitting models, while CVC selects the correct model the majority of times. Although we observe in the top right corner of the table, when we have relatively high number of communities with low network densities, NCV outperforms CVC. Cross-validation with confidence tends to select under-fitting models in this case.

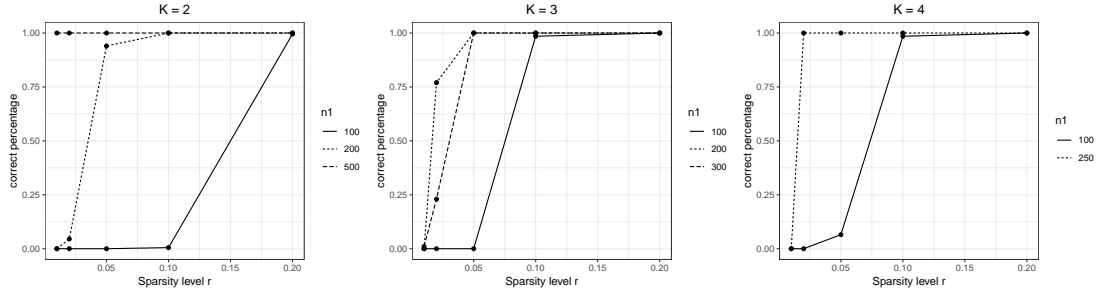


Figure 3.2: Results for simulation. The plots show the proportions of 200 simulated datasets for which K is correctly estimated. The datasets are generated from SBMs with $K = 2, 3, 4$, sparsity levels $r \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$, and various levels of community imbalance. n_1 represents the size of the first community with other community sizes being equal. Total number of nodes of 1000.

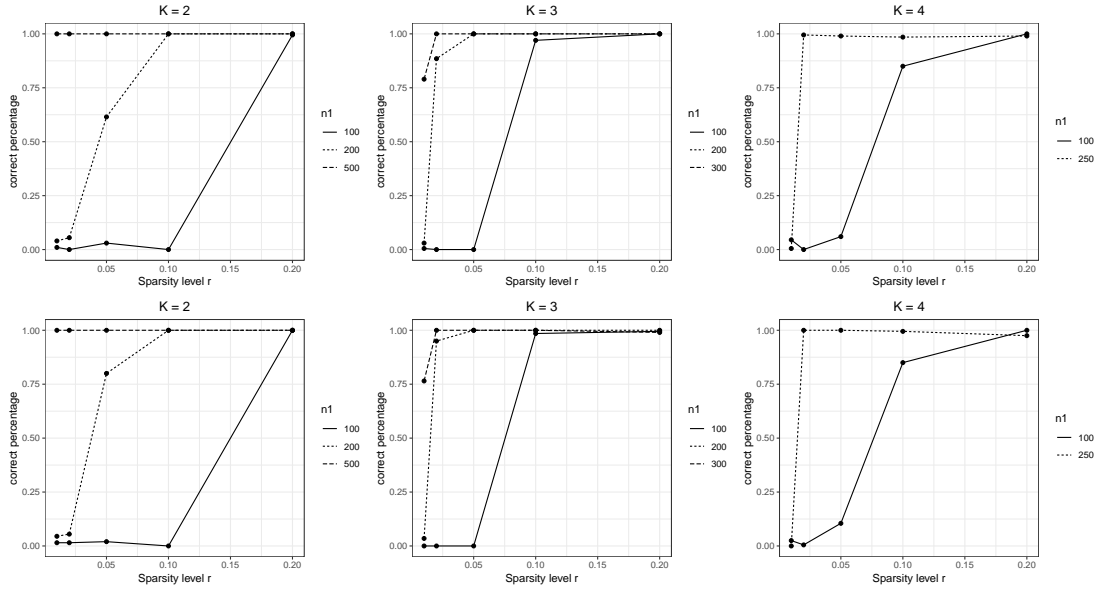


Figure 3.3: The same results, but using Network Cross-Validation proposed in [CL17]. The datasets are generated from SBMs with $K = 2, 3, 4$, sparsity levels $r \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$, and various levels of community imbalance. n_1 represents the size of the first community with other community sizes being equal. Total number of nodes of 1000. The top plot uses the likelihood function as the objective while the bottom plot uses squared error loss as the objective function.

3.1.4 Sensitivity to mis-specification

This subsection is motivated by the question of a committee member during the thesis proposal. He would like to know how sensitive our method would be against mis-specified models. In particular, if the data set is generated using a more general and flexible model, such as the degree-corrected stochastic block model, or mixed membership stochastic block model, how would our method perform if we just treat the data set as a realization of standard stochastic block model? Intuitively, we would see the answer depends on how 'close' the mis-specified model is to the assumption we are making. However, it is not entirely clear what parameter ranges would correspond to a close enough approximation by the standard stochastic block model. We will set out to explore this problem in this section.

in this section we explore this question by applying our method for standard Stochastic Block Models, while using the model selection procedures for standard Stochastic Block Models on them. Here we are considering evenly-sized communities with changing number of communities, changing network density parameter values, and changing range for the distribution of the degree correction parameters in the degree-corrected stochastic block models. We then compare the selected K value with the true value and check the percentage of under-fitting, correct, and over-fitting selections. Table 3.5 - 3.7 shows the results of these experiments. We added a more illustrative representation of the results in Table 3.8 - 3.10. Area of different colors represent the percentage of under-fitting (light red on the left of green), correct (green), and over-fitting (blue on right of green) model selection results.

From the results we see that there are certain settings where our method works well even under mis-specification. When the degree correction parameter has a narrow range (sampled from $\text{Uniform}(0.8, 1)$) our method performs really well. The correct selection percentage is very high ($> 95\%$) except for the case of low-density network with five communities. When the degree correction parameter is sampled from $\text{Uniform}(0.5, 1)$, our method performs decent. There are many under-fitting models selected when we are working with low-density networks and more than 3 true communities. The performance is the worst when the degree correction parameter has a wide range ($\text{Uniform}(0.2, 1)$). In this case we see lots of over-fitting selections when we are working with high-density networks, while on low-density networks we tend to select under-fitting models.

The intuition for these results is also easy to understand, standard Stochastic Block Models can be seen as the limit of Degree Corrected Stochastic Block Models with the distribution of degree correction parameters concentrated to one point. The 'degree' of mis-specification would depend on how close the actual distribution is to this limit. From our experiment we see that when the degree correction parameters have a relatively narrow distribution, stochastic block models are a good enough approximation for degree-corrected stochastic block models, therefore using our method designed for standard SBMs would still yield a reasonable performance for DC-SBMs.

		2	3	4	5
0.05	CVC: ls	(0,1,0)	(0.395,0.49,0.115)	(0.965,0.03,0.005)	(1,0,0)
	CVC: likelihood	(0,1,0)	(0.155,0.595,0.25)	(0.71,0.12,0.17)	(0.96,0.025,0.015)
0.1	CVC: ls	(0,1,0)	(0,0.51,0.49)	(0,0.055,0.945)	(0.065,0.155,0.78)
	CVC: likelihood	(0,1,0)	(0,0.74,0.26)	(0,0.07,0.93)	(0.005,0.09,0.905)
0.15	CVC: ls	(0,0.995,0.005)	(0,0.355,0.645)	(0,0,1)	(0,0.005,0.995)
	CVC: likelihood	(0,0.995,0.005)	(0,0.57,0.43)	(0,0.015,0.985)	(0,0.005,0.995)
0.2	CVC: ls	(0,1,0)	(0,0.39,0.61)	(0,0,1)	(0,0,1)
	CVC: likelihood	(0,1,0)	(0,0.575,0.425)	(0,0,1)	(0,0,1)

Table 3.5: Under-fitting, correct, and over-fitting percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.2, 1).

		2	3	4	5
0.05	CVC: ls	(0,1,0)	(0,1,0)	(0.32,0.68,0)	(1,0,0)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0.27,0.725,0.005)	(0.99,0.01,0)
0.1	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
0.15	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.99,0.01)
	CVC: likelihood	(0,1,0)	(0,0.995,0.005)	(0,1,0)	(0,0.985,0.015)
0.2	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.985,0.015)

Table 3.6: Under-fitting, correct, and over-fitting percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.5, 1).

		2	3	4	5
0.05	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0.295,0.7,0.005)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0.205,0.795,0)
0.1	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
0.15	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.99,0.01)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,0.995,0.005)	(0,0.985,0.015)
0.2	CVC: ls	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	CVC: likelihood	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.99,0.01)

Table 3.7: Under-fitting, correct, and over-fitting percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.75, 1).






















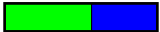










		2	3	4	5
0.05	CVC: ls				
	CVC: likelihood				
0.1	CVC: ls				
	CVC: likelihood				
0.15	CVC: ls				
	CVC: likelihood				
0.2	CVC: ls				
	CVC: likelihood				

Table 3.8: Under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.2, 1).

































		2	3	4	5
0.05	CVC: ls				
	CVC: likelihood				
0.1	CVC: ls				
	CVC: likelihood				
0.15	CVC: ls				
	CVC: likelihood				
0.2	CVC: ls				
	CVC: likelihood				

Table 3.9: Under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.5, 1).

































		2	3	4	5
0.05	CVC: ls				
	CVC: likelihood				
0.1	CVC: ls				
	CVC: likelihood				
0.15	CVC: ls				
	CVC: likelihood				
0.2	CVC: ls				
	CVC: likelihood				

Table 3.10: Under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) percentage for mis-specified models when degree correction parameters are sampled from Uniform(0.75, 1).

3.1.5 One case of unidentifiability

An interesting special case we are interested in is the case where a data set can be seen as being generated from a degree-corrected stochastic block models, or a more complicated standard stochastic block model.

For example, if we have a standard stochastic block model with the following community interaction matrix:

$$B^* = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{8} \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

It can be seen as a degree-corrected stochastic block model with only one community but two types of nodes, the more active ones have degree-correction parameters three times as much as the less active ones.

$$\Theta = (1, 1, \dots, 1, \frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2})^T$$

We ran our algorithm for several such B^* 's and with different network densities. We repeated the experiment under each setting for 200 times to see how stable the results are under random fold assignment. We compared our algorithm (4 versions) against the network cross-validation algorithm.

In our first experiment, we use

$$B_0 = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix} \tag{3.2}$$

as the prototype of the true community interaction matrix. For different values of $\rho \in \{0.05, 0.1, 0.2, 0.4, 0.6\}$, we generate network data sets with equally sized communities and interaction matrix $B = \rho \cdot B_0$. Then we use different model selection methods on the data set to check their effectiveness. Table 3.11 - 3.12 shows the percentage of each candidate $K \in \{1, 2, \dots, 6\}$ being selected by the algorithm assuming standard stochastic block model (using different loss functions).

CVC using squared error loss and $d = 1$						
$\rho \setminus \hat{K}$	1	2	3	4	5	6
0.05	0.135	0.86	0.005	0	0	0
0.1	0	1	0	0	0	0
0.2	0	1	0	0	0	0
0.4	0	1	0	0	0	0
0.6	0	1	0	0	0	0

Table 3.11: Proportion of each candidate K being selected using CVC method with squared error loss using $d = 2$, under various network density levels. The column for $\hat{K} = 2$ is highlighted since it is the correct number of communities assuming **standard stochastic block model** (according to Equation 3.2).

CVC using squared error loss and spectral weighting						
$\rho \setminus \hat{K}$	1	2	3	4	5	6
0.05	0.88	0.12	0	0	0	0
0.1	0	0.95	0.04	0.01	0	0
0.2	0	0.98	0.02	0	0	0
0.4	0	0.99	0.01	0	0	0
0.6	0	1	0	0	0	0

Table 3.12: Proportion of each candidate K being selected using CVC method with squared error loss and weighting each column of the singular vector matrix \mathbf{U} by the square-root of the corresponding singular value d_i , under various network density levels. The column for $\hat{K} = 2$ is highlighted since it is the correct number of communities assuming **standard stochastic block model** (according to Equation 3.2).

It is clear that the results are not very satisfactory, the correct number of communities is very rarely selected even when have relatively high network density. The selection results are highly unstable over repeated runs, showing great variability under random fold assignment.

Table 3.13 shows the same percentage using CVC method for degree-corrected stochastic block models. In this case the correct number of communities would be $K^* = 1$ since we can account for the difference in degree distribution by different level of degree correction parameter alone. It is clear that the CVC method for degree-corrected stochastic block models is very stable and consistent. The only case where it doesn't pick the correct number of communities all the time is when we use it under low network density ($\rho = 0.05$) with the negative likelihood loss function.

Using squared error loss							Using negative likelihood loss						
$\rho \setminus \hat{K}$	1	2	3	4	5	6	$\rho \setminus \hat{K}$	1	2	3	4	5	6
0.05	1	0	0	0	0	0	0.05	0.145	0.855	0	0	0	0
0.1	1	0	0	0	0	0	0.1	1	0	0	0	0	0
0.2	1	0	0	0	0	0	0.2	1	0	0	0	0	0
0.4	1	0	0	0	0	0	0.4	1	0	0	0	0	0
0.6	1	0	0	0	0	0	0.6	1	0	0	0	0	0

Table 3.13: Proportion of each candidate K being selected using CVC method with different loss functions, under various network density levels. The column for $\hat{K} = 1$ is highlighted since it is the correct number of communities assuming **degree-corrected stochastic block model** (according to Equation 3.2).

For the purpose of comparison, we also ran the same experiments using network cross-validation methods as proposed in [CL17]. It is also clear that when assuming standard stochastic block models, the algorithm isn't very good at selecting the correct number of communities. The method for degree-corrected stochastic block models, on the other hand, performs very well. We list the results for the latter (NCV for DC-SBM) in Table 3.14. Note that most of the time it has very good performance, although in low density situations it under-performs its counterpart in our work.

NCV for DC-SBM, with squared error loss							NCV for DC-SBM, with negative likelihood loss						
$\rho \setminus \hat{K}$	1	2	3	4	5	6	$\rho \setminus \hat{K}$	1	2	3	4	5	6
0.05	0.995	0.005	0	0	0	0	0.05	0.005	0.825	0.09	0.03	0.015	0.035
0.1	1	0	0	0	0	0	0.1	0.93	0.035	0.005	0	0.01	0.02
0.2	1	0	0	0	0	0	0.2	1	0	0	0	0	0
0.4	1	0	0	0	0	0	0.4	1	0	0	0	0	0
0.6	1	0	0	0	0	0	0.6	1	0	0	0	0	0

Table 3.14: Proportion of each candidate K being selected using NCV method with different loss functions, under various network density levels. The column for $\hat{K} = 1$ is highlighted since it is the correct number of communities assuming **degree-corrected stochastic block model** (according to Equation 3.2).

We can also apply Algorithm 9 to these data sets. We summarize the frequency of each model being retained in the confidence set in Table 3.15. It is clear that when we have higher network density, the DC-SBM with 1 community is getting more and more likely to be selected, even over the equally correct model, i.e. standard stochastic block model with $K = 2$.

ρ	SBM.1	SBM.2	SBM.3	SBM.4	SBM.5	DCBM.1	DCBM.2	DCBM.3	DCBM.4	DCBM.5
0.05	0.00	0.00	0.00	0.00	0.00	0.41	0.97	0.90	0.86	0.74
0.1	0.00	0.00	0.00	0.00	0.02	0.19	0.99	0.69	0.33	0.06
0.15	0.00	0.00	0.20	0.21	0.89	0.00	0.74	0.35	0.12	0.02
0.2	0.00	0.00	0.30	0.42	0.66	0.00	0.00	0.00	0.00	0.00
0.25	0.00	0.00	0.57	0.64	0.69	0.00	0.00	0.00	0.00	0.00

Table 3.15: Frequency for each model to be selected using Algorithm 9. The data sets here are generated using prototype community interaction matrix as defined in Equation 3.2.

We repeat the same experiments as above, but for a new prototype community interaction matrix:

$$B_0 = \begin{bmatrix} 1 & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & 1 & \frac{1}{3} \\ \frac{1}{8} & \frac{1}{3} & \frac{1}{9} \end{bmatrix} \quad (3.3)$$

This model would again have different interpretation under different models. When we assume it comes from the standard stochastic block model, then it has 3 true communities. While when we use degree-corrected stochastic block model, it can be modeled using a 2-community network by noting that

$$\begin{bmatrix} 1 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{9} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{3} \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{1}{3} \end{bmatrix}$$

Here the second community needs to contain two kinds of nodes, the more active ones have degree correction parameter three times as much as the less active ones. We summarize our results in Table 3.16, 3.17, and 3.18.

CVC using squared error loss and $d = 2$						
$\rho \setminus \hat{K}$	1	2	3	4	5	6
0.05	0	0.01	0.90	0.09	0	0
0.1	0	0	1	0	0	0
0.2	0	0	1	0	0	0
0.4	0	0	1	0	0	0
0.6	0	0	1	0	0	0

Table 3.16: Proportion of each candidate K being selected using CVC method with squared error loss, under various network density levels. The column for $\hat{K} = 3$ is highlighted since it is the correct number of communities assuming **standard stochastic block model** (according to Equation 3.3).

$\rho \setminus \hat{K}$	1	2	3	4	5	6
0.05	0	0.44	0.54	0.02	0	0
0.1	0	0	0.98	0.015	0.01	0
0.2	0	0	0.99	0	0.01	0
0.4	0	0	1	0	0	0
0.6	0	0	1	0	0	0

Table 3.17: Proportion of each candidate K being selected using CVC method with squared error loss, under various network density levels. The column for $\hat{K} = 3$ is highlighted since it is the correct number of communities assuming **standard stochastic block model** (according to Equation 3.3).

$\rho \setminus \hat{K}$	1	2	3	4	5	6
0.05	0.06	0.94	0	0	0	0
0.1	0	1	0	0	0	0
0.2	0	1	0	0	0	0
0.4	0	1	0	0	0	0
0.6	0	1	0	0	0	0

Table 3.18: Proportion of each candidate K being selected using CVC method with different loss functions, under various network density levels. The column for $\hat{K} = 2$ is highlighted since it is the correct number of communities assuming **degree-corrected stochastic block model** (according to Equation 3.3).

When we apply Algorithm 9 to these data sets. We summarize the frequency of each model being retained in the confidence set in Table 3.19. Here the trend is a little different. When we are dealing with lower density network data sets, degree-corrected model with $K = 2$ is selected with very high probability, while standard stochastic block models with $K = 3$ has a low yet increasing frequency of being retained. When we increase the network density parameter above 0.2, all degree-corrected models start to get rejected, while the standard stochastic block models with $K = 3$ is retained in the confidence set with increasingly high frequency.

