KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

From graph patterns to networked statistics

Yuyi Wang

Supervisor:
Prof. Maurice Bruynooghe
Dr. Jan Ramon

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering Science: Computer Science

June 2015

From graph patterns to networked statistics

Yuyi WANG

Examination committee:
Prof. Pierre Verbaeten, chair
Prof. Maurice Bruynooghe, supervisor
Dr. Jan Ramon, supervisor
Prof. Hendrik Blockeel
Prof. Marc Van Barel
Prof. Toon Calders
(Université Libre de Bruxelles)
Prof. Rui Castro
(Technische Universiteit Eindhoven)

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Engineering Science: Computer Science

June 2015

ISBN XXX-XX-XXXX-XXX-X
D/XXXX/XXXX/XX

Acknowledgements

First of all, I would like to express my sincere gratefulness to my two supervisors: dr. Jan Ramon, who patiently guided me to many interesting research questions and showed insights to tackle these challenges, prof. Maurice Bruynooghe, who carefully read my texts and provided thoughtful comments. If without their support, writing this thesis would be impossible. I would also like to thank the members of the examination committee: prof. Hendrik Blockeel, prof. Marc Van Barel, prof. Toon Calders and prof. Rui Castro for providing many suggestions to improve this thesis and ideas for future work. In addition, I am grateful to prof. Pierre Verbaeten for charing this examination committee.

I owe my deep thankfulness to the professors I visited during my PhD career: prof. Dingxuan Zhou, prof. Haijun Zhou, prof. Xiaodong Hu, prof. Zheng-Chu Guo and prof. Zhongping Qin. I learned a lot from the discussions with them.

Many thanks of course to all my former and current colleagues for creating an inspiring and relaxing working environment. I would also like to say "thank you" to my friends who helped me in overcoming several difficulties during my PhD career.

Last but not least, I would also thank my family: my grand parents, my parents, my cousin and her parents.

Abstract

The amount of available data grows rapidly nowadays. Data mining and machine learning techniques deal with discovering interesting knowledge from data and improving the performance of methods to do so. When the collected data points are independently drawn, these tools work well in practice. Statistical theories make sure that, if the sample size is large enough, then the mined knowledge is very likely to be convincing, and the trained model has a good generalization ability. However, the structure of available data becomes more and more complex, which violates the assumption that every data point is independent of each other. The fact that data points are dependent usually makes the data less informative.

We begin with the task of defining support measures which gauge the frequency of a given pattern in a given dataset. If the dataset is transactional, one only needs to count the occurrences of the pattern in the dataset. If the dataset is a large network, occurrences of a subgraph pattern may overlap, which makes the support measure definition less straightforward. We lose the important property of anti-monotonicity, which allows us to effectively prune search spaces, if we ignore overlaps and directly count the occurrences. An important class of anti-monotonic support measures relies on overlap graphs, but all earlier overlap graph based support measures are expensive to compute. In this thesis, we introduce the concept of overlap hypergraphs, and propose an overlap hypergraph based support which is anti-monotonic and can be computed efficiently.

In order to understand this support measure from a statistical point of view, we consider the problem of statistics on networked examples. We build a model for networked examples and represent them by hypergraphs, and make a reasonable assumption to replace the classical i.i.d. assumption. Based on this model, we design an estimator on networked examples with the minimum variance and generalize Chernoff-Hoeffding inequalities to weighted estimators on networked examples. We then minimize these bounds by properly choosing a weight vector

which is closely related to the graph support measure we proposed.

These statistical concentration results are used to adapt some crucial learning principles, e.g., empirical risk minimization, to the networked case, and show generalization error bounds for learning from networked examples.