ρ	SBM.1	SBM.2	SBM.3	SBM.4	SBM.5	DCBM.1	DCBM.2	DCBM.3	DCBM.4	DCBM.5
0.05	0.00	0.00	0.01	0.00	0.00	0.00	0.99	0.41	0.17	0.06
0.1	0.00	0.00	0.21	0.29	0.60	0.00	0.99	0.43	0.38	0.12
0.2	0.00	0.00	0.58	0.79	0.71	0.00	0.00	0.00	0.00	0.00

Table 3.19: Frequency for each model to be selected using Algorithm 9. The data sets here are generated using prototype community interaction matrix as defined in Equation 3.3.

With the above examples we see that for the constructed examples of un-identifiable network data sets, cross-validation with confidence can find the best approximation within the range of models it is given. Its performance improves as we are dealing with higher density networks, as expected, since the spectral clustering results become closer to the actual community labels.

One intriguing observation we have for both examples is that when we increase the network density parameter above a certain level, Algorithm 9 starts to only leave standard stochastic block models in the confidence set. We think this goes back to the model estimation step for degree-corrected stochastic block models. For degree-corrected stochastic block models, we project the truncated (or weighted) U matrix row vectors onto a sphere and then get community labels by running clustering algorithms on these unit vectors. Then we will estimate both the community interaction matrix B and the degree-correction parameter vector Θ . The estimation of Θ will bring in an extra source of noise, since each node is allowed its own degree-correction parameter value. When the network density parameter is low, the estimator for B matrix is noisy even for standard stochastic block models, which makes the more flexible degree-corrected models still comparable. Yet when density parameter is high enough and the standard stochastic block models can be more accurate, the degree-corrected models might have worse performance due to the extra variance introduced in the degree-correction parameter estimation.

3.1.6 Sensitivity to different community interaction matrix setting

In Section 3.1.1, we set the prototype of community interaction matrix as

$$B_0 = \begin{bmatrix} 1 & \frac{1}{q+1} & \cdots & \frac{1}{q+1} \\ \frac{1}{q+1} & 1 & \cdots & \frac{1}{q+1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{q+1} & \frac{1}{q+1} & \cdots & 1 \end{bmatrix} \quad (3.4)$$

where we set q to be a integer like 2 or 3. This guarantees that the true community interaction matrix is diagonally dominant and each row is different from each other. In this section we want to see how well our method would work if we relax these conditions. In fact, the cases in Section 3.1.5 can be seen as one example of not diagonally dominant community interaction matrix.

Here we want to see how the performance of cross-validation with confidence is influenced when we make the community interaction matrix 'less' diagonally dominant. In the following experiments, we set $q = 2, 1, \frac{1}{2}, \frac{1}{4}$. As the value of q gets smaller, the prototype community interaction matrix B_0 becomes less and less diagonally dominant. We can therefore see how the CVC method would perform. We set the network densities to $\rho = 0.075, 0.15, 0.3, 0.45, 0.6$ and generate network data sets using community interaction matrix $B = \rho \cdot B_0$, where B_0 is set as Equation 3.4.

In Table 3.20 - 3.27 we summarize the results of these experiments. For each combination of number of communities K^* , diagonal dominance parameter Q and network density ρ , we generate 200 data sets and apply our method. Table 3.20 - 3.23 contain the case where each network has 300 total nodes, and Table 3.24 - 3.27 contain the case where each network has 600 total nodes.

$\rho \setminus K^*$	2	3	4	5
0.075	(0.245,0.755,0)	(0.995,0.005,0)	(1,0,0)	(1,0,0)
0.15	(0,1,0)	(0.025,0.975,0)	(0.81,0.19,0)	(1,0,0)
0.3	(0,1,0)	(0,1,0)	(0,1,0)	(0.005,0.995,0)
0.45	(0,1,0)	(0,1,0)	(0,0.995,0.005)	(0,0.995,0.005)
0.6	(0,1,0)	(0,1,0)	(0,1,0)	(0,0.995,0.005)

Table 3.20: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 2$. Each community is equally sized. The entire network has **300** nodes.

$\rho \setminus K^*$	2	3	4	5
0.075	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.15	(0.1,0.9,0)	(0.99,0.01,0)	(1,0,0)	(1,0,0)
0.3	(0,1,0)	(0,1,0)	(0.565,0.435,0)	(1,0,0)
0.45	(0,1,0)	(0,1,0)	(0,1,0)	(0.15,0.84,0.01)
0.6	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)

Table 3.21: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 1$. Each community is equally sized. The entire network has **300** nodes.

$\rho \setminus K^*$	2	3	4	5
0.075	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.15	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.3	(0.175,0.825,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.45	(0,1,0)	(0.25,0.75,0)	(1,0,0)	(1,0,0)
0.6	(0,1,0)	(0,1,0)	(0.23,0.77,0)	(1,0,0)

Table 3.22: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{2}$. Each community is equally sized. The entire network has **300** nodes.

$\rho \setminus K^*$	2	3	4	5
0.075	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.15	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.3	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.45	(0.95,0.05,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.6	(0.06,0.94,0)	(1,0,0)	(1,0,0)	(1,0,0)

Table 3.23: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{4}$. Each community is equally sized. The entire network has **300** nodes.

$\rho \setminus K^*$	2	3	4	5
0.075	(0,1,0)	(0,1,0)	(0.73,0.27,0)	(1,0,0)
0.15	(0,1,0)	(0,1,0)	(0,1,0)	(0.005,0.99,0.005)
0.3	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
0.45	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
0.6	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)

Table 3.24: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 2$. Each community is equally sized. The entire network has **600** nodes.

	2	3	4	5
0.075	(0.025,0.975,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.15	(0,1,0)	(0,1,0)	(0.655,0.345,0)	(1,0,0)
0.3	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
0.45	(0,1,0)	(0,0.995,0.005)	(0,1,0)	(0,0.99,0.01)
0.6	(0,1,0)	(0,0.995,0.005)	(0,1,0)	(0,0.99,0.01)

Table 3.25: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 1$. Each community is equally sized. The entire network has **600** nodes.

	2	3	4	5
0.075	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.15	(0.26,0.74,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.3	(0,1,0)	(0,1,0)	(0.93,0.07,0)	(1,0,0)
0.45	(0,1,0)	(0,1,0)	(0,1,0)	(0.38,0.61,0.01)
0.6	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)

Table 3.26: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{2}$. Each community is equally sized. The entire network has **600** nodes.

	2	3	4	5
0.075	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.15	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.3	(0.915,0.085,0)	(1,0,0)	(1,0,0)	(1,0,0)
0.45	(0,1,0)	(0.945,0.055,0)	(1,0,0)	(1,0,0)
0.6	(0,1,0)	(0,1,0)	(0.945,0.055,0)	(1,0,0)

Table 3.27: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{4}$. Each community is equally sized. The entire network has **600** nodes.

When we compare the experiments among networks with the same size (i.e. among Table 3.20 - 3.23 and Table 3.24 - 3.27), we see that as q increases (and the community interaction matrix becomes less diagonally dominant), the accuracy of our algorithm decreases. Especially when $q = \frac{1}{4}$, it would select under-fitting model most of the time. The only exception is when we have relatively few true communities (2 or 3) and high network densities ($\rho = 0.45, 0.6$). When we compare between network data sets of different sizes (for example, between Table 3.23 and Table 3.27), we see that when we hold the community interaction matrix constant and increase the network size, the accuracy increases. This is also very intuitive. As we increase the size of network, the data set contains more information, and the estimation results tend to more closely approximate the true data generating process.

For better illustrative effect, we can refer to the following tables as alternatives of Table 3.20 - 3.23 and Table 3.24 - 3.27. Area of different colors represent the percentage of under-fitting (light red on the left of green), correct (green), and over-fitting (blue on right of green) model selection results.





















$\rho \setminus K^*$	2	3	4	5
0.075				
0.15				
0.3				
0.45				
0.6				

Table 3.28: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 2$. Each community is equally sized. The entire network has **300** nodes.





















$\rho \backslash K^*$	2	3	4	5
0.075				
0.15				
0.3				
0.45				
0.6				

Table 3.29: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 1$. Each community is equally sized. The entire network has **300** nodes.





















$\rho \backslash K^*$	2	3	4	5
0.075				
0.15				
0.3				
0.45				
0.6				

Table 3.30: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{2}$. Each community is equally sized. The entire network has **300** nodes.





















$\rho \backslash K^*$	2	3	4	5
0.075				
0.15				
0.3				
0.45				
0.6				

Table 3.31: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{4}$. Each community is equally sized. The entire network has **300** nodes.





















$\rho \backslash K^*$	2	3	4	5
0.075				
0.15				
0.3				
0.45				
0.6				

Table 3.32: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 2$. Each community is equally sized. The entire network has **600** nodes.





















	2	3	4	5
0.075				
0.15				
0.3				
0.45				
0.6				

Table 3.33: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = 1$. Each community is equally sized. The entire network has **600** nodes.





















	2	3	4	5
0.075				
0.15				
0.3				
0.45				
0.6				

Table 3.34: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{2}$. Each community is equally sized. The entire network has **600** nodes.





















	2	3	4	5
0.075				
0.15				
0.3				
0.45				
0.6				

Table 3.35: Under-fitting, correct, and over-fitting percentage for CVC method using squared error loss. Here each data set is generated using prototype community interaction matrix as defined in Equation 3.4, where $q = \frac{1}{4}$. Each community is equally sized. The entire network has **600** nodes.

We can also examine the effectiveness of our algorithm on community interaction matrices with more general structures. One such structure we are looking at is as in [CL17], deterministic prototype community interaction matrices B_0 where all off-diagonal elements are 2, while the diagonal elements are $(3, 1), (3, 2, 1), (3, 3, 1, 1), (3, 3, 2, 1, 1)$ for $K = 2, 3, 4, 5$. And the true community interaction matrices are $\rho \cdot B_0$, where $\rho \in \{0.2, 0.3, 0.4, 0.5, 0.6\}$. Now that the interaction matrix is no longer diagonal dominant, the network density required for satisfactory results becomes higher. We summarize our results numerically in Table 3.36 and visually in Table 3.37.

$\rho \setminus K$	2	3	4	5
0.15	(0.795,0.105,0.1)	(0.085,0.915,0)	(0.94,0.05,0.01)	(1,0,0)
0.2	(0.02,0.515,0.465)	(0,1,0)	(0.13,0.855,0.015)	(0.955,0.04,0.005)
0.25	(0,1,0)	(0,1,0)	(0,1,0)	(0.81,0.175,0.015)
0.3	(0,1,0)	(0,0.995,0.005)	(0,1,0)	(0.305,0.62,0.075)

Table 3.36: Under-fitting, correct, and over-fitting model selection percentages for running Algorithm 8 on network data sets generated on deterministic and non-diagonal-dominant community interaction matrices.

















$\rho \setminus K$	2	3	4	5
0.15				
0.2				
0.25				
0.3				

Table 3.37: Under-fitting (red), correct(green), and over-fitting(blue) model selection percentages for running Algorithm 8 on network data sets generated on deterministic and non-diagonal-dominant community interaction matrices.

Another community interaction matrix structure we tested out is the random community interaction matrix. We generate random prototype community interaction matrices using algo-

rithm 11. This algorithm will always generate a random matrix, where the each row and each column will have enough difference, while it isn't guaranteed to be diagonally dominant.

Algorithm 11 Random prototype community interaction matrix generation

Data: Number of communities K . Standard deviation σ . Average absolute gap ϵ .

Result: Random symmetric prototype community matrix B_0 . Its rows are on average at least ϵ apart from each other in absolute difference.

1. Set $B_0 = \begin{pmatrix} 0 \end{pmatrix}_{K \times K}$ as the initial value of the output.
2. Set B_{01} , by sampling from independent normal distributions $N(1, \sigma^2)$.
3. For $i \in \{2, \dots, K\}$:
4. Generate random vector $\mathbf{tmp}_j \sim N(1, \sigma^2)$ of length $K - i + 1$ as the tentative value for row i of B_0 .
5. If all values of \mathbf{tmp} are within $(0, 2)$, and

$$\frac{1}{K - i + 1} \sum_{j=i}^K |\mathbf{tmp}_j - (\mathbf{B}_0)_{l,j}| > \epsilon, \quad l \in \{1, \dots, i - 1\}$$

then we accept \mathbf{tmp} as the final value of the i th row of B_0 , otherwise, we repeat (a) and generate the random vector again

6. Copy over the lower triangle part of B_0 over to the upper triangle part so that we get a symmetric B_0 . Return B_0 .

Similar to above settings, we generate network data sets with 600 nodes and equally-sized communities with prototype community interaction matrices generated with Algorithm 11. We summarize the results after running our algorithm in the following tables.

$\rho \backslash K$	2	3	4	5
0.05	(0.935,0.065,0)	(0.995,0.005,0)	(1,0,0)	(1,0,0)
0.1	(0.73,0.27,0)	(0.94,0.055,0.005)	(1,0,0)	(1,0,0)
0.15	(0.58,0.42,0)	(0.82,0.17,0.01)	(1,0,0)	(1,0,0)
0.2	(0.46,0.535,0.005)	(0.67,0.325,0.005)	(1,0,0)	(1,0,0)

Table 3.38: Under-fitting, correct, and over-fitting model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.4$.

$\rho \backslash K$	2	3	4	5
0.05	(0.875,0.125,0)	(0.985,0.015,0)	(1,0,0)	(1,0,0)
0.1	(0.635,0.365,0)	(0.88,0.12,0)	(0.98,0.02,0)	(0.99,0.01,0)
0.15	(0.49,0.51,0)	(0.755,0.24,0.005)	(0.935,0.06,0.005)	(0.97,0.025,0.005)
0.2	(0.425,0.57,0.005)	(0.62,0.37,0.01)	(0.835,0.165,0)	(0.89,0.105,0.005)

Table 3.39: Under-fitting, correct, and over-fitting model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.6$.

$\rho \backslash K$	2	3	4	5
0.05	(0.835,0.165,0)	(0.995,0.005,0)	(1,0,0)	(1,0,0)
0.1	(0.65,0.35,0)	(0.9,0.1,0)	(0.96,0.035,0.005)	(1,0,0)
0.15	(0.505,0.49,0.005)	(0.695,0.3,0.005)	(0.92,0.08,0)	(0.97,0.03,0)
0.2	(0.31,0.685,0.005)	(0.61,0.39,0)	(0.84,0.155,0.005)	(0.87,0.13,0)

Table 3.40: Under-fitting, correct, and over-fitting model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.8$.

$\rho \backslash K$	2	3	4	5
0.05				
0.1				
0.15				
0.2				

Table 3.41: Under-fitting (red), correct (green), and over-fitting (blue) model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.4$.

$\rho \backslash K$	2	3	4	5
0.05				
0.1				
0.15				
0.2				

Table 3.42: Under-fitting (red), correct (green), and over-fitting (blue) model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.6$.

$\rho \backslash K$	2	3	4	5
0.05				
0.1				
0.15				
0.2				

Table 3.43: Under-fitting (red), correct (green), and over-fitting (blue) model selection percentages for running Algorithm 7 on network data sets generated on random and non-diagonal-dominant community interaction matrices with absolute average gap $\epsilon = 0.8$.

The general observation we see from simulation studies in this section as well as in Section 3.1.3 is that our algorithm tends to perform better on networks with higher network density and fewer true communities. This result is quite intuitive. We estimate the community labels by effectively doing clustering on a rotated version of the adjacency matrix's row vectors. The higher the network density is, the more non-zero elements the row vectors would have, and the easier nodes from different true communities can be distinguished. Also, the more true communities the network contains, the more precise the estimation algorithm needs to be in distinguishing between different vectors, making it harder (and requiring more edges / higher density from the network) to get precise estimates.

3.1.7 Changing up the train-test splitting: comparison with other CV methods

We run the same simulations with equally sized community networks as in Subsection 3.1.3 here. We change the train-test splitting method according to Section 2.2, both with and without the matrix completion after blocking out the test set each time.

Again, we generate network data sets with 600 nodes with different (2, 3, 4, 5) equally sized communities and different density parameters and apply different model selection algorithms on them. For each setting, we generate 200 instances of such data sets and record the percentage of under-fitting, correct, and over-fitting model selection results. We summarize these results numerically in Table 3.44 and also visually in Table 3.45. We can see that in general, the percentage of correct model selection increases as we increase the density parameter, while decreases as we increase true number of communities. This is pretty intuitive, since higher density indicates more edges in the realization and thus easier community detection via clustering. Also, when we have more communities, it might be harder to exactly detect all the communities, especially when the density parameter is low, which is why we mostly see under-fitting models being selected for $K^* = 5$.

Another observation we have is that random edge sampling outperforms LatinCV by quite a lot. This is especially obvious when we are dealing with $K^* = 4, 5$, where random edge sampling has much higher correct model selection percentage, while the LatinCV variant would make a lot of mistakes. Also, it seems that the matrix completion step doesn't improve the model selection result. When we have $\rho = 0.05$ and $K^* = 5$, the variant with matrix completion actually has a significant portion of under-fitting model selection, while the variant without the step is making most of selections correctly.

ρ	Algorithm	2	3	4	5
0.025	Block-wise splitting	(0,1,0)	(0.005,0.995,0)	(0.725,0.275,0)	(1,0,0)
	Random CV w.o. completion	(0,1,0)	(0.005,0.995,0)	(1,0,0)	(1,0,0)
	Random CV w. completion	(0,0.985,0.015)	(0.005,0.98,0.015)	(1,0,0)	(1,0,0)
	Latin CV w.o. completion	(0,1,0)	(0.01,0.99,0)	(1,0,0)	(1,0,0)
	Latin CV w. completion	(0,1,0)	(0.12,0.875,0.005)	(1,0,0)	(1,0,0)
0.05	Block-wise splitting	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	Random CV w.o. completion	(0,1,0)	(0,1,0)	(0,1,0)	(0.02,0.98,0)
	Random CV w. completion	(0,0.99,0.01)	(0,0.995,0.005)	(0,0.995,0.005)	(0.235,0.765,0)
	Latin CV w.o. completion	(0,1,0)	(0,1,0)	(0.055,0.655,0.29)	(1,0,0)
	Latin CV w. completion	(0,0.995,0.005)	(0,1,0)	(0.12,0.54,0.34)	(0.995,0.005,0)
0.1	Block-wise splitting	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	Random CV w.o. completion	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	Random CV w. completion	(0,1,0)	(0,0.995,0.005)	(0,0.99,0.01)	(0,0.98,0.02)
	Latin CV w.o. completion	(0,1,0)	(0,1,0)	(0,0.765,0.235)	(0.72,0.085,0.195)
	Latin CV w. completion	(0,0.995,0.005)	(0,0.99,0.01)	(0,0.725,0.275)	(0.78,0.045,0.175)
0.15	Block-wise splitting	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	Random CV w.o. completion	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	Random CV w. completion	(0,1,0)	(0,1,0)	(0,0.995,0.005)	(0,0.97,0.03)
	Latin CV w.o. completion	(0,1,0)	(0,1,0)	(0,0.87,0.13)	(0.655,0.085,0.26)
	Latin CV w. completion	(0,1,0)	(0,1,0)	(0,0.885,0.115)	(0.63,0.06,0.31)
0.2	Block-wise splitting	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	Random CV w.o. completion	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	Random CV w. completion	(0,1,0)	(0,1,0)	(0,1,0)	(0,1,0)
	Latin CV w.o. completion	(0,1,0)	(0,1,0)	(0,0.92,0.08)	(0.605,0.015,0.38)
	Latin CV w. completion	(0,1,0)	(0,1,0)	(0,0.945,0.055)	(0.605,0.045,0.35)

Table 3.44: Percentage of under-fitting, correct and over-fitting model selection results for CVC methods with random edge sampling ('random CV' in the table) and LatinCV, under different network density parameter and true community numbers.









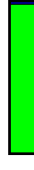
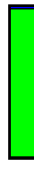














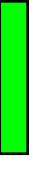
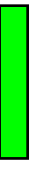
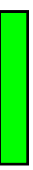
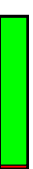
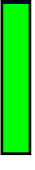
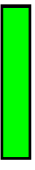
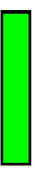
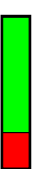
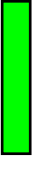
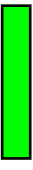
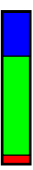

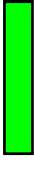
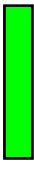
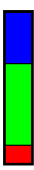
























































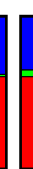




ρ	Algorithm	2	3	4	5
0.025	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				
0.05	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				
0.1	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				
0.15	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				
0.2	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				

Table 3.45: Under-fitting (light red on left of green), correct (green), and over-fitting (blue on right of green) percentage of model selection results for CVC methods with random edge sampling ('random CV' in the table) and LatinCV, under different network density parameter and true community numbers.

3.1.8 Swapping out Gaussian Multiplier Bootstrap: comparison with parametric bootstrap

We also test the impact of using parametric bootstrap instead of using Gaussian multiplier bootstrap. As mentioned in Section 2.6, we now obtain the sampling distribution of the test statistics in the hypothesis testing step by generating partial network adjacency matrices corresponding to the testing set according to the null hypothesis, i.e. the current K value is the true number of communities.

We check the effectiveness of parametric bootstrap by running Algorithm 10 on networks with equally sized communities, just as in Section 3.1.3. In other words, we generate networks with $K = 2, 3, 4, 5$ equally sized communities and different density parameters. Repeat such generations 200 times each and see what percentage of times the algorithm can find the correct number of communities K^* . Table 3.46 summarizes the results in this regard. Here we see quite similar properties as the version with Gaussian multiplier bootstrap. The results become better when we move from low-density networks to high-density networks. When we are working with low-density networks, the algorithm finds it hard to correctly choose the higher K values (such as 4 or 5). Some unique aspects of these variants also exist. For example, when we look closely at the ternary arrays, we notice that sometimes they don't add up to 1. This is because in some cases, the hypothesis testing step is rejecting all candidate K values, instead of accepting at least one of them. This is likely because the bootstrap samples of the test statistic we obtained do not exactly follow the sampling distribution of the test statistic. Although we are using the community interaction matrix estimators using only the community label and edges from the test set, it is still likely that the distribution doesn't exactly follow that given the null hypothesis. We also see that the Latin CV variants don't work very well compared to block-wise splitting and random edge sampling. Also, the matrix completion step doesn't influence the performance much.

ρ	Algorithm	2	3	4	5
0.025	Block-wise splitting	(0,0.835,0)	(0,0.64,0.015)	(0.37,0.42,0.09)	(1,0,0)
	Random CV w.o. completion	(0,0.955,0)	(0,0.9,0)	(1,0,0)	(1,0,0)
	Random CV w. completion	(0,0.94,0.045)	(0,0.8,0.105)	(1,0,0)	(1,0,0)
	Latin CV w.o. completion	(0,0.94,0)	(0.015,0.88,0)	(1,0,0)	(1,0,0)
	Latin CV w. completion	(0,1,0)	(0.105,0.86,0.035)	(1,0,0)	(1,0,0)
0.05	Block-wise splitting	(0,0.945,0)	(0,0.835,0)	(0,0.795,0)	(0,0.445,0)
	Random CV w.o. completion	(0,0.97,0)	(0,0.92,0)	(0,0.915,0)	(0.01,0.885,0)
	Random CV w. completion	(0,0.96,0.035)	(0,0.865,0.025)	(0,0.83,0.02)	(0.205,0.62,0.09)
	Latin CV w.o. completion	(0,0.975,0)	(0,0.925,0)	(0.04,0.435,0.365)	(1,0,0)
	Latin CV w. completion	(0,1,0)	(0,0.99,0.01)	(0.055,0.555,0.39)	(1,0,0)
0.1	Block-wise splitting	(0,0.915,0)	(0,0.85,0.055)	(0,0.79,0.08)	(0,0.7,0.045)
	Random CV w.o. completion	(0,0.965,0)	(0,0.92,0)	(0,0.915,0)	(0,0.85,0)
	Random CV w. completion	(0,0.95,0.05)	(0,0.915,0.03)	(0,0.815,0.025)	(0,0.615,0.025)
	Latin CV w.o. completion	(0,0.98,0)	(0,0.965,0.01)	(0,0.565,0.13)	(0.915,0.01,0.065)
	Latin CV w. completion	(0,0.97,0.03)	(0,0.99,0.01)	(0,0.705,0.26)	(0.875,0.015,0.11)
0.15	Block-wise splitting	(0,0.92,0.005)	(0,0.79,0.14)	(0,0.845,0.125)	(0,0.82,0.11)
	Random CV w.o. completion	(0,0.97,0)	(0,0.94,0.015)	(0,0.91,0.015)	(0,0.86,0.03)
	Random CV w. completion	(0,0.94,0.05)	(0,0.915,0.07)	(0,0.85,0.055)	(0,0.805,0.055)
	Latin CV w.o. completion	(0,0.93,0.02)	(0,0.965,0.025)	(0,0.605,0.08)	(0.775,0.035,0.17)
	Latin CV w. completion	(0,0.99,0.01)	(0,1,0)	(0,0.89,0.105)	(0.785,0.04,0.17)
0.2	Block-wise splitting	(0,0.89,0.015)	(0,0.835,0.15)	(0,0.82,0.18)	(0,0.815,0.175)
	Random CV w.o. completion	(0,0.94,0)	(0,0.935,0.02)	(0,0.955,0.02)	(0,0.815,0.11)
	Random CV w. completion	(0,0.92,0.065)	(0,0.9,0.08)	(0,0.91,0.05)	(0,0.89,0.075)
	Latin CV w.o. completion	(0,0.88,0.04)	(0,0.93,0.05)	(0,0.765,0.08)	(0.72,0.025,0.22)
	Latin CV w. completion	(0,1,0)	(0,0.995,0.005)	(0,0.91,0.085)	(0.725,0.02,0.24)

Table 3.46: Percentage of under-fitting, correct and over-fitting model selection results for CVC methods with parametric bootstrap, under different network density parameter and true community numbers. Data sets in this table are generated using the standard Stochastic Block Model.

ρ	Algorithm	2	3	4	5
0.025	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				
0.05	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				
0.1	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				
0.15	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				
0.2	Block-wise splitting				
	Random CV w.o. completion				
	Random CV w. completion				
	Latin CV w.o. completion				
	Latin CV w. completion				

Table 3.47: Percentage of under-fitting(light red on the left of green), correct (green), over-fitting (blue on right of green), and none (gray) model selection results for CVC methods with parametric bootstrap. Data sets generated using the standard Stochastic Block Model.

3.1.9 Note on Computation Speed

In our above discussions, we tested the properties of our method, as well its several variants. As we have discussed, there are several steps where we need to make a choice between several options.

As the discussion in Section 2.2 (and in particular the comparison in Figure 2.3) shows, block-wise node-pair splitting would always lead to splitting that leaves the entire rows and columns of the adjacency matrix in one fold only. One important advantage this feature bring is that we always get to operate on partial adjacency matrices and thus have access to all the optimizations for matrix operations that come with any modern statistical software. However, when we are working with random edge sampling or LatinCV edge sampling, each fold would come as very random subsets of the adjacency matrix. The operations become much harder to vectorize and thus will take longer time.

For example, after obtaining the model estimation for a given K , we would need to get edge probability predictions over the test set. And for convenience, let's assume now we are dealing with the first fold out of V folds, so the test set contains node 1 through node $\frac{n}{V}$. Then given the model estimates $\hat{g}^{(tr,K)}$, $\hat{\mathbf{B}}^{(tr,K)}$, we simply need to turn the community label vector for the test set

$$\hat{\mathbf{g}}_i^{(tr,K)} \quad 1 \leq i \leq \frac{n}{V}$$

into a matrix $\mathbf{G}_{\frac{n}{V} \times K}$, such that

$$\mathbf{G}_{i,k} = 1 \text{ iff. } \hat{\mathbf{g}}_i = k$$

And we can obtain the predictions by the following:

$$\hat{P}^{(test,K)} = \mathbf{G} \cdot \hat{\mathbf{B}}^{(tr,K)} \cdot \mathbf{G}^T$$

When we are dealing with random edge sampling or LatinCV, however, the test set with each fold will be a very random set of node pairs which we need to keep track of individually. In table 3.48 we show an example of such a test set. For the same edge probability prediction task, we would need to loop through each row of this list and get its prediction by calculating

$$\hat{\mathbf{B}}_{\hat{g}_i^{(tr,K)}, \hat{g}_j^{(tr,K)}}^{(tr,K)}$$

and we would lose all the speed up from vectorized operations.

	Row index	Column index
1	1	2
2	2	3
3	1	5
4	3	5
5	2	6

Table 3.48: Example of a test set using LatinCV. Here we are splitting all node pair into 3 fold in a network of 6 nodes.

When we use the Gaussian multiplier bootstrap for obtaining sampling distribution of our test statistic, we only need the cross-validated loss matrix, which can be obtained by looping through each candidate K value and each fold exactly once, in order to obtain the predictions for each node pair and get the validated loss by calculating the loss function value using the prediction and its corresponding test set value. While for the parametric bootstrap, we will need to loop through them multiple times, obtaining the bootstrap samples and calculate the test statistic for each sample. We thus have to pre-train and store all the model estimates and test set edge probability predictions for each fold and candidate K combination, which would sacrifice some memory but save the time of repeated model estimation. When implemented completely using native R code, this procedure is painfully slow. Each run would take roughly 180 seconds, making it almost impossible to test its consistency properties comprehensively. We had to instead profile the algorithm and turn its slower components into C++ functions with the help of Rcpp package ([EFA⁺11]) as well as utilize parallel computation in the bootstrap sample generation step (assigning independent generations to different cores). The time for each run is thus lowered to a relatively tolerable level (~ 40 seconds).

The low-rank matrix completion step can also be a drag on the computational speed. When we are dealing with larger networks, calculating the singular value decomposition and giving the low-rank approximation by adding up the components corresponding to the leading singular values could take a while. In our experiments, we find for a network of 600 nodes, adding the matrix completion step roughly adds 5 seconds to the running time of the algorithm.

In Table 3.49, we summarize the average runtime of all variants of these algorithms. Most of the experiments are on two kinds of machines: a 32-core computer with 64G memory and each core with 2.6GHz speed, a 4-core computer with 16G memory and each core with 3.6GHz speed (special thanks to the department of mathematics for granting us access to these machines). Some subroutines of our algorithm are well-optimized R functions that would utilize parallelism when it is possible, yet we still notice for some variants of our algorithm, the running time is better on the 4-core machine, likely because the cost of transmitting data back and forth overweighs the speedup from parallel computation, especially since gaussian multiplier bootstrap can be largely vectorized and thus perform almost as well on a single core.

Algorithm	Time (seconds)	Remark
block-wise node pair splitting & multiplier bootstrap	3.63 (2.46)	Parallel computing doesn't really help with speed. Data transmission cost overweighs the gain from parallel computing.
random edge sampling w.o. matrix completion & multiplier bootstrap	11.13 (7.51)	
random edge sampling w. matrix completion & multiplier bootstrap	16.34 (11.25)	
LatinCV w.o. matrix completion & multiplier bootstrap	11.13 (8.26)	
LatinCV w. matrix completion & multiplier bootstrap	16.38 (10.64)	Speed-up possible through parallel computing. Bottleneck is in removing identical columns in the cross-validated loss matrix.
block-wise node pair splitting & parametric bootstrap	29.87	
random edge sampling w.o. matrix completion & parametric bootstrap	37.98	
random edge sampling w. matrix completion & parametric bootstrap	44.38	
LatinCV w.o. matrix completion & parametric bootstrap	37.58	
LatinCV w. matrix completion & parametric bootstrap	44.20	

Table 3.49: Average running time on a network with 600 nodes with 3 equally sized communities and bootstrap sample size $B = 200$. For all the tasks we are using 32-core computer with 64G memory, each core with 2.6GHz speed. Note that for the first five tasks, parallelism doesn't really help with running time, so a 4-core computer each with 3.6GHz speed (numbers in parentheses) would actually be faster for the same tasks. Each average running time estimate comes from average of 100 runs. Note that for the parametric bootstrap tasks, we heavily optimized some slower R functions by re-writing them in C++ with the help of Rcpp package and parallelism. So this is not really a fair comparison. In fact, block-wise node pair splitting with Gaussian multiplier bootstrap is much faster if we implement all of algorithms verbatim.

3.2 Application on real-world data sets

In this section, we apply our method on some real-world data sets to test its effectiveness. The data sets we will be using are mostly network data sets that are widely explored in the network community, particularly for the purpose of community detection.

3.2.1 Political Blog Data Set

The political blog data set contains a set of 1494 American political blogs and their connection in terms of hyperlinks. This data set was first collected and analyzed in [AG05] before the 2004 US Election. It has since been widely used in the network community, especially for community detection purposes. [KN11], [ZLZ11], [Jin15], [Lei16], [Lei18] all used it as an example of network with clear community structures as well as heterogeneous degree distribution within communities. We have a subset of 1222 political blogs with manual labels of the leaning of each blog as either liberal or conservative. In Figure 3.4 we show variances explained by the top 15 principal components as well as the projection onto the first two principal components of the adjacency matrix among these blogs. We can see clearly from the figure that the rank-2 approximation can explain a decent amount of variance in the data set. Also, there are two major communities in this network and the division follows the blogs' party affiliations very closely.

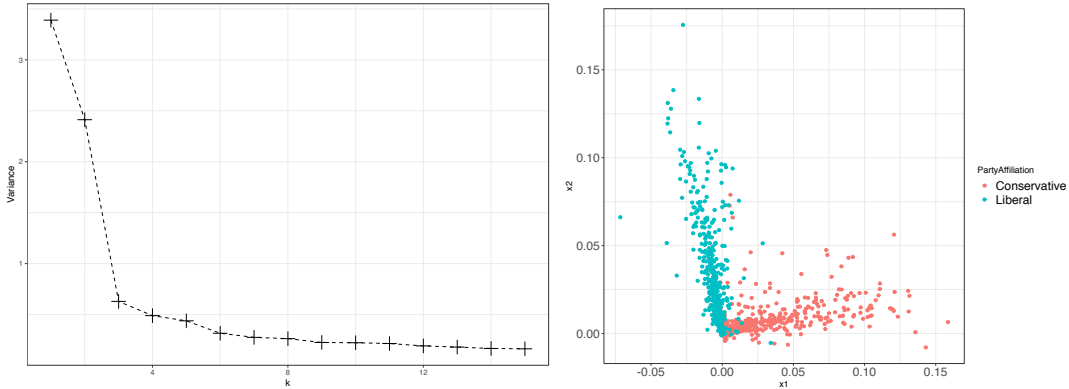


Figure 3.4: Left plot shows the variances explained by each principal component of the political blogs adjacency matrix. Right plot shows the projections of all political blogs onto first two principal components of their adjacency matrix.

Here we apply the cross-validation with confidence method on this data set. We will try both the version for standard stochastic block models as well as the version for degree-corrected stochastic block models. It turns out that when we use the standard SBM version of algorithm 8, it tends to select an over-fitting model (usually the largest K value in the candidate set). While the DCBM version of the algorithm would consistently select $K = 2$ correctly. This is likely due to the fact that the political blogs are very heterogeneous in terms of degree distribution. When using the standard stochastic block model, we cannot account for the individual difference in number of connections each blog makes and therefore end up with an over-fitting model. The additional flexibility DCBM affords us helps with the problem and leads us to the correct model.

In Table 3.50, we show the confusion matrix between estimated communities using the adjacency matrix and the manual labeling of each blog. We see that the estimated communities are very close to the ground truth.

	1	2
1	534	14
2	52	622

Table 3.50: Confusion matrix between manual labels (columns) and estimated community labels assuming a degree-corrected stochastic model with $K = 2$.

We ran the algorithms 200 times to see its consistency under randomness of the random fold assignment. We show the P-values for these repetitions in Figure 3.5 and 3.6. It is clear that degree-corrected model is the more appropriate assumption to make. When we use the algorithm for standard stochastic block models, the K value selected is clearly too large and not stable, while when we use the method for degree-corrected stochastic block models, we consistently select 2 as the correct K value. The results for 3-fold CVC is also shown in Table 3.51.

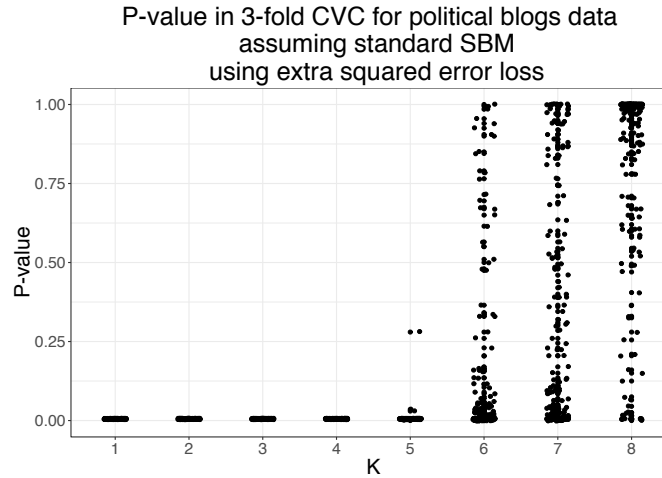


Figure 3.5: P-values for different candidate K values using 3-fold CVC assuming standard stochastic block model.