List of Abbreviations

ANOVA	analysis of variance
CC	clique contraction
cf.	confer (compare)
ect.	et alii (and others)
Eq.	Equation
EQW	equal weights
ER	edge removal
ERM	empirical risk minimization
FMN	fractional matching number
e.g.	exampli gratia (for example)
Fig.	Figure
HS	hyperedge split
IND	independent set
i.e.	id est (that is)
LP	linear program
MCP	minimum clique partition
MinVar	minimum variance
MIS	maximum independent set
MSC	minimum set cover
MWCWSV	minimum worst-case weighted-sum variance game
NP	nondeterministic polynomial time
OGSM	overlap graph based support measure
OHSM	overlap hypergraph based support measure
PAC	probably approximately correct
SC	subset contraction
SDP	semidefinite program
SGM	Schrijver graph measure
SRL	statistical relational learning
VA	vertex addition

List of Symbols

Symbol	Definition
G	graph or hypergraph
V	vertex set
E	hyperedge set
$V^{(i)}$	i-th partition of V
N_v	neighborhood of v
d_v	degree of v
K_n	complete graph on n vertices
χ^*	fractional hyperedge-chromatic number
ν^*	fractional matching number
ω	maximum degree
$\bar{G}$	complement graph of G
$\mathcal{X}$	feature space of vertices
$\mathcal{X}^{(i)}$	feature space of i-th partition
$\mathcal{X}^{(S)}$	$\times_{i \in S} \mathcal{X}^{(i)}$
$\mathbb{X}$	space of feature vectors of (complete) examples
$\mathcal{Y}$	target value space
ϕ	vertex labeling function
λ	labeling function
ρ	distribution
$\rho_{\mathcal{X}}$	feature distribution of objects
$\rho_{\mathbb{X}}$	distribution over the feature space $\mathbb{X}$
$\rho^{(i)}$	distribution of i-th feature
$\rho_{\mathcal{Y}\|\mathbb{X}}$	conditional target value distribution
ξ_i	G-networked random variable
E_{IND}	maximum matching
w	weight vector
σ^2	variance of ξ_i
σ_T^2	variance component of set T
J^E	overlap index matrix of hyperedge E

ρ	distribution over examples
Z	training examples
$\mathcal{F}$	space of all measurable functions
$\mathcal{H}$	hypothesis space
$\mathcal{Z}$	$\mathbb{X} \times \mathcal{Y}$
f	any function $\mathbb{X} \mapsto \mathcal{Y}$
$L(\cdot, \cdot)$	loss function
$\mathcal{E}(\cdot)$	expected risk (square loss)
$\mathcal{E}_Z(\cdot)$	empirical risk (square loss)
$f_{\rho,\mathcal{H}}$	minimizer of $\mathcal{E}(\cdot)$ in $\mathcal{H}$
$f_{\rho,\mathcal{F}}$	global minimizer of $\mathcal{E}(\cdot)$
$f_{Z,\mathcal{H}}$	minimizer of $\mathcal{E}_Z(\cdot)$ in $\mathcal{H}$
$f_{Z,\mathcal{F}}$	global minimizer of $\mathcal{E}_Z(\cdot)$
$\mathcal{C}(\cdot)$	Banach space of continuous functions
$N(\cdot, \cdot)$	covering number
$\mathcal{E}_S(\cdot)$	sample error
$\mathcal{E}_A(\cdot)$	approximation error

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The topic of this thesis is *graph patterns and networked statistics*, and the goal is to build statistically sound and efficient methods to mine and learn from networked data. This thesis consists of the present introductory part in the first two chapters, other four chapters which delve into different aspects of the topic and a conclusion chapter.

In this chapter, we introduce the topic, describe the motivations of the work, give an overview of the contributions, and list the original articles.

1.1 Thesis topics and research questions

The amount of available data is rapidly increasing everyday. In order to obtain interesting knowledge from data and to improve the performance of machines when learning from data, we need accurate and efficient data mining and machine learning algorithms. An nonnegligible issue in the design of these algorithms is the structure of the data. Informally speaking, the data is usually called *propositional* when the observed data can be stored in a single table and there is no relationship between any pair of two tuples. Traditional mining and learning techniques work well for most cases of propositional data.