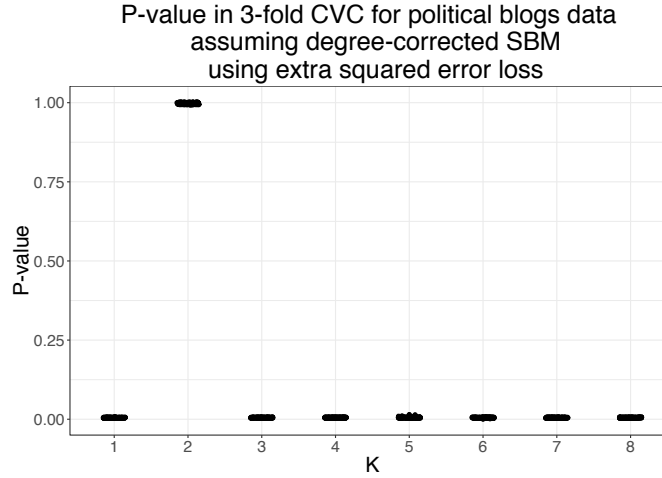


Figure 3.6: P-values for different candidate K values using 3-fold CVC assuming degree-corrected stochastic block model.

Selected K for SBM				Selected K for DC-SBM	
K	5	6	7	K	2
Frequency	55	67	78	Frequency	200

Table 3.51: Frequency table for selected K values using 3-fold CVC on the political book data set, with extra squared error loss function.

In order to check the robustness under different fold numbers and loss functions, we also ran the algorithm for degree-corrected stochastic block models on the political blogs data set using 5-fold, 10-fold CVC as well as using the negative likelihood loss function. The results are still very desirable. We show the results for the 5-fold and 10-fold CVC with extra squared error loss in Figure 3.7. We see that the algorithm is still consistently choosing $K = 2$ as the correct model, although some over-fitting model (such as $K = 4, 5, 6$) might also end up in the confidence set, although they won't be selected since $K = 2$ is still the most parsimonious model in the confidence set.

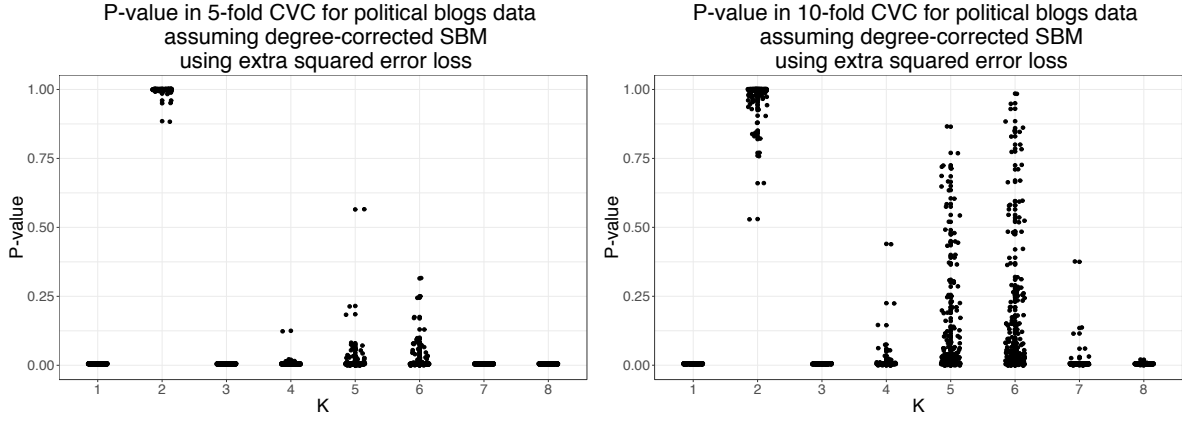


Figure 3.7: P-values for different candidate K values using 5-fold and 10-fold CVC assuming degree-corrected stochastic block model. Here we are using the extra squared error loss function.

We also ran the algorithm using negative likelihood loss function. The similar P-value results are shown in Figure 3.8. The performance are roughly the same. The algorithm will consistently choose $K = 2$ as the correct output, while some over-fitting models such as $K = 4, 5, 6$ would remain in the confidence set when we use 5-fold and 10-fold CVC. The CVC method for degree-corrected stochastic block models performs very well on the political blogs data set. It very consistently choose the correct model under random fold assignment and is robust under changing fold numbers and loss functions.

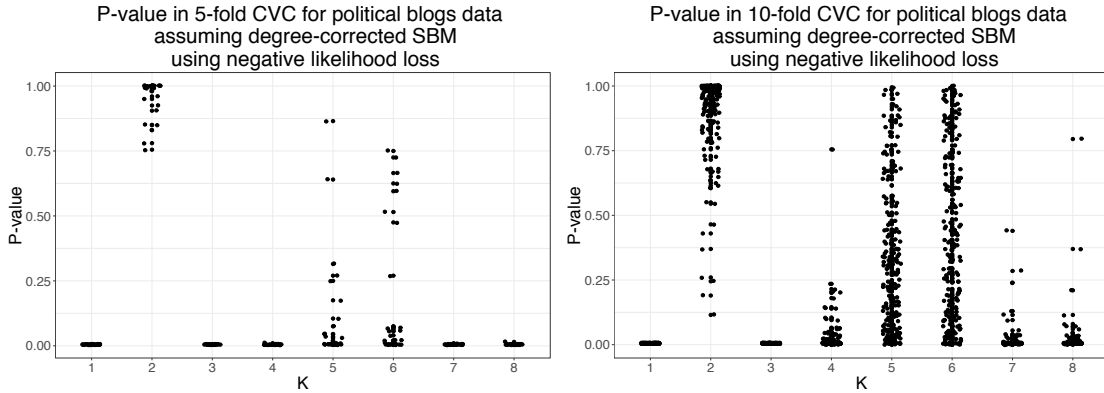


Figure 3.8: P-values for different candidate K values using 5-fold and 10-fold CVC assuming degree-corrected stochastic block model. Here we are using the negative likelihood loss function.

When we use Algorithm 9, almost 100% of the time, only the degree-corrected stochastic block model with $K = 2$ is retained in the confidence set. In other words, only the degree-corrected stochastic block model is not rejected by the model selection process. The results are so robust that there isn't much variation in 200 runs of the algorithm. We therefore won't show

any visualization of the results here. This is likely due to the fact that the political blog data set has lots of nodes (1222 total blogs), and thus the randomness in train-test split won't affect the model selection outcome as much.

3.2.2 Political Books Data Set

The political books data set is a set of 105 political books and the undirected link relations between them according to co-purchase records on Amazon. We also have the manual labeling of these books according to their party-affiliation: liberal, conservative, and independent. We visualize the principal component analysis result for this data set as well and show the results in Figure 3.9. We can see again that the rank-2 approximation would be a decent picture of the variations in the data set. The projections onto the first two principal components show two clear communities along the party line, although the independent books are mixed with the two major communities and likely will be hard to distinguish. This might be because people tend to not buy books written by authors on the other aisle, while they are fine buying books with independent viewpoints.

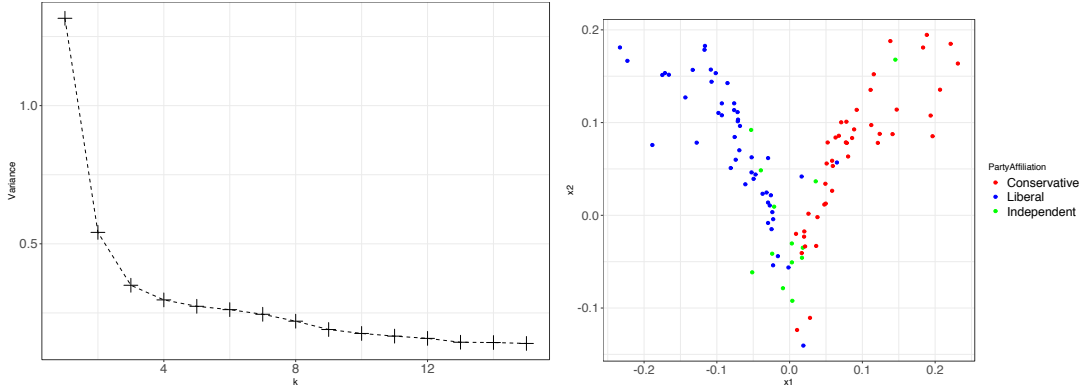


Figure 3.9: Left plot shows the variances explained by each principal component of the political blogs adjacency matrix. Right plot shows the projections of all political blogs onto first two principal components of their adjacency matrix.

When we apply the CVC method on this data set with candidate set $\mathbb{K} = \{1, 2, 3, 4, 5\}$, we very consistently get $K = 2$ as the selected number of communities. This is true whether we use the SBM or the DC-SBM version of the algorithm. This is consistent with what we see from the previous visualizations. Although the independent books are usually classified separately, it cannot quite distinguish itself and form its own community. Table 3.52 shows the comparison between the estimated community labels assuming degree-corrected stochastic block models and the manual labels which come with the data set. It is clear that each estimated community mostly corresponds to books with one party affiliation, with the independent books evenly split between the two communities.

	1	2	3
1	7	43	3
2	6	0	46

Table 3.52: Confusion matrix between book labels (columns) and estimated communities (rows) assuming DC-SBM with $K = 2$.

Selected models	Frequency
SBM: 3, DCBM: 2	78
SBM: 3, DCBM: 3	43
SBM: 4, DCBM: 2	15
SBM: 4, DCBM: 3	15
SBM: 2, DCBM: 2	12
SBM: 5, DCBM: 2	8
SBM: 3, DCBM: 4	7
DCBM: 3	5
SBM: 5, DCBM: 3	5

Table 3.53: Most frequent model selection results selecting in both standard stochastic block models and degree-corrected stochastic block models, using spectral weighting of the singular vector matrix. We select the most parsimonious model within the retained K 's in each category. We omitted the results that appear fewer than 5 times out of 200 runs.

When we use the Algorithm 9, i.e. the CVC method which can select among both standard stochastic block models and degree-corrected stochastic block models, we get some quite interesting results. Now there are two kinds of models in the confidence set, so there is not a clear-cut most parsimonious model any more. We can only pick the smallest candidate K accepted within each category. Through 200 runs, we summarize the results appearing the most frequently in Table 3.53 (omitting all the results appearing less than 5 times). In this case, we see that the degree-corrected stochastic block model with $K = 2$, which is widely accepted as the most appropriate model for this data set, is retained in the confidence set at least half of the time. There is clearly a lot of variability in the model selection result in this case. We think this is likely due to the fact that there are only 105 books in the data set and therefore the randomness in fold assignment would impact the model estimation and selection results significantly.

Chapter 4

Theoretical Results

As mentioned in chapter 2, the CVC procedure would take in a set of candidate models and conduct hypothesis testing between each pair of them. A model would be eliminated as soon as it fares significantly worse compared to one other model. And the output would be the set of all models not eliminated in the procedure. Then out of all these models, we would select the most stringent one as the final output.

In order to prove that this output would contain the optimal (true) model with guaranteed probability, we would need to show that **(i)** an under-fitting model is guaranteed to be rejected by the hypothesis test defined in chapter 2 and **(ii)** the true model won't be rejected when compared with an over-fitting model. Therefore, the output set should rarely contain an under-fitting model, and would only contain optimal and some over-fitting models. And the most stringent model in this set would very likely be the true model.

Compared to the original CVC procedure under linear regression settings as mentioned in 2.4, the technical challenges in this project are as follows:

1. Unlike the relatively straightforward estimation for simple linear models, estimation of stochastic block models is usually a two-step process, where one first determines the community labels and then estimates the model parameters. As introduced in chapter 1, there are many approaches to determine the community labels, and the properties of these methods differ a lot from each other. The estimation with correct K value is well understood and thoroughly studies in the literature. We want to make study as general as possible and not to impose unrealistic assumptions on the estimation with incorrect K value. Therefore we proceed assuming the following:
 - 1.1. When $K = K^*$, and the network density $\rho_n = \Omega(\frac{\log n}{n})$, where $\Omega(\cdot)$ is the asymptotic lower bound notation as defined in Table 1.1, we assume that the community recovery is exact. This is a well-studied result, for example in [CWA12], [CX14], [AS15]. When this assumption doesn't hold, our theoretical study would be a little more difficult. Given the community labels assigned by the community recovery algorithm, the edges within the same identified community are no longer independent. But intuitively, it should still be possible to overcome this challenge if we make reasonable assumption about the precision of the recovery algorithm. For example, we can assume that the symmetric difference between an identified community and

the true community cannot exceed a certain portion of the true community, leading to an upper bound in the error of the edge probability estimate in the shape of

$$\left| \hat{B}_{k,k'} - B_{k,k'}^* \right| \leq c\rho_n \quad (4.1)$$

for some $c > 0$.

- 1.2. When $K \neq K^*$, we make an assumption about the size of each identified communities by the incorrect model. More specifically, we assume that each identified community in the test set at least has size on the order of $\sqrt{\frac{n}{\log n}}$. Noticing that this is an lower-order term compared to n , this is not a very strong assumption. For under-fitting models, we know that at least two communities will have significant mixture since there isn't enough degrees of freedom in the model, and our proof will be mostly based on that. While in over-fitting models, we don't know much about their behavior, and such size assumptions will help us a lot in characterizing the behavior of such models and make the proof more convenient.
2. Under the regression setting as addressed in Section 2.4, each validated loss term is independent and identically distributed in a single fold, while in stochastic block models, nodes are correlated with each other through their community labels. More specifically, the test set validated losses are no longer i.i.d. They are conditionally independent given the correct community labels of the nodes. We will need to keep this in mind and be more cautious while using concentration inequalities to depict the behavior of these validated losses.
3. When we prove that under-fitting models are guaranteed to be eliminated, it is easy to see intuitively that the under-fitting model would have extra loss due to the fact that at least two communities will have significant mixture in the same identified community since the model doesn't have large enough K , just as shown in [CL17]. However, since the test statistic is normalized, as in Equation 2.6, it isn't straightforward that this would necessarily lead to the under-fitting model being rejected in the hypothesis testing step. We would need to characterize the normalizing term with care in order to show it rigorously.
4. When we show over-fitting models would not significantly outperform true models under our hypothesis testing framework, we would need to upper bound the extra loss for the true model as well as over-fitting models. This will require completely different technique compared to the methods we used for proving the under-fitting case. As we would expect, the over-fitting model might have very similar performances compared to the true model, especially when it is a refinement of the true model (i.e. most true communities remain unchanged, only one or several got split into smaller ones). Then when we recall Equation 2.6, the numerator and denominator might both be very small. We need a community size assumption to exclude the case that some extremely small identified communities might lead to extreme edge probability estimates.

Here we prove the consistency of selecting number of communities for stochastic block models using cross validation with confidence while using the rectangular set for model estimation and assuming exact consistent recovery of the community labels. More specifically,

we have the following main theorem under certain assumptions for Algorithm 8 we defined in Chapter 2.

The assumptions we will be using will be the following:

- A.1 The smallest true community has size at least $\pi'_0 n$.
- A.2 The smallest identified community in the test set has size on the order $\sqrt{\frac{1}{\rho_n}}$.
- A.3 When we use the correct K value: $K = K^*$, we can achieve exact recovery of communities.
- A.4 $B = \rho_n B_0$, where B_0 is a $K \times K$ symmetric matrix with entries in $(0, 1]$, and the rows of B_0 are all distinct. The rate ρ_n , which controls the network sparsity, satisfies $\rho_n = \Omega(\frac{\log n}{n})$.
- A.5 The P-value threshold in Algorithm 8 is selected to be a sequence α_n such that $\alpha_n \in (\frac{1}{n}, 1)$ and $\alpha_n \rightarrow 0$.
- A.6 Number of candidate models $|\mathbb{K}|$ is finite.

Remark: our assumptions still have potential to be relaxed. For example, the lower bound for size of communities in A.1 can be even smaller. The size of candidate model set $|\mathbb{K}|$ can change slowly with respect to n in A.6 without influencing most of the results we will be showing.

Our theoretical results can be summarized as the following theorem:

Theorem 4.1

Under assumptions A.1 - A.6, with the squared error loss as defined in Equation 2.3. We have the following:

1. For $\tilde{K} < K^*$, we have

$$P(T_{\tilde{K}, K^*} > Z_{\alpha_n}) \rightarrow 1$$

where $T_{\tilde{K}, K^*}$ is the test statistic in our hypothesis testing as defined in Equation 2.6, Z_{α_n} is the upper α_n quantile of a standard normal distribution.

In other words, an under-fitting model is guaranteed to be eliminated in Algorithm 8.

2. For $\tilde{K} > K^*$, we have

$$\sup_{K > K^*} T_{K^*, \tilde{K}} = O_P(1)$$

Since $Z_{\alpha_n} \rightarrow \infty$ as $n \rightarrow \infty$ and $\alpha_n \rightarrow 0$, we know that $\forall K > K^*$,

$$P(T_{K^*, \tilde{K}} < Z_{\alpha_n}) \rightarrow 1$$

In other words, the true model won't be eliminated by an over-fitting model and will be included in the confidence set \mathbb{A}_{cv} .

In the following two sections and the corresponding sections in the appendix we will be stating and proving the two parts in this main theorem.

4.1 Under-fitting case

In [CL17], it is shown that the under fitting model will always be rejected with high probability since it will surely have higher squared error loss using regular cross validation. This is because when the nodes are assigned into fewer communities than there truly are, the pigeon hole principle guarantees that there will be significant portions of at least two true communities assigned to the same identified community. And that misalignment would lead to significant extra loss.

In our case, we use the squared error loss for measuring the fit of a model. Let $\hat{I}_{k,k'} = \{(i, j) : (\hat{g}_i, \hat{g}_j) = (k, k'), i > j\}$ denote the pairs (i, j) where i is assigned to community k and j is assigned to community k' . The extra squared error loss on cell (i, j) between using community number K_1, K_2 is denoted as:

$$\begin{aligned}\xi_{i,j}^{(K_1, K_2)} &= (A_{ij} - \hat{B}_{ij}^{(K_1, tr)})^2 - (A_{ij} - \hat{P}_{ij}^{(K_2, tr)})^2 \\ &= (2A_{ij} - \hat{B}_{ij}^{(K_1, tr)} - \hat{B}_{ij}^{(K_2, tr)})(\hat{B}_{ij}^{(K_2, tr)} - \hat{B}_{ij}^{(K_1, tr)}) \\ &= 2A_{ij}(\hat{B}_{ij}^{(K_2, tr)} - \hat{B}_{ij}^{(K_1, tr)}) + \left(\hat{B}_{ij}^{(K_1, tr)}\right)^2 - \left(\hat{B}_{ij}^{(K_2, tr)}\right)^2\end{aligned}$$

Here, $\hat{B}_{ij}^{(K, tr)}$ denotes the the probability assigned to cell (i, j) when using community number K . In other words, if $(i, j) \in \hat{I}_{k,k'}$, then $\hat{B}_{ij}^{(K_1, tr)} = \frac{1}{|\hat{I}_{k,k'}^{(tr)}|} \sum_{(s,t) \in \hat{I}_{k,k'}^{(tr)}} A_{st}$. In the following part,

when there is no danger of confusion, we would write $\xi_{i,j}^{(K_1, K_2)}$ as $\xi^{(i,j)}$. When we do not specify the models being contrasted, it is most likely the comparison between the correct model and an over-fitting/under-fitting model. The context should make it very easy to tell.

Our main result is summarized in the following theorem.

Theorem 4.2

Under assumptions A.1, A.3 - A.5,

With the squared error loss, we have when $\tilde{K} < K$,

$$P(T_{\tilde{K},K} > Z_{\alpha_n}) \rightarrow 1$$

where $T_{\tilde{K},K}$ is the test statistic established using Gaussian multiplier bootstrap approach and test statistic is defined as

$$T_{\tilde{K}} = \sqrt{\frac{n^2}{V^2}} \cdot \frac{\sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \xi_{i,j}^{(\tilde{K},K)}}{\sqrt{\sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} (\xi_{i,j}^{(\tilde{K},K)})^2}}$$

the average extra loss normalized by the second moment of the extra loss, just as defined in Equation 2.6.

Compared to the original cross-validation with confidence method, here we are normalizing the empirical mean of ξ 's using their uncentered second moment, instead of the standard deviation. We are making this choice mostly for the convenience it brings to our later theoretical work. Though we would argue that this won't affect the effectiveness of the algorithm by much. The purpose of normalization is to bring all the ξ 's to the same scale, which uncentered second moment should work almost as well as standard deviations. Also, the comparisons in Chapter 3 also show that two versions don't differ much in terms of performance.

The proof of this theorem can be found in appendix A. The assumptions made here are also very similar to those in [CL17].

In our settings, the pigeon hole principle argument would only apply under some additional assumptions. Furthermore, the Gaussian multiplier bootstrap approach requires studentization and therefore complicates the problem. It is possible that dividing by the standard deviation would make the ratio vanish, if the standard deviation is large enough. So in order to show that this statistical test will reject under-fitting models, we would need to not only lower bound

$$\hat{\mu}_{K,\tilde{K}} = \frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \xi_{i,j}^{(\tilde{K},K)}$$

but also upper bound

$$\hat{M}_{K,\tilde{K}} = \frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} (\xi_{i,j}^{(\tilde{K},K)})^2$$

i.e. the second moment of $\xi_{i,j}^{(\tilde{K},K)}$'s over the test set (from here on we loosely use the notation \hat{M} to denote the empirical uncentered second moment of $\xi^{(i,j)}$'s).

It is possible that over some region the estimates of edge probabilities might 'blow up' and thus making the denominator larger than ideal. Therefore we split the Cartesian product of all

assigned communities into two parts. When the average of interactions is within a constant factor of the network density ρ_n , there is no need to worry about the standard deviation term blowing up and we can lower bound the empirical mean term using the pigeon hole principle argument. When the average of interactions is not bounded by a constant multiple of ρ_n , we will then show that the corresponding increase in the numerator will also be on the same order and thus the test statistic will still be lower bounded and is guaranteed to be significant asymptotically.

When we have $\tilde{K} < K$, where K is the true number of communities in the data generating process. Also, note that $Z_{\alpha_n} \approx \sqrt{2 \log n}$ when we set $\alpha_n = \frac{1}{n}$. And by assumption A.4 we have $\alpha_n \in (\frac{1}{n}, 1)$. We want to show that

$$P(T_{\tilde{K}} \geq \sqrt{2 \log n}) \rightarrow 1$$

In other words, we ultimately want to show that

$$P \left[\sqrt{\frac{n^2}{V^2}} \cdot \frac{\sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \xi_{i,j}^{(\tilde{K}, K)}}{\sqrt{\sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \left(\xi_{i,j}^{(\tilde{K}, K)} \right)^2}} \geq \sqrt{2 \log n} \right] \rightarrow 1 \quad (4.2)$$

We can further rewrite $\hat{\mu}_{\tilde{K}, K}$ and $\hat{M}_{\tilde{K}, K}$ as follows

$$\begin{aligned} \hat{\mu}_{\tilde{K}, K} &= \sum_{k, k' \in \{1, \dots, \tilde{K}\}} \frac{|\hat{I}_{k, k'}|}{\binom{n}{2}} \left[\frac{1}{|\hat{I}_{k, k'}|} \sum_{(i,j) \in \hat{I}_{k, k'}} \xi^{(i,j)} \right] \\ \hat{M}_{\tilde{K}, K} &= \sqrt{\sum_{k, k' \in \{1, \dots, \tilde{K}\}} \frac{|\hat{I}_{k, k'}|}{\binom{n}{2}} \left[\frac{1}{|\hat{I}_{k, k'}|} \sum_{(i,j) \in \hat{I}_{k, k'}} \left(\xi^{(i,j)} \right)^2 \right]} \end{aligned}$$

As we will show later, the denominator terms are controlled by a constant factor of ρ_n^2 . So in order to show Equation 4.2, we need to lower bound the numerator and show that it is greater than $\frac{(\log n)^{\frac{3}{2}}}{n^2}$ up to a constant.

For each $(k, k') \in \{1, \dots, \tilde{K}\} \times \{1, \dots, \tilde{K}\}$, let $t_{k, k'}$ be such that $\hat{B}_{k, k'}^{(\tilde{K}, tr)} = t_{k, k'} \rho_n$. Let $A_s = \{\hat{I}_{k, k'} : t_{k, k'} < s\}$. Here s can be a large enough constant. Then we can split $\hat{\mu}_{\tilde{K}, K}$ and

$\hat{M}_{\tilde{K},K}$ into

$$\begin{aligned} \frac{\hat{\mu}_{\tilde{K},K}}{\hat{M}_{\tilde{K},K}} &= \frac{\sum_{A_s} \frac{|\hat{I}_{k,k'}|}{\binom{n}{2}} \left[\frac{1}{|\hat{I}_{k,k'}|} \sum_{(i,j) \in \hat{I}_{k,k'}} \xi^{(i,j)} \right] + \sum_{A_s^c} \frac{|\hat{I}_{k,k'}|}{\binom{n}{2}} \left[\frac{1}{|\hat{I}_{k,k'}|} \sum_{(i,j) \in \hat{I}_{k,k'}} \xi^{(i,j)} \right]}{\sqrt{\sum_{A_s} \frac{|\hat{I}_{k,k'}|}{\binom{n}{2}} \left[\frac{1}{|\hat{I}_{k,k'}|} \sum_{(i,j) \in \hat{I}_{k,k'}} \left(\xi^{(i,j)} \right)^2 \right] + \sum_{A_s^c} \frac{|\hat{I}_{k,k'}|}{\binom{n}{2}} \left[\frac{1}{|\hat{I}_{k,k'}|} \sum_{(i,j) \in \hat{I}_{k,k'}} \left(\xi^{(i,j)} \right)^2 \right]}} \\ &= \frac{(1-w) \cdot (I) + w \cdot (II)}{\sqrt{(1-w) \cdot (III) + w \cdot (IV)}} \end{aligned}$$

where

$$w = \frac{\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|}{\binom{n}{2}}$$

(I) and (II) are the average of $\xi^{(i,j)}$ over A_s and A_s^c , respectively, (III) and (IV) are the average of $\left(\xi^{(i,j)} \right)^2$ over A_s and A_s^c , respectively.

We separate the proof into three cases depending on the relative sizes and speed of growth in A_s and A_s^c .

1. A_s is large enough and A_s^c grows sufficiently fast. We can lower bound (I) by the pigeon hole principle argument since A_s is large enough. We can also lower bound (II) and upper bound (IV) using Bernstein's inequality since size of A_s^c is growing at a sufficient rate such that (II) and (IV) cannot deviate too far from their expectations.
2. A_s is relatively large, and A_s^c grows very slowly. (I) can still be lower bounded, and notice that $(1-w)$ is very small and the negative side of (II) can be bounded, we can still obtain a lower bound for the test statistic since both numerator and denominator is dominated by the first term.
3. A_s isn't large enough for using the pigeon hole principle. Now the second term has comparable weight as the first term in both numerator and denominator. And since (II) is lower bounded above 0 and larger than the lower bound than the negative side of (I), we can still lower bound the test statistic.

In the above section, our proof is based on the assumption of exact recovery in the model estimation process when we use the correct K (A.3). However, this is not a very realistic assumption in many cases. It would be an interesting and challenging extension to relax the assumption and base our argument on a weaker assumption such as approximate recovery, for example. The challenge would be that the independence conditions we are relying on a lot in the proof of the above Theorem would no longer hold and we will need to find other approaches for making the same argument. Intuitively, we might still be able to achieve similar guarantees if we add assumptions about the precision of the approximate recovery algorithm, such as bounding the error of edge probability estimates to a constant factor of ρ_n , as we mentioned in Equation 4.1.

4.2 Over-fitting case

Here we use some simple scenario to motivate more general settings of over-fitting in order to depict the behavior of validated loss under CVC. For the convenience of proof, we define the following:

$$T_{K,\tilde{K}} = \sqrt{\frac{n^2}{V^2}} \cdot \frac{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \xi_{i,j}^{(K,\tilde{K})}}{\sqrt{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} (\xi_{i,j}^{(K,\tilde{K})})^2}}$$

where n is the total number of nodes in the test set and ξ is defined as in section 4.1.

Eventually we want to show that the over-fitting model will not significantly outperform the true model. Again, since $\alpha_n \in (\frac{1}{n}, 1)$ and $Z_{\frac{1}{n}} < \sqrt{2 \log n}$ for large enough n 's, we want to show the following:

$$P \left(\sqrt{\frac{n^2}{V^2}} \cdot \frac{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \xi_{i,j}^{(K,\tilde{K})}}{\sqrt{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} (\xi_{i,j}^{(K,\tilde{K})})^2}} > \sqrt{2 \log n} \right) \rightarrow 0$$

i.e.

$$P \left(\frac{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \xi_{i,j}^{(K,\tilde{K})}}{\sqrt{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} (\xi_{i,j}^{(K,\tilde{K})})^2}} > \frac{\sqrt{2 \log n}}{n} \right) \rightarrow 0$$

for $\tilde{K} > K$.

Our main result in this case is summarized in the following theorem.

Theorem 4.3

Under assumptions A.1 - A.6, With the squared error loss, we have when $\tilde{K} > K^*$,

$$T_{K^*,\tilde{K}} = O_P(1)$$

where $T_{K,\tilde{K}}$ is the test statistic defined by the empirical mean of squared error losses over the test set divided by the square root of its second moment, as in Equation 2.6.

One challenge in proving this result lies in the fact that it is hard to characterize the behavior of the over-fitting model. It is possible that it would be a refinement of the true model, i.e. it is just further splitting some communities in the true model into smaller ones, in which case it is relatively straightforward to prove the result, since the over-fitting model's behavior is very

clear. This case is very possible when we are dealing a small K . But when we are dealing with a true model with moderately large K value, it isn't a practical assumption any more.

Therefore we added an extra assumption about the minimum size of discovered community in the test set. This is helpful in later trying to bound the behavior of estimated edge probabilities by the over-fitting model. Since the density parameter $\rho_n = \Omega\left(\frac{\log n}{n}\right)$, we know that this minimum size order is roughly $\Omega\left(\sqrt{\frac{n}{\log n}}\right)$, which is a lower order term compared to n . This is therefore not a very strong assumption.

We will see the other challenge when we expand the test statistic:

$$\begin{aligned}
& T_{K^*, \tilde{K}} \\
&= \sqrt{\frac{n^2}{V^2}} \cdot \frac{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \xi_{i,j}^{(\tilde{K}, K)}}{\sqrt{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} (\xi_{i,j}^{(\tilde{K}, K)})^2}} \\
&= \frac{n}{V} \frac{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \left(2A_{ij} - \hat{B}_{\hat{g}_i^{(K)}, \hat{g}_j^{(K)}}^{(K, tr)} - \hat{B}_{\hat{g}_i^{(\tilde{K})}, \hat{g}_j^{(\tilde{K})}}^{(\tilde{K}, tr)} \right) \left(\hat{B}_{\hat{g}_i^{(\tilde{K})}, \hat{g}_j^{(\tilde{K})}}^{(\tilde{K}, tr)} - \hat{B}_{\hat{g}_i^{(K)}, \hat{g}_j^{(K)}}^{(K, tr)} \right)}{\sqrt{\frac{1}{\binom{n^{(te)}}{2}} \sum_{(i,j) \in \mathbb{N}^{(te)} \times \mathbb{N}^{(te)}} \left(2A_{ij} - \hat{B}_{\hat{g}_i^{(K)}, \hat{g}_j^{(K)}}^{(K, tr)} - \hat{B}_{\hat{g}_i^{(\tilde{K})}, \hat{g}_j^{(\tilde{K})}}^{(\tilde{K}, tr)} \right)^2 \left(\hat{B}_{\hat{g}_i^{(\tilde{K})}, \hat{g}_j^{(\tilde{K})}}^{(\tilde{K}, tr)} - \hat{B}_{\hat{g}_i^{(K)}, \hat{g}_j^{(K)}}^{(K, tr)} \right)^2}}
\end{aligned}$$

Another challenge is that the term $(\hat{B}_{\hat{g}_i^{(K)}, \hat{g}_j^{(K)}}^{(K, tr)} - \hat{B}_{\hat{g}_i^{(\tilde{K})}, \hat{g}_j^{(\tilde{K})}}^{(\tilde{K}, tr)})$ in the numerator and the denominator. It is easy to see that this term can be very small in many cases. For example, when the over-fitting model is a refinement of the true model, the over-fitting model should give the same estimates as the true model on most regions where it didn't conduct the extra splitting, while in the other regions it can still give estimates pretty close to what the true model gives. Therefore the denominator could be very small. Fortunately the same term exists on the same order in the numerator. We can try to bound the term in the first pair of parentheses and then handle the numerator and denominator simultaneously at the same time.

The strategy we are using is to slice the summation in the numerator and the denominator into regions according to the assigned community labels by both the true model and the over-fitting model. Since we don't have assumption about the over-fitting model's behavior on the training set, it is possible that the over-fitting model might give edge probability estimates that deviates a lot from the network density. Then we split the proof into two cases depending on whether the edge probability estimates of the over-fitting model.

1. If the over-fitting model's edge probability estimate $\hat{B}^{(\tilde{K}, tr)}$'s are moderate in the sense that they are always a constant factor of ρ_n . Then the ratio can be relatively easy to bound using the Cauchy-Schwartz inequality after bounding some terms in the numerator and the denominator.
2. When the over-fitting mode is giving extreme edge probability estimate $\hat{B}^{(\tilde{K}, tr)}$'s over some regions, we look into the term corresponding to the region in both the numerator

and the denominator and realize that dropping both terms from the ratio would lead to the ratio being larger than the original value. And this over-estimated ratio will fall back to the first scenario, which we can bound using Cauchy-Schwartz inequality.

In the above section, we are still relying on the assumption of exact recovery when using the correct K (A.3). Besides that we are adding some mild assumptions for the convenience of the proof, especially to characterize the over-fitting model's behavior. The proof would become considerably more difficult if we drop these assumptions, especially since we know too little about an over-fitting model in general and many extreme corner cases might ruin the argument.

Chapter 5

Conclusions and Future Work

In this thesis, we proposed an extension of cross validation with confidence to solve the problem of selecting number of communities for stochastic block models. We started by reviewing the relevant literature and finding the critical building blocks to construct our method. Network cross validation is an intuitive, generic, and widely applicable framework for model selection problems, yet its classical form isn't statistical consistent. We therefore adapted the cross validation with confidence framework make the model selection more systematic by adding a formal hypothesis testing, and only reject a model if it is underperforming after considering the randomness in the train-test splitting.

Train-test splitting is no longer a straightforward process in network context. Nodes, edges and node-pairs can all be seen as entities in the network, while there are subtleties with viewing any as the entity to be split. We explored block-wise node-pair splitting, random edge sampling, and LatinCV node pair splitting by implementing them and testing their effectiveness. It turns out block-wise node-pair splitting is conceptually easier to understand and also more straightforward to implement. Also, its implementation can naturally take advantage of many optimized default behavior of statistical software (the R language in our case) such as vectorized operation and matrix calculation. Therefore it has by far the best running time when paired with Gaussian multiplier bootstrap. Random edge sampling and LatinCV has very similar performance in terms of running speed, and both will get further dragged if we add a low-rank matrix recovery step to it. When we compare the model selection accuracy, however, we find that random edge sampling is performing much better than LatinCV, although the latter promises equally representation of all nodes in each fold.

Another decision we are facing is how to obtain the reference distribution given the validated loss matrix. If we follow the cross-validation with confidence procedure closely, we would go with Gaussian multiplier bootstrap. But we also notice that it is designed mostly for some less accommodating scenarios (for example, large number of columns, unable to obtain validated loss bootstrap samples generatively). In our case, we can actually generatively obtain bootstrap samples of the validated loss matrix. We implemented the parametric bootstrap approach and paired it with three variants of train-test splitting method, and compared the model selection performance. It turns out that the reference distribution obtained through this approach isn't very ideal. Sometimes the algorithm wouldn't select any candidate K since none

of the P-values turn out to be greater than 0.05. This is also validated by comparing the P-value distribution under Null hypothesis.

After studying the properties of our method via extensive simulation studies, we applied the method on real-world data sets such as the US political blog data set and political book data set. We found that our method pretty consistently picks the correct number of communities for both data sets when restricting the candidate models to just one category of stochastic block models. When we put candidate models from both categories as inputs of the algorithm, we still find that the model widely accepted as the consensus truth (degree-corrected stochastic block model with $K = 2$) is contained in the confidence set a lot of the times for the political books data set, largely due to the fact that it has only 105 nodes and therefore the fold assignment would have significant impact on the model selection result. The political blog data set, on the other hand, doesn't get impacted by the randomness in fold assignment much, since it has over 1000 nodes and therefore is much more robust.