However, collected data tends to have more and more complex structures which make most traditional tools lose effectiveness. The goal of this research work is to propose methods to mine and learn from the data with complex structures, in particular, networked data which is usually represented by a graph or a hypergraph.

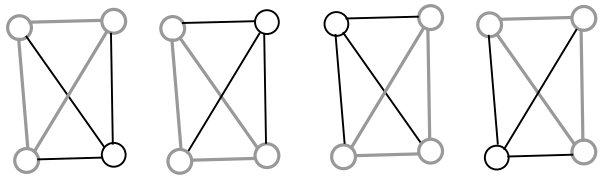


Figure 1.1: Four triangles in the complete graph on four vertices

Many applications produce and exploit linked data which is often called networks, such as social networks, economic networks, citation networks and chemical interaction networks. Several graph mining techniques have been proposed to explore this type of data. For example, vertex clustering methods find communities in a large network [24]; network abstraction techniques transform a large network into a smaller one which is a representative of the original network [76]. Our research starts from another graph mining task: *frequent subgraph pattern mining*, which finds frequent patterns in a large network [41] or a database of many disjoint graphs [74].

Frequent subgraph pattern mining plays an important role in many applications. For example, it helps biologists in analyzing gene regulatory networks to find motifs, which are subgraph patterns that occur much more often than they do in random networks, such as self-regulations and feed forward loops [59].

For frequent pattern mining, an important component is the support measure, which measures how frequently a given pattern occurs in a given database. When the database is transactional, i.e., data can be stored in a single table and there is no dependency relationship between any two data points, one only needs to directly count the occurrences of the pattern. However, occurrences of a subgraph pattern in a large graph may overlap, and then it becomes challenging to define a support measure.

As an example, consider a social network, such as Facebook in which vertices represent individuals and edges represent friendships. Suppose four users A, B, C and D are friends. In this local network of A, B, C and D, what is the support of the triangle pattern? If we do not set a standard, the answer is not determined. The support can be 4, if we follow the same idea as in the transactional case, see Fig. 1.1. It can be 1, if we cannot endure any overlap. It can be 4/3, if we weight every triangle by 1/3 because every vertex is shared by 3 triangles. It even can be some number that is not intuitive at all but one may find a reason (or an excuse) to use this number as the support.

The first main question of this thesis helps to set the standard.

- What are the criteria of a good graph support measure?

The answer to this question suggests that a good graph support measure should inherit some good properties from those for transactional databases, such as anti-monotonicity (i.e., the support of a pattern is always greater than or equal to those of its superpatterns) since it helps graph miners in pruning search spaces.

Given that we are able to tell if a graph support measure is good or not, we face the second question:

- Is there any good graph support measure that is efficiently computable?

To answer this question, we review existing graph support measures but find they are not efficiently computable and then design a new graph measure that can be efficiently computed.

To deeply understand this new support measure, we ask the third question:

- Is there any appealing interpretation of graph support measures?

This question leads us to statistics on networks and statistical learning theory on networked examples.

Learning from structured data becomes one of the most important machine learning topics because the available data shows more and more complex structures. For example, statistical relational learning does not only deal with learning from data points but also from the relationships between them [27]; link prediction tries to find (missing or potential) connections between vertices in a social network [42].

When learning from propositional data, one usually assumes that every data point is independently and identically distributed (i.i.d.). Based on this assumption, many accurate and efficient learning algorithms have been proposed, e.g., support vector machine (SVM) [62], random forests [8], etc. These learning algorithms do not only perform well in practice, but also provide theoretical guarantees. These guarantees show that, some principles such as empirical risk minimization or structural risk minimization [57] make sure that the expected risk of a trained model is close to its empirical risk [17]. However, these results cannot be directly applied to networked data because networked data does not satisfy the i.i.d. assumption as two or more training examples may share some objects and their features [64].