We studies the theoretical property of (one main variant of) our algorithm. It turns out the under mild conditions (i.e. if the density parameter diminishes at the rate of $\frac{\log n}{n}$, where n is the number of nodes in the network), our method with squared error loss and block-wise node-pair splitting will pick the correct model with guaranteed probability. The proof comes in two parts, we first show that any under-fitting model will be eliminated, since there is always an overhead when we try to model a more complicated data generating process with a simplistic model. We then show that any over-fitting model is unlikely to eliminate the true K value in the hypothesis testing. This part is more challenging since it is hard to model the behavior of over-fitting models. We had to add an assumption about the minimum community size (on the order of $\frac{n}{\log n}$) when using the over-fitting model. Since this is still smaller compared to the average community size $\frac{n}{K}$, it isn't a very aggressive additional assumption.

Possible future work include more extensively testing and packaging our code for the usage of the network community. We also note that it might be interesting to look into the deeper reason why parametric approach doesn't give equivalent results compared to the Gaussian multiplier bootstrap approach. Extending our theoretical work to more relaxed settings (for example, not assuming exact recovery, include DC-SBM, etc.) will also be an exciting future direction. Note that cross-validation with confidence is a framework that should broadly apply to most model selection problems where we can conduct classical cross-validation and have a finite candidate model space, therefore it makes sense to explore more areas to extend this framework to.

Appendix A

Proofs for results in 4.1

One result we will be using repeatedly is the following lemma:

Lemma 1.1

Suppose We have a set of node pairs $\tau = \{(i, j) | (i, j) \in \tau\} \subset A_{k,k'}$, and

$$A_{ij} \sim \text{Bernoulli}(B_{k,k'}^*), i.i.d$$

Then we have

$$\left| \frac{\sum_{(i,j) \in \tau} A_{ij}}{|\tau|} - B_{k,k'}^* \right| = O_P \left(\sqrt{\frac{\rho_n}{|\tau|}} \right)$$

where $\mathbb{E}(A_{i,j}) = B_{k,k'}^*, \forall (i, j) \in \tau \subset I_{k,k'}$.

Proof: Notice that here $A_{ij} \sim \text{Bernoulli}(B_{k,k'}^*)$ and $B_{k,k'}^* = c\rho_n$ for some constant c . Notice that

$$\mathbb{E} \left(\frac{\sum_{(i,j) \in \tau} A_{ij}}{|\tau|} - B_{k,k'}^* \right)^2 = \frac{1}{|\tau|^2} \text{Var} \left(\sum_{(i,j) \in \tau} A_{ij} \right) = \frac{B_{k,k'}^*(1 - B_{k,k'}^*)}{|\tau|} \sim \frac{\rho_n}{|\tau|}$$

Then using the Markov inequality, we have

$$\left(\frac{\sum_{(i,j) \in \tau} A_{ij}}{|\tau|} - B_{k,k'}^* \right)^2 = O_P \left(\frac{\rho_n}{|\tau|} \right)$$

and therefore

$$\left| \frac{\sum_{(i,j) \in \tau} A_{ij}}{|\tau|} - B_{k,k'}^* \right| = O_P \left(\sqrt{\frac{\rho_n}{|\tau|}} \right)$$

A direct corollary of the lemma 1.1 would be that if we have two sets containing each other $\tau_1 \subset \tau_2 \subset I_{k,k'}$, let \hat{P}_{τ_1} and \hat{P}_{τ_2} denote the average of A_{ij} 's over τ_1 and τ_2 , respectively. Then

$$|\hat{P}_{\tau_1} - \hat{P}_{\tau_2}| = O_P \left(\sqrt{\frac{\rho_n}{|\tau_1|}} \right)$$

A.1 When A_s is relatively large, and A_s^c grows sufficiently fast.

To be more specific, under this scenario we have $\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}| \leq \frac{(\pi'_0)^2}{8K^2} \cdot \binom{n}{2}$, yet still $\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}| \geq c_0 n$ for some constant c_0 . In other words, $\frac{c_0 n}{\binom{n}{2}} \leq w \leq \frac{(\pi'_0)^2}{8K^2}$.

When A_s is large enough, the extra loss within it $\sum_{\hat{I}_{k,k'} \in A_s} \sum_{(i,j) \in \hat{I}_{k,k'}} \xi^{(i,j)}$ will be lower bounded using pigeon hole principle in similar fashion as in [CL17]. But since A_s^c is also growing fast enough, Bernstein's inequality guarantees the extra loss cannot be too far away from its expectation. Therefore we can lower bound the test statistic in this case.

(i) Lower bound (I)

Given the assumptions above, here we want to show that

$$(I) \gtrsim \rho_n^2$$

We will be closely following the argument given in [CL17] using the pigeon hole principle. When we are considering an under fitting model, there has to be at least two true communities which have significant portions being falsely assigned to the same community label. The additional squared error loss is guaranteed to be large enough for this lower bound we want to show.

Since we have the above assumption, we know that $\forall \hat{I}_{k,k'} \in A_s^c, \min\{|\hat{I}_k|, |\hat{I}_{k'}|\} \leq \sqrt{2 \cdot \frac{(\pi'_0)^2}{8K^2} \cdot \binom{n}{2}} \leq \frac{\pi'_0 n}{2K}$ (considering it is possible that $k = k'$ and $\hat{I}_{k,k'}$ is a triangle instead of a rectangle). Let $C_{\tilde{K}} = \{k : |\hat{I}_k| > \frac{\pi'_0 n}{2K}\}$. Then if $k, k' \in C_{\tilde{K}}$, then $\hat{I}_{k,k'} \in A_s$, and we have

$$\begin{aligned} \sum_{k \in C_{\tilde{K}}} |\hat{I}_k| &= n - \sum_{k \notin C_{\tilde{K}}} |\hat{I}_k| \\ &\geq n - \tilde{K} \cdot \frac{\pi'_0 n}{2\tilde{K}} \\ &= n \cdot \left(1 - \frac{\pi'_0}{2}\right) \end{aligned}$$

In other words, the union of \hat{I}_k 's all the communities in $C_{\tilde{K}}$ would at most miss $\frac{\pi'_0 n}{2}$ nodes, which is half the size of the smallest possible community. Therefore, follow the argument of [CL17], $\exists k \in C_{\tilde{K}}$ and $1 \leq l_1 < l_2 < K$ such that $|\hat{I}_k \cap I_{l_j}| \geq \frac{|I_{l_j}|}{2\tilde{K}} \geq \frac{\pi'_0 n}{2\tilde{K}}$ for $j = 1, 2$. Since the community interaction matrix B_0 doesn't have identical rows, $\exists 1 \leq l_3 \leq K$ such that $B_0(l_1, l_3) \neq B_0(l_2, l_3)$. There exists $k' \in C_{\tilde{K}}$ such that $|\hat{I}_{k'} \cap I_{l_3}| \geq \frac{|I_{l_3}|}{2\tilde{K}} \geq \frac{\pi'_0 n}{2\tilde{K}}$. Without loss, we can set $k = 1, k' = 2, l_1 = 1, l_2 = 2, l_3 = 3$.

Let $\tau_{k,k',l,l'}$ represent the pairs (i, j) such that $i \in \hat{I}_k \cap I_l, j \in \hat{I}_{k'} \cap I_{l'}, i \neq j$. Let \hat{P} be the average of A_{ij} over $\tau_{1,2,1,3} \cup \tau_{1,2,2,3}$ and $\hat{P}_{k,k',l,l'}$ be the average of A_{ij} over $\tau_{k,k',l,l'}$. Here $B_{1,3}$ and $B_{2,3}$ represent the average of A_{ij} 's over $I_{1,3}$ and $I_{2,3}$. $I_{1,3}$ represents the pairs between community 1 and community 3 according to the labels assigned using community number K .

Then we have

$$\begin{aligned}
(I) &= \frac{1}{\sum_{\hat{I}_{k,k'} \in A_s} |\hat{I}_{k,k'}|} \sum_{\hat{I}_{k,k'} \in A_s} \sum_{(i,j) \in \hat{I}_{k,k'}} \xi^{(i,j)} \\
&= \frac{1}{\sum_{\hat{I}_{k,k'} \in A_s} |\hat{I}_{k,k'}|} \left(\sum_{\hat{I}_{k,k'} \in (C_{\tilde{K}} \times C_{\tilde{K}})} \sum_{(i,j) \in \hat{I}_{k,k'}} \xi^{(i,j)} + \sum_{\hat{I}_{k,k'} \in A_s \setminus (C_{\tilde{K}} \times C_{\tilde{K}})} \sum_{(i,j) \in \hat{I}_{k,k'}} \xi^{(i,j)} \right) \\
&= \frac{1}{\sum_{\hat{I}_{k,k'} \in A_s} |\hat{I}_{k,k'}|} \left(\sum_{(i,j) \in \tau_{1,2,1,3}} \xi^{(i,j)} + \sum_{(i,j) \in \tau_{1,2,2,3}} \xi^{(i,j)} + \sum_{\substack{\hat{I}_{k,k'} \in (C_{\tilde{K}} \times C_{\tilde{K}}) \\ (k,k',l,l') \notin \{(1,2,1,3), (1,2,2,3)\}}} \sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} \right. \\
&\quad \left. + \sum_{\hat{I}_{k,k'} \in A_s \setminus (C_{\tilde{K}} \times C_{\tilde{K}})} \sum_{(i,j) \in \hat{I}_{k,k'}} \xi^{(i,j)} \right) \\
&\geq \frac{1}{\sum_{\hat{I}_{k,k'} \in A_s} |\hat{I}_{k,k'}|} \left(\sum_{(i,j) \in \tau_{1,2,1,3}} [(A_{ij} - \hat{P})^2 - (A_{ij} - B_{1,3})^2] + \sum_{(i,j) \in \tau_{1,2,2,3}} [(A_{ij} - \hat{P})^2 - (A_{ij} - B_{2,3})^2] \right. \\
&\quad + \sum_{\substack{\hat{I}_{k,k'} \in (C_{\tilde{K}} \times C_{\tilde{K}}) \\ (k,k',l,l') \notin \{(1,2,1,3), (1,2,2,3)\}}} \sum_{(i,j) \in \tau_{k,k',l,l'}} [(A_{ij} - \hat{P}_{k,k',l,l'})^2 - (A_{ij} - B_{l,l'})^2] \\
&\quad \left. + \sum_{\hat{I}_{k,k'} \in A_s \setminus (C_{\tilde{K}} \times C_{\tilde{K}})} \sum_{(i,j) \in \hat{I}_{k,k'}} \xi^{(i,j)} \right)
\end{aligned}$$

Let $\lambda = \frac{|\tau_{1,2,1,3}|}{|\tau_{1,2,1,3}| + |\tau_{1,2,2,3}|}$. We know that $\frac{(\pi'_0)^2}{1 + (\pi'_0)^2} \leq \lambda \leq \frac{1}{1 + (\pi'_0)^2}$. We consider the first term first.

$$\begin{aligned}
& \sum_{(i,j) \in \tau_{1,2,1,3}} \xi^{(i,j)} \\
&= |\tau_{1,2,1,3}| [(\hat{P} - \hat{P}_{1,2,1,3})^2 - (\hat{P}_{1,2,1,3} - B_{1,3})^2] \\
&= |\tau_{1,2,1,3}| [(1-\lambda)^2 (\hat{P}_{1,2,1,3} - \hat{P}_{1,2,2,3})^2 - (\hat{P}_{1,2,1,3} - B_{1,3})^2] \text{ (... plugging in the definition of } \lambda) \\
&\geq |\tau_{1,2,1,3}| \left[\frac{(1-\lambda)^2}{2} (B_{1,3} - B_{2,3})^2 - (1-\lambda)^2 [(\hat{P}_{1,2,1,3} - B_{1,3}) - (\hat{P}_{1,2,2,3} - B_{2,3})]^2 - (\hat{P}_{1,2,1,3} - B_{1,3})^2 \right] \\
&\quad \text{(... difference is a perfect square)} \\
&\geq |\tau_{1,2,1,3}| \left[\frac{(1-\lambda)^2}{2} (B_{1,3} - B_{2,3})^2 - 2(1-\lambda)^2 (\hat{P}_{1,2,1,3} - B_{1,3})^2 - 2(1-\lambda)^2 (\hat{P}_{1,2,2,3} - B_{2,3})^2 \right. \\
&\quad \left. - (\hat{P}_{1,2,1,3} - B_{1,3})^2 \right] \\
&\quad \text{(... difference is a perfect square)} \\
&\geq c(n\rho_n)^2 + O_P(\rho_n)
\end{aligned}$$

The first term of the last step depends on the fact that $|\tau_{1,2,1,3}| \geq \frac{(\pi'_0)^2 n^2}{\tilde{K}^2}$, and $|B_{1,2} - B_{1,3}| \geq c' \rho_n$, where c' only depends on B_0 . The second step again comes from applying the corollary of Lemma 1.1.

Similarly we can lower bound the second term by $c(n\rho_n)^2 + O_P(\rho_n)$.

Also notice that

$$\begin{aligned}
\sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} &= \sum_{(i,j) \in \tau_{k,k',l,l'}} [(A_{ij} - \hat{P}_{k,k',l,l'})^2 - (A_{ij} - B_{l,l'})^2] \\
&= \sum_{(i,j) \in \tau_{k,k',l,l'}} \left[\left(A_{ij} - \frac{1}{|\tau_{k,k',l,l'}|} \sum_{(i,j) \in \tau_{k,k',l,l'}} A_{ij} \right)^2 - (A_{ij} - B_{l,l'})^2 \right] \\
&= -|\tau_{k,k',l,l'}| (B_{l,l'} - \hat{P}_{k,k',l,l'})^2 \\
&= O_P(\rho_n) \dots \dots \text{(Using Corollary of Lemma 1.1)}
\end{aligned}$$

where $\hat{P}_{k,k',l,l'}$ the average of A_{ij} over $\tau_{k,k',l,l'}$ and $B_{l,l'}$ represents the probability assigned to $I_{l,l'}$ in the exact model. We can thus limit the negative side of the latter two terms.

Therefore we have

$$(I) \gtrsim \frac{1}{\binom{n}{2}} (n^2 \rho_n^2 + O(\rho_n)) \gtrsim \rho_n^2$$

(ii) Lower bounding (II)

Here we want to show that

$$(II) \gtrsim t^2 \rho_n^2$$

where $t = \max_{k,k' \in \{1,2,\dots,\tilde{K}\}} t_{k,k'}$.

We will see that the expectation of (II) is on the order $t^2 \rho_n^2$. Since we are assuming A_s^c is also growing fast enough, we can apply Bernstein's inequality to $(II) - \mathbb{E}(II)$ and show that (II) cannot deviate far from its expectation with high probability. Therefore with a high probability it will be at least on the order of $t^2 \rho_n^2$.

For $\hat{I}_{k,k'} \in A_s^c$, we know $t_{k,k'} > s$ from the definition of A_s . For all the pairs $(i, j) \in \hat{I}_{k,k'}$, we have

$$\begin{aligned}\xi^{(i,j)} &= -2A_{ij}(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij}) + \hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2 \\ \left(\xi^{(i,j)}\right)^2 &= 4A_{ij}^2(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2 - 4A_{ij}(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})\left(\hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2\right) + \left(\hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2\right)^2 \\ (\text{noting } A_{ij}^2 = A_{ij}) &= 4A_{ij}(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2(1 - \hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij}) + \left(\hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2\right)^2\end{aligned}$$

And thus

$$\begin{aligned}\mathbb{E}\xi^{(i,j)} &= -2P_{ij}^*(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij}) + \hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2 \\ &\gtrsim t_{k,k'}^2 \rho_n^2 - 2t_{k,k'} \rho_n^2 \\ \mathbb{E}\left(\xi^{(i,j)}\right)^2 &= \mathbb{E}\left[4A_{ij}(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2(1 - \hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij}) + \left(\hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2\right)^2\right] \\ &\lesssim 4\rho_n \cdot t_{k,k'}^2 \rho_n^2 - 4\rho_n \cdot t_{k,k'} \rho_n \cdot t_{k,k'}^2 \rho_n^2 + (t_{k,k'}^2 \rho_n^2)^2 \\ &\lesssim t_{k,k'}^2 \rho_n^3 - 4t_{k,k'}^3 \rho_n^4 + t_{k,k'}^4 \rho_n^4\end{aligned}$$

we can set $\tilde{\xi}^{(i,j)} = \xi^{(i,j)} - \mathbb{E}(\xi^{(i,j)})$, it's easy to see it has mean 0 and variance $P_{ij}^*(1 - P_{ij}^*)(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2$, where P_{ij}^* denotes the true interaction probability between i and j and it would be on the order of ρ_n . Apply Bernstein's inequality to $\tilde{\xi}^{(i,j)}$'s over A_s^c :

$$\begin{aligned}&P\left((II) \leq \mathbb{E}(II) - c_1 t^2 \rho_n^2\right) \\ &= P\left(\frac{1}{\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|} \sum_{\hat{I}_{k,k'} \in A_s^c} \sum_{(i,j) \in \hat{I}_{k,k'}} \tilde{\xi}^{(i,j)} \leq -c_1 t^2 \rho_n^2\right) \\ &\leq \exp\left(-\frac{\frac{1}{2}\left(-c_1 t^2 \rho_n^2 \left(\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|\right)\right)^2}{\sum_{\hat{I}_{k,k'} \in A_s^c} \sum_{(i,j) \in \hat{I}_{k,k'}} E(\tilde{\xi}^{(i,j)})^2 + \frac{1}{3}M \cdot \left(c_1 t^2 \rho_n^2 \sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|\right)}\right) \\ &= \exp\left(-\frac{\frac{1}{2}c_1^2 t^4 \rho_n^4 \left(\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|\right)^2}{\left(\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|\right) P_{ij}^*(1 - P_{ij}^*)(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2 + \frac{1}{3}M \cdot (c_1 t^2 \rho_n^2 \sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|)}\right)\end{aligned}$$

Note here $M = \max\left\{|2P_{ij}^*(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})|, |2(P_{ij}^* - 1)(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})|\right\} \sim t\rho_n$

Therefore

$$\begin{aligned}
& P\left(\frac{1}{\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|} \sum_{\hat{I}_{k,k'} \in A_s^c} \sum_{(i,j) \in \hat{I}_{k,k'}} \tilde{\xi}^{(i,j)} \leq -c_1 t^2 \rho_n^2\right) \\
& \lesssim \exp\left(-\frac{\frac{1}{2} c_1^2 t^4 \rho_n^4 \left(\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|\right)^2}{\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}| (t^2 \rho_n^3 + \frac{1}{3} c_1 t^3 \rho_n^3)}\right) \\
& = \exp\left(-\frac{\frac{1}{2} c_1^2 t^2 \rho_n \sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}|}{1 + \frac{1}{3} c_1 t}\right)
\end{aligned}$$

Since we know $\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}| = O(n)$, $\rho_n \cdot \sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}| = O(\log n) \rightarrow \infty$. This probability is vanishing as $n \rightarrow \infty$. In conclusion, we know that with high probability

$$(II) \geq (1 - c_1) \mathbb{E}(II) \gtrsim (1 - c_1) t^2 \rho_n^2$$

We can set $c_1 = \frac{1}{2}$ for example and have that with high probability

$$(II) \gtrsim t^2 \rho_n^2$$

(iii) Upper bound (III)

Here we want to show that $(III) \lesssim s^2 \rho_n^2$.

For $(i, j) \in \hat{I}_{k,k'}$, we already know that

$$\begin{aligned}
\left(\xi^{(i,j)}\right)^2 &= 4A_{ij}^2 (\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2 - 4A_{ij} (\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij}) (\hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2) + (\hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2)^2 \\
(\text{noting } A_{ij}^2 &= A_{ij}) &= 4A_{ij} (\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2 (1 - \hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij}) + (\hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2)^2 \\
&\leq 4(\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2 (1 - \hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij}) + (\hat{P}_{\tilde{K},ij}^2 - \hat{P}_{K,ij}^2)^2 \\
&\lesssim 4(t_{k,k'} \rho_n)^2 + (t_{k,k'}^2 \rho_n^2)^2 \\
&\lesssim t_{k,k'}^2 \rho_n^2
\end{aligned}$$

Since for $\hat{I}_{k,k'} \in A_s$, we have $t_{k,k'} \leq s$.

$$\begin{aligned}
(III) &= \frac{1}{\sum_{\hat{I}_{k,k'} \in A_s} |\hat{I}_{k,k'}|} \left[\sum_{\hat{I}_{k,k'} \in A_s} \sum_{(i,j) \in \hat{I}_{k,k'}} \left(\xi^{(i,j)}\right)^2 \right] \\
&\lesssim s^2 \rho_n^2
\end{aligned}$$

(iv) Upper bounding (IV)

Similar to (iii), we have

$$(IV) \lesssim t^2 \rho_n^2$$

where $t = \max_{k,k' \in \{1,2,\dots,\tilde{K}\}} t_{k,k'}$

(v) Combining the results

From (i) - (iv) we know that

$$\begin{aligned} \frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} &\geq \frac{(1-w) \cdot (I) + w \cdot (II)}{\sqrt{(1-w) \cdot (III) + w \cdot (IV)}} \\ &\gtrsim \frac{(1-w) \cdot \rho_n^2 + w \cdot \frac{1}{2} t^2 \rho_n^2}{\sqrt{(1-w) \cdot s^2 \rho_n^2 + w \cdot t^2 \rho_n^2}} \text{ with high prob.} \end{aligned}$$

where $\frac{c_0 n}{\binom{n}{2}} \leq w \leq \frac{(\pi'_0)^2}{8K^2}$, in other words, w is between the order $\frac{1}{n}$ and a constant. Either w is on the order $\frac{1}{n}$ and the second part will be dominated in the weighted sum and we have

$$\frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \gtrsim \frac{\rho_n}{s}$$

or w is on the order of a constant, then we have

$$\frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \gtrsim \min\left\{\frac{\rho_n^2}{\sqrt{s^2 \rho_n^2}}, \frac{\frac{1}{2} t^2 \rho_n^2}{\sqrt{t^2 \rho_n^2}}\right\} \gtrsim \rho_n$$

Either way, we will always have

$$\frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \gtrsim \rho_n = \Omega\left(\frac{\log n}{n}\right)$$

A.2 When A_s is relatively large, and A_s^c grows very slowly.

More specifically, we have $\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}| \leq \frac{(\pi'_0)^2}{2K^2} \cdot \binom{n}{2}$, and furthermore $\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}| \leq c_0 n$. In other words, $w \leq \frac{c_0 n}{\binom{n}{2}}$.

Here we still assume that A_s is large enough yet A_s^c doesn't grow fast enough. Therefore now we cannot use the same argument to lower bound (II). Notice that we can limit the negative side of (II), and its weight in the numerator will be very small in this case, we can still achieve our ultimate goal.

In this case we can lower bound (I) and upper bound (III)(IV) in exactly the same way as in A.1.

And apply Lemma 1.1 to (II), noting now for each $\tau_{k,k',l,l'} \in A_s^c$, we have

$$\begin{aligned}
\sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} &= \sum_{(i,j) \in \tau_{k,k',l,l'}} (\hat{P}_{\tilde{K},ij} - A_{ij})^2 - (\hat{P}_{K,ij} - A_{ij})^2 \\
&= -|\tau_{k,k',l,l'}| (\hat{P}_{\tilde{K},ij} - \hat{P}_{K,ij})^2 \\
&\gtrsim -|\tau_{k,k',l,l'}| \frac{\rho_n}{|\tau_{k,k',l,l'}|} \\
&\gtrsim -\rho_n
\end{aligned}$$

Therefore

$$\begin{aligned}
(II) &= \frac{1}{\sum_{\tau_{k,k',l,l'} \in A_s^c} |\tau_{k,k',l,l'}|} \sum_{\tau_{k,k',l,l'} \in A_s^c} \sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} \geq \frac{K\tilde{K} \cdot (-\rho_n)}{\sum_{\tau_{k,k',l,l'} \in A_s^c} |\tau_{k,k',l,l'}|} \gtrsim -\rho_n \\
&\gtrsim \frac{\frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}}}{\sqrt{(1-w) \cdot (III) + w \cdot (IV)}} \\
&\gtrsim \frac{(1-w) \cdot \rho_n^2 - w \cdot \rho_n}{\sqrt{(1-w) \cdot s^2 \rho_n^2 + w \cdot t^2 \rho_n^2}}
\end{aligned}$$

Here we know that $w \leq \frac{c_0 n}{\binom{n}{2}} \lesssim \frac{1}{n}$. Therefore the first items in the numerator and denominator would both dominate the latter term. And we have

$$\frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \gtrsim \rho_n = \Omega\left(\frac{\log n}{n}\right)$$

A.3 When A_s is not large enough.

In other words, we have $\sum_{\hat{I}_{k,k'} \in A_s^c} |\hat{I}_{k,k'}| > \frac{(\pi'_0)^2}{2K^2} \cdot \binom{n}{2}$, and therefore $\sum_{\hat{I}_{k,k'} \in A_s} |\hat{I}_{k,k'}| \leq (1 - \frac{(\pi'_0)^2}{2K^2}) \cdot \binom{n}{2}$.

In other words, $w > \frac{(\pi'_0)^2}{2K^2}$.

Now we consider the case where A_s isn't large enough for the pigeon hole argument to hold. Now we can still bound (II), (III), (IV) using the same arguments as in A.1. We will instead bound the negative side of (I) and show that it doesn't affect the validity of the overall inequality.

Noting now for each $\tau_{k,k',l,l'} \in A_s$, let $\bar{A}_{k,k',l,l'}$ represent the average of A_{ij} 's over $\tau_{k,k',l,l'}$. We have the following

$$\begin{aligned}
\sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} &= \sum_{(i,j) \in \tau_{k,k',l,l'}} (\hat{P}_{\tilde{K},ij} - A_{ij})^2 - (\hat{P}_{K,ij} - A_{ij})^2 \\
&\geq \sum_{(i,j) \in \tau_{k,k',l,l'}} (\bar{A}_{k,k',l,l'} - A_{ij})^2 - (\hat{P}_{K,ij} - A_{ij})^2 \\
&= -|\tau_{k,k',l,l'}|(\bar{A}_{k,k',l,l'} - \hat{P}_{K,ij})^2 \\
&= |\tau_{k,k',l,l'}| O_P\left(\frac{\rho_n}{|\tau_{k,k',l,l'}|}\right) \\
&\gtrsim -\rho_n
\end{aligned}$$

Therefore we have

$$\begin{aligned}
(I) &= \frac{1}{\sum_{\tau_{k,k',l,l'} \in A_s} |\tau_{k,k',l,l'}|} \sum_{\tau_{k,k',l,l'} \in A_s} \sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} \\
&\gtrsim \frac{K\tilde{K} \cdot (-\rho_n)}{\sum_{\tau_{k,k',l,l'} \in A_s} |\tau_{k,k',l,l'}|} \\
&\gtrsim -\rho_n
\end{aligned}$$

So now

$$\begin{aligned}
&\frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \\
&\gtrsim \frac{(1-w) \cdot (I) + w \cdot (II)}{\sqrt{(1-w) \cdot (III) + w \cdot (IV)}} \\
&\gtrsim \frac{\frac{1}{\binom{n}{2}} \cdot (-\rho_n) + w \cdot t^2 \rho_n^2}{\sqrt{(1-w) \cdot s^2 \rho_n^2 + w \cdot t^2 \rho_n^2}} (\dots \text{ plug in the definition of } (1-w))
\end{aligned}$$

where $\frac{(\pi'_0)^2}{2\tilde{K}^2} < w \leq 1$. In the numerator the second term dominates and in the denominator both terms are on the same order, therefore we get

$$\frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \gtrsim \rho_n = \Omega\left(\frac{\log n}{n}\right)$$

Combining A.1 \sim AsSmall, we know that in the under fitting case,

$$\frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \gtrsim \Omega\left(\frac{\log n}{n}\right)$$

Therefore

$$\begin{aligned}
& P(T_{\tilde{K}} \geq \sqrt{2 \log n}) \\
= & P\left[\sqrt{\frac{n^2}{V^2}} \frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \geq \sqrt{2 \log n}\right] \\
= & P\left[\frac{1}{V} \frac{\hat{\mu}_{\tilde{K},K}}{\hat{\sigma}_{\tilde{K},K}} \geq \frac{\sqrt{2 \log n}}{n}\right] \\
= & P\left[\Omega\left(\frac{\log n}{n}\right) \geq \frac{\sqrt{2 \log n}}{n}\right] \\
\rightarrow & 1
\end{aligned}$$

In summary, using our method, the under fitting models will always be rejected.

Appendix B

Proofs for results in 4.2

Here we assume we have a network with K^* true communities and we are fitting a model with $\tilde{K} > K^*$. Again, let I_k represent the set of nodes in the k th true community, while \hat{I}_l represent the l th community labeled by the over-fitting model. Let $\hat{I}_{k,k'}^{(K,tr)}$ represent the node pairs (i, j) Let $\tau_{k,k',l,l'}$ represent the pairs (i, j) such that $i \in \hat{I}_k \cap I_l, j \in \hat{I}_{k'} \cap I_{l'}, i \neq j$, here $k, k' \in \{1, \dots, K\}$ and $l, l' \in \{1, \dots, \tilde{K}\}$. $\hat{I}_k^{(K,tr)}$ represents the k th community identified by model with K in the training set, and $n_k^{(K,tr)}$ represents its size. Similarly with $\hat{I}_k^{(K,tr)}$ and $n_k^{(K,te)}$. $\hat{I}_k^{(K)}$ represents the k th community, in both training and testing set.

Here $\hat{B}_{k,k'}^{(K,tr)}$ represent the edge probability estimate for node pairs (i, j) where $i \in \hat{I}_k, j \in \hat{I}_{k'}$ when we use a model with K (when K is the correct value, then $I_k = \hat{I}_k$ since we assume exact recovery of communities). These edge probability estimates are the empirical mean of edges on the corresponding set in the rectangular set:

$$\hat{B}_{k,k}^{(K,tr)} = \frac{1}{\binom{n_k^{(K,tr)}}{2} + n_k^{(K,tr)} \cdot n_k^{(K,te)}} \sum_{\substack{(i,j) \in \hat{I}_k^{(K,tr)} \times \hat{I}_k \\ i < j}} A_{ij}$$

$$\hat{B}_{k,k'}^{(K,tr)} = \frac{1}{n_k^{(K,tr)} n_{k'}^{(K,tr)} + n_k^{(K,tr)} \cdot n_{k'}^{(K,te)} + n_{k'}^{(K,tr)} \cdot n_k^{(K,te)}} \sum_{\substack{(i,j) \in \left(\hat{I}_k^{(K,tr)} \times \hat{I}_{k'}^{(K,tr)} \right) \\ \cup \left(\hat{I}_k^{(K,te)} \times \hat{I}_{k'}^{(K,te)} \right) \\ \cup \left(\hat{I}_{k'}^{(K,te)} \times \hat{I}_k^{(K,te)} \right)}} A_{ij}$$

$$\begin{aligned}
\text{numerator} &= \frac{1}{\binom{n^{(te)}}{2}} \sum_{k,k',l,l'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} \\
&= \frac{1}{\binom{n^{(te)}}{2}} \sum_{k,k',l,l'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \left[(A_{ij} - \hat{B}_{k,k'}^{(K,tr)})^2 - (A_{ij} - \hat{B}_{l,l'}^{(\tilde{K},tr)})^2 \right] \\
&= \frac{1}{\binom{n^{(te)}}{2}} \sum_{k,k',l,l'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (2A_{ij} - \hat{B}_{k,k'}^{(K,tr)} - \hat{B}_{l,l'}^{(\tilde{K},tr)}) (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \\
&= \frac{1}{\binom{n^{(te)}}{2}} \sum_{l,l'} \left[\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (2A_{ij} - \hat{B}_{k,k'}^{(K,tr)} - \hat{B}_{l,l'}^{(\tilde{K},tr)}) (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \right] \\
&= \frac{1}{\binom{n^{(te)}}{2}} \sum_{l,l'} \left[\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} 2A_{ij} (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) + \left(\hat{B}_{k,k'}^{(K,tr)} \right)^2 - \left(\hat{B}_{l,l'}^{(\tilde{K},tr)} \right)^2 \right]
\end{aligned}$$

We use the following set to denote such combinations of (l, l') : $D_{c_0} = \{(l, l') : \hat{B}_{l,l'}^{(\tilde{K},tr)} > c_0 \rho_n\}$. Here we let c_0 be a constant that is much larger than 1. And we consider the following two scenarios:

B.1 When all the estimates are moderate from the over-fitting model

In this case $D_{c_0} = \emptyset$. i.e. $\hat{B}_{l,l'}^{(\tilde{K},tr)} \leq c_0 \rho_n$ for all (l, l') .

Here we can focus on only the summation over $I_{l,l'}^{(\tilde{K},te)}$ in the numerator, which is the following:

$$\begin{aligned}
&\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (2A_{ij} - \hat{B}_{k,k'}^{(K,tr)} - \hat{B}_{l,l'}^{(\tilde{K},tr)}) (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \\
&= \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \left[2(A_{ij} - \hat{B}_{k,k'}^{(K,tr)}) - (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \\
&= \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} 2(A_{ij} - \hat{B}_{k,k'}^{(K,tr)}) (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) - \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \\
&\leq \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} 2(A_{ij} - \hat{B}_{k,k'}^{(K,tr)}) (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \\
&= \sum_{k,k'} \left[\left(\sum_{(i,j) \in \tau_{k,k',l,l'}} A_{ij} - |\tau_{k,k',l,l'}| \hat{B}_{k,k'}^{(K,tr)} \right) (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \right]
\end{aligned}$$

Using Lemma 1.1, we know that

$$\left| \frac{\sum_{(i,j) \in \tau_{k,k',l,l'}} A_{ij}}{|\tau_{k,k',l,l'}|} - B_{k,k'}^* \right| = O_P\left(\sqrt{\frac{\rho_n}{|\tau_{k,k',l,l'}|}}\right)$$

therefore

$$\left| \sum_{(i,j) \in \tau_{k,k',l,l'}} A_{ij} - |\tau_{k,k',l,l'}| B_{k,k'}^* \right| = O_P(\sqrt{|\tau_{k,k',l,l'}| \rho_n})$$

Also,

$$|\hat{B}_{k,k'}^{(K,tr)} - B_{k,k'}^*| = O_P\left(\sqrt{\frac{\rho_n}{|\hat{I}_{k,k'}^{(K,tr)}|}}\right)$$

Noting that $\tau_{k,k',l,l'} \subset \hat{I}_{k,k'}^{(K,te)}$ and $|\hat{I}_{k,k'}^{(K,te)}| < \hat{I}_{k,k'}^{(K,tr)}$. We can combine these two and we know that

$$\sum_{(i,j) \in \tau_{k,k',l,l'}} A_{ij} - |\tau_{k,k',l,l'}| \hat{B}_{k,k'}^{(K,tr)} = O_P(\sqrt{|\tau_{k,k',l,l'}| \rho_n})$$

By definition of the O_p operator, we know that $\forall \epsilon > 0, \exists M > 0$, such that

$$P\left(\left| \sum_{(i,j) \in \tau_{k,k',l,l'}} A_{ij} - |\tau_{k,k',l,l'}| \hat{B}_{k,k'}^{(K,tr)} \right| > M \cdot \sqrt{|\tau_{k,k',l,l'}| \rho_n}\right) < \epsilon$$

In other words, $\exists c_1 > 0$ that is large enough s.t.

$$\left| \sum_{(i,j) \in \tau_{k,k',l,l'}} A_{ij} - |\tau_{k,k',l,l'}| \hat{B}_{k,k'}^{(K,tr)} \right| \leq c_1 \cdot \sqrt{|\tau_{k,k',l,l'}| \rho_n} \quad \text{with high prob. } \forall (k, k', l, l')$$

For the denominator, we have

$$\begin{aligned} & \text{denominator}^2 \\ = & \frac{1}{\binom{n^{(te)}}{2}} \sum_{k,k',l,l'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (\xi^{(i,j)})^2 \\ = & \frac{1}{\binom{n^{(te)}}{2}} \sum_{k,k',l,l'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (2A_{ij} - \hat{B}_{k,k'}^{(K,tr)} - \hat{B}_{l,l'}^{(\tilde{K},tr)})^2 (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \\ = & \frac{1}{\binom{n^{(te)}}{2}} \sum_{k,k',l,l'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \left[4A_{ij} - 4A_{ij}(\hat{B}_{k,k'}^{(K,tr)} + \hat{B}_{l,l'}^{(\tilde{K},tr)}) + (\hat{B}_{k,k'}^{(K,tr)} + \hat{B}_{l,l'}^{(\tilde{K},tr)})^2 \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \end{aligned}$$

We can again look at the sum over $I_{l,l'}^{(\tilde{K},te)}$ first:

$$\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \left[4A_{ij} - 4A_{ij}(\hat{B}_{k,k'}^{(K,tr)} + \hat{B}_{l,l'}^{(\tilde{K},tr)}) + (\hat{B}_{k,k'}^{(K,tr)} + \hat{B}_{l,l'}^{(\tilde{K},tr)})^2 \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2$$

Notice in the first bracket, $\hat{B}_{l,l'}^{(\tilde{K},tr)} \leq c_0 \rho_n$ by assumption of this scenario, and $\hat{B}_{k,k'}^{(K,tr)} \sim \rho_n$ since $|I_{k,k'}| \propto n^2$ by the assumption of the theorem and $I_{k,k'}^{(K,tr)}$ is a random subset of it. Therefore they are both on the order ρ_n . We can find a positive constant $c_2 > 0$, such that

$$|\hat{B}_{l,l'}^{(\tilde{K},tr)} + \hat{B}_{k,k'}^{(K,tr)}| \leq c_2 \rho_n, \forall k, k', l, l'$$

The the above sum

$$\begin{aligned} & \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \left[4A_{ij} - 4A_{ij}(\hat{B}_{k,k'}^{(K,tr)} + \hat{B}_{l,l'}^{(\tilde{K},tr)}) + (\hat{B}_{k,k'}^{(K,tr)} + \hat{B}_{l,l'}^{(\tilde{K},tr)})^2 \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \\ \leq & \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \left[4A_{ij} + 4A_{ij}c_2\rho_n + c_2^2\rho_n^2 \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \\ = & \sum_{(i,j) \in \hat{I}_{l,l'}^{(\tilde{K},te)}} \left[4A_{ij} + 4A_{ij}c_2\rho_n + c_2^2\rho_n^2 \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \\ = & |\hat{I}_{l,l'}^{(\tilde{K},te)}| \left[4\hat{B}_{l,l'}^{(\tilde{K},te)} + 4c_2\hat{B}_{l,l'}^{(\tilde{K},te)}\rho_n + c_2^2\rho_n^2 \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \\ = & \left(\sum_{k,k'} |\tau_{k,k',l,l'}| \right) \cdot \left[4\hat{B}_{l,l'}^{(\tilde{K},te)} + 4c_2\hat{B}_{l,l'}^{(\tilde{K},te)}\rho_n + c_2^2\rho_n^2 \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \end{aligned}$$

Here since we have the size assumption of $\hat{I}_{l,l'}^{(\tilde{K},tr)}$, we know that

$$\left| \hat{B}_{l,l'}^{(\tilde{K},te)} - B_{l,l'}^* \right| \leq c_3 \rho_n$$

for some $c_3 > 0$.

Therefore within the bracket, the first term is on the order ρ_n , the second and third term are on the order ρ_n^2 . The first term is the leading term. Thus

$$\begin{aligned} & \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \left[4A_{ij} - 4A_{ij}(\hat{B}_{k,k'}^{(K,tr)} + \hat{B}_{l,l'}^{(\tilde{K},tr)}) + (\hat{B}_{k,k'}^{(K,tr)} + \hat{B}_{l,l'}^{(\tilde{K},tr)})^2 \right] (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \\ \gtrsim & \left(\sum_{k,k'} |\tau_{k,k',l,l'}| \right) c_4 \rho_n (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \end{aligned}$$

for some $c_4 > 0$. We achieve this lower bound by dropping the less significant term (on the order ρ_n^2) and only retain the leading term.

We know therefore that

$$\begin{aligned}
&= \frac{\text{denominator}^2}{\binom{n^{(te)}}{2}} \sum_{l,l'} \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (\xi^{(i,j)})^2 \\
&\gtrsim \frac{1}{\binom{n^{(te)}}{2}} \sum_{l,l'} \left(\sum_{k,k'} |\tau_{k,k',l,l'}| c_4 \rho_n \left(\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)} \right)^2 \right) \\
&= \frac{1}{\binom{n^{(te)}}{2}} \sum_{l,l'} \sum_{k,k'} |\tau_{k,k',l,l'}| c_4 \rho_n \left(\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)} \right)^2
\end{aligned}$$

And then consider the test statistic as a whole:

$$\begin{aligned}
T_{K^*, \tilde{K}} &= \sqrt{\frac{n^2}{V^2}} \frac{\frac{1}{\binom{n^{(te)}}{2}} \sum_{l,l'} \left(\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} \right)}{\sqrt{\frac{1}{\binom{n^{(te)}}{2}} \sum_{l,l'} \left(\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (\xi^{(i,j)})^2 \right)}} \\
&\lesssim \frac{n}{V \cdot \sqrt{\binom{n^{(te)}}{2}}} \frac{\sum_{(l,l')} \left(\sum_{k,k'} c_1 \sqrt{|\tau_{k,k',l,l'}| \rho_n} (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \right)}{\sqrt{\sum_{(l,l')} \left(\sum_{k,k'} (c_4 |\tau_{k,k',l,l'}| \rho_n (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \right)}} \\
&\leq \frac{n}{V \cdot \sqrt{\binom{n^{(te)}}{2}}} \frac{c_1}{\sqrt{c_4 K \tilde{K}}} \dots \text{Cauchy-Schwartz inequality}
\end{aligned}$$

Here the upper bound for $T_{K^*, \tilde{K}}$ is a constant that doesn't change with n . It is influenced by some constants only such as c_1, c_4, V . Therefore in this scenario we have

$$T_{K^*, \tilde{K}} = O_P(1)$$

B.2 When some estimates are extreme from the over-fitting model

In this case $D_{c_0} \neq \emptyset$. i.e. for some $(l, l'), \hat{B}_{l,l'}^{(\tilde{K},tr)} > c_0 \rho_n$.

Then we consider the terms in the numerator and denominator corresponding to $(l, l') \in D_{c_0}$ separately.

For the numerator, the term corresponding to (l, l') is

$$\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} 2A_{ij} (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) + \left(\hat{B}_{k,k'}^{(K,tr)} \right)^2 - \left(\hat{B}_{l,l'}^{(\tilde{K},tr)} \right)^2$$

Since we have the size assumption about true communities, we know that

$$\left| \hat{B}_{k,k'}^{(K,tr)} - B_{k,k'}^* \right| \lesssim \rho_n$$

And therefore $c_5 \rho_n < \hat{B}_{k,k'}^{(K,tr)} < c_6 \rho_n$ for some $0 < c_5 < c_6$.

Therefore for the above term we have

$$\begin{aligned} & \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} 2A_{ij} (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) + \left(\hat{B}_{k,k'}^{(K,tr)} \right)^2 - \left(\hat{B}_{l,l'}^{(\tilde{K},tr)} \right)^2 \\ & \leq \sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} 2A_{ij} (\hat{B}_{l,l'}^{(\tilde{K},tr)} - c_5 \rho_n) + (c_6 \rho_n)^2 - \left(\hat{B}_{l,l'}^{(\tilde{K},tr)} \right)^2 \\ & = \left(\sum_{k,k'} |\tau_{k,k',l,l'}| \right) \cdot \left[2\hat{B}_{l,l'}^{(\tilde{K},te)} (\hat{B}_{l,l'}^{(\tilde{K},tr)} - c_5 \rho_n) + (c_6 \rho_n)^2 - \left(\hat{B}_{l,l'}^{(\tilde{K},tr)} \right)^2 \right] \end{aligned}$$

Here we know that $\left(\hat{B}_{l,l'}^{(\tilde{K},tr)} \right)^2 > c_0^2 \rho_n^2$, while $|\hat{B}_{l,l'}^{(\tilde{K},te)} - B_{l,l'}^*| \leq c_7 \rho_n$, when c_0 is the largest constant among $\{c_0, c_5, c_6, c_7\}$ (which we can make sure by choice), the last term would be the largest in the bracket. Therefore this term is negative in the numerator.

While at the same time, the denominator term

$$\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \left(\xi_{ij} \right)^2$$

is clearly non-negative. If we drop both terms, the ratio will only become larger. And we have

$$\begin{aligned} T_{K^*, \tilde{K}} &= \sqrt{\frac{n^2}{V^2}} \frac{\frac{1}{\binom{n^{(te)}}{2}} \sum_{l,l'} \left(\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} \right)}{\sqrt{\frac{1}{\binom{n^{(te)}}{2}} \sum_{l,l'} \left(\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (\xi^{(i,j)})^2 \right)}} \\ &\leq \sqrt{\frac{n^2}{V^2}} \frac{\frac{1}{n \binom{n^{(te)}}{2}} \sum_{(l,l') \notin D_{c_0}} \left(\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} \xi^{(i,j)} \right)}{\sqrt{\sum_{(l,l') \notin D_{c_0}} \left(\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} (\xi^{(i,j)})^2 \right)}} \\ &\leq \frac{n}{V \cdot \sqrt{\binom{n^{(te)}}{2}}} \frac{\sum_{(l,l') \notin D_{c_0}} \left(\sum_{k,k'} \sum_{(i,j) \in \tau_{k,k',l,l'}} c_1 \sqrt{|\tau_{k,k',l,l'}| \rho_n} (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)}) \right)}{\sqrt{\sum_{(l,l') \notin D_{c_1}} \left(\sum_{k,k'} (c_4 |\tau_{k,k',l,l'}| \rho_n (\hat{B}_{l,l'}^{(\tilde{K},tr)} - \hat{B}_{k,k'}^{(K,tr)})^2 \right)}} \\ &\leq \frac{n}{V \cdot \sqrt{\binom{n^{(te)}}{2}}} \frac{c_1}{\sqrt{c_4 K \tilde{K}}} \dots \text{Cauchy-Schwartz inequality} \end{aligned}$$

Again, the upper bound of $T_{K^*, \tilde{K}}$ is a constant that doesn't change with n . And again we have

$$T_{K^*, \tilde{K}} = O_P(1)$$

for $\tilde{K} > K^*$.

Combining the results from the two scenarios, we conclude our proof for the theorem.

Bibliography

- [ABFX08] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008. 3
- [ABH16] Emmanuel Abbe, Afonso S Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2016. 3
- [ACBL13] Arash A Amini, Aiyou Chen, Peter J Bickel, and Elizaveta Levina. Pseudo-likelihood methods for community detection in large sparse networks. *The Annals of Statistics*, 41(4):2097–2122, 2013. 3
- [AG05] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43. ACM, 2005. 67
- [AS15] Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 670–688. IEEE, 2015. 75
- [BC09] Peter J Bickel and Aiyou Chen. A nonparametric view of network models and newman–girvan and other modularities. *Proceedings of the National Academy of Sciences*, pages pnas–0907096106, 2009. 3
- [BCCZ13] Peter Bickel, David Choi, Xiangyu Chang, and Hai Zhang. Asymptotic normality of maximum likelihood and its variational approximation for stochastic blockmodels. *The Annals of Statistics*, 41(4):1922–1943, 2013. 3
- [BS16] Peter J Bickel and Purnamrita Sarkar. Hypothesis testing for automated community detection in networks. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(1):253–273, 2016. 3
- [CCK⁺13] Victor Chernozhukov, Denis Chetverikov, Kengo Kato, et al. Gaussian approximations and multiplier bootstrap for maxima of sums of high-dimensional random vectors. *The Annals of Statistics*, 41(6):2786–2819, 2013. 19, 24
- [CDP12] Alain Celisse, Jean-Jacques Daudin, and Laurent Pierre. Consistency of maximum-likelihood and variational estimators in the stochastic block model. *Electronic Journal of Statistics*, 6:1847–1899, 2012. 3

- [CL15] Etienne Côme and Pierre Latouche. Model selection and clustering in stochastic block models based on the exact integrated complete data likelihood. *Statistical Modelling*, 15(6):564–589, 2015. 4
- [CL17] Kehui Chen and Jing Lei. Network cross-validation for determining the number of communities in network data. *Journal of the American Statistical Association*, pages 1–11, 2017. xi, xii, 4, 11, 12, 13, 17, 32, 38, 44, 54, 76, 78, 79, 88, 89
- [CWA12] David S Choi, Patrick J Wolfe, and Edoardo M Airoldi. Stochastic blockmodels with a growing number of classes. *Biometrika*, 99(2):273–284, 2012. 75
- [CX14] Yudong Chen and Jiaming Xu. Statistical-computational tradeoffs in planted problems and submatrix localization with a growing number of clusters and submatrices. *arXiv preprint arXiv:1402.1267*, 2014. 75
- [Dab16] Beau Dabbs. *Characteristics of cross-validation methods for model selection in the stochastic block networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2016. xi, 11, 12
- [DJ16] Beau Dabbs and Brian Junker. Comparison of cross-validation methods for stochastic block models. *arXiv preprint arXiv:1605.03000*, 2016. xi, 12, 13
- [DKMZ11] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011. 3
- [DPR08] J-J Daudin, Franck Picard, and Stéphane Robin. A mixture model for random graphs. *Statistics and computing*, 18(2):173–183, 2008. 3
- [EFA⁺11] Dirk Eddelbuettel, Romain François, J Allaire, Kevin Ushey, Qiang Kou, N Russell, John Chambers, and D Bates. Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. 65
- [EK⁺10] David Easley, Jon Kleinberg, et al. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010. 2
- [FST⁺13] Donniell E Fishkind, Daniel L Sussman, Minh Tang, Joshua T Vogelstein, and Carey E Priebe. Consistent adjacency-spectral partitioning for the stochastic block model when the model parameters are unknown. *SIAM Journal on Matrix Analysis and Applications*, 34(1):23–39, 2013. 3
- [FW81] Stephen E. Fienberg and Stanley Wasserman. Categorical data analysis of single sociometric relations. In Samuel Leinhardt, editor, *Sociological Methodology 1981*, pages 156–192. Jossey-Bass, San Francisco, 1981. 2
- [GN02] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences (USA)*, 99:7821–7826, 2002. 3
- [GS18] John M Griffin and Amin Shams. Is bitcoin really un-tethered? 2018. xi, 1
- [GZF⁺10] Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, Edoardo M Airoldi, et al. A survey of statistical network models. *Foundations and Trends® in Machine Learning*, 2(2):129–233, 2010. 2

- [HLL83] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983. 2
- [Hof08] Peter Hoff. Modeling homophily and stochastic equivalence in symmetric relational data. In *Advances in neural information processing systems*, pages 657–664, 2008. 4, 10, 13
- [HRH02] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002. 2
- [HRT07] Mark S Handcock, Adrian E Raftery, and Jeremy M Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007. 32
- [HW08] Jake M. Hofman and Chris H. Wiggins. A Bayesian approach to network modularity. *Physical Review Letters*, 100:258701, 2008. 4
- [Jac10] Matthew O Jackson. *Social and economic networks*. Princeton university press, 2010. xi, 1, 2
- [Jin15] Jiashun Jin. Fast community detection by score. *The Annals of Statistics*, 43(1):57–89, 2015. 3, 67
- [KMM⁺13] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 110(52):20935–20940, 2013. 3
- [KN11] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011. 2, 67
- [LBA12] Pierre Latouche, Etienne Birmele, and Christophe Ambroise. Variational bayesian inference and complexity control for stochastic block models. *Statistical Modelling*, 12(1):93–115, 2012. 3
- [Lei16] Jing Lei. A goodness-of-fit test for stochastic block models. *The Annals of Statistics*, 44(1):401–424, 2016. 4, 67
- [Lei17] Jing Lei. Cross-validation with confidence. *arXiv preprint arXiv:1703.07904*, 2017. 4, 7, 9, 18, 19, 20, 31
- [Lei18] Jing Lei. Network representation using graph root distributions. *arXiv preprint arXiv:1802.09684*, 2018. 67
- [Lin10] Crystal Linkletter. Statistical analysis of network data: Methods and models by kolaczyk, ed. *Biometrics*, 66(2):663–664, 2010. 2
- [LL15] Can M Le and Elizaveta Levina. Estimating the number of communities in networks by spectral methods. *arXiv preprint arXiv:1507.00827*, 2015. 4
- [LLZ16] Tianxi Li, Elizaveta Levina, and Ji Zhu. Network cross-validation by edge sampling. *arXiv preprint arXiv:1612.04717*, 2016. xi, 4, 10, 12, 13, 15

- [LR15] Jing Lei and Alessandro Rinaldo. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 43(1):215–237, 2015. 3
- [MMFH13] Aaron F McDaid, Thomas Brendan Murphy, Nial Friel, and Neil J Hurley. Improved bayesian inference for the stochastic block model with application to large networks. *Computational Statistics & Data Analysis*, 60:12–31, 2013. 4
- [New06] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006. 3
- [New10] Mark Newman. *Networks: an introduction*. Oxford university press, 2010. 2
- [NG03a] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2003. 3
- [NG03b] Mark E. J. Newman and Michelle Girvan. Mixing patterns and community structure in networks. In R. Pastor-Satorras, J. Rubi, and A. Diaz-Guilera, editors, *Statistical Mechanics of Complex Networks*, pages 66–87, Berlin, 2003. Springer-Verlag. 3
- [PA93] John F Padgett and Christopher K Ansell. Robust action and the rise of the medici, 1400-1434. *American journal of sociology*, 98(6):1259–1319, 1993. xi, 1
- [RCPHLR⁺16] Francisco J Romero-Campero, Ignacio Perez-Hurtado, Eva Lucas-Reina, Jose M Romero, and Federico Valverde. Chlamynet: a chlamydomonas gene co-expression network reveals global properties of the transcriptome and the early setup of key co-expression patterns in the green lineage. *BMC genomics*, 17(1):227, 2016. xi, 1
- [RCY11] Karl Rohe, Sourav Chatterjee, and Bin Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 39(4):1878–1915, 2011. 3
- [Sha93] Jun Shao. Linear model selection by cross-validation. *Journal of the American statistical Association*, 88(422):486–494, 1993. 4, 9
- [VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. 14
- [Vuo89] Quang H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57:307–333, 1989. 3
- [WB17] YX Rachel Wang and Peter J Bickel. Likelihood-based model selection for stochastic block models. *The Annals of Statistics*, 45(2):500–528, 2017. 4
- [YSJ⁺14] Xiaoran Yan, Cosma Rohilla Shalizi, Jacob E. Jensen, Florent Krzakala, Christopher Moore, Lenka Zdeborova, Pan Zhang, and Yaojia Zhu. Model selection for degree-corrected block models. *Journal of Statistical Mechanics: Theory and Experiment*, 2014:P05007, 2014. 3
- [Zha93] Ping Zhang. Model selection via multifold cross validation. *The Annals of Statistics*, pages 299–313, 1993. 4, 9

- [ZLZ11] Yunpeng Zhao, Elizaveta Levina, and Ji Zhu. Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 2011. 3, 67