Like the classical learning theory which is based on statistical results on i.i.d. samples, the corresponding learning theory for networked examples needs

foundations in statistics. To lay such a foundation stone, the first question we try to answer is

- What are reasonable assumptions for networked examples?

To set exact goals, we choose two quality measures of statistical estimators on networked examples, variance [68] and concentration bound [15]. For variance, we ask the following question:

- How can we minimize the variance of an unbiased estimator on networked examples?

For concentration bound, we ask a similar question:

- How can we obtain tight concentration bounds for an estimator on networked examples?

Given that we are able to derive tight concentration bounds, we face the last question in this thesis:

- How can these concentration bounds for networked examples guide our learning algorithm design?

This thesis aims to shed light on mining graphs and learning from networked data by answering aforementioned questions. The answers to these questions are discussed in several original articles.

1.2 Contributions

This thesis contributes in different but related aspects to the field of networked data analysis. These are displayed in the following subsections.

Graph support measures (Chapter 3)

In Chapter 3, we propose a graph support measure that has good properties and can be computed efficiently.

In the task of frequent subgraph pattern mining, it is important to properly define the support measure: how frequently a subgraph pattern occurs in a given

large network. An important class of support measures relies on the notion of overlap graphs. A major advantage of overlap graph based approaches is that they combine anti-monotonicity with counting the occurrences of a subgraph pattern which are independent according to certain criteria. However, existing overlap graph based support measures are expensive to compute.

Therefore, we propose a new notion of overlap hypergraphs and show that some parameter of these hypergraphs naturally defines a graph support measure. We study the anti-monotonicity and other properties of this measure. This new measure can be computed by solving a linear program that is known to be efficiently computable. It is experimentally shown that, in contrast to earlier overlap graph based proposals, this support measure makes it feasible to mine subgraph patterns in large networks.

Networked variances (Chapter 4)

In Chapter 4, we propose the problem of learning from networked examples, introduce the concept of networked random variables and analyze variances of weighted mean value estimators on these random variables. This analysis results in a convex quadratically constrained linear program which minimizes the variance of the worst case.

Given a set $\{\xi_i | i = 1, \ldots, n\}$ which is sampled from an unknown distribution, a fundamental problem is to estimate the expected value $\mu = \mathbb{E}[\xi]$. If these random variables are independently distributed, μ is typically approximated by averaging all ξ_i,

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \xi_i$$

because this estimator has the minimal variance. According to the definition of networked random variables, we would not have an independent sample. In such case, we could answer the question what is an effective way to combine the observations to get a good estimate. We will limit ourselves to weighted average estimators

$$\hat{\mu}_w = \frac{\sum_{i=1}^{n} w_i \xi_i}{\sum_{i=1}^{n} w_i}.$$

We exploit Hoeffding decomposition to write the variance of any $\hat{\mu}_w$ as a sum of several variance components. For every weight vector, one can choose possible variance components to maximize the variance which is called the worst-case variance. That is, for every networked sample, the worst-case variance of the corresponding $\hat{\mu}_w$ is a function of the weight vector w. We find an optimal

weight vector w minimizing the variance of the worst case, which is called a Min-Var weighting scheme.

Networked concentration inequalities (Chapter 5)

In Chapter 5, we prove exponential concentration inequalities for weighted mean value estimators on networked random variables. We find good weights based on these inequalities by solving linear programs. These linear programs and the linear programs mentioned in Chapter 3, which are used to compute our graph support measures, are essentially the same.

Consider again the mean value estimation problem. In the i.i.d. case, one can apply classical results, e.g., Chernoff-Hoeffding inequalities [14, 33] to bound the probability $\Pr(\hat{\mu} - \mu \geq \epsilon)$ or $\Pr(|\hat{\mu} - \mu| \geq \epsilon)$ where ϵ is an arbitrary positive number. We derive Chernoff-Hoeffding style concentration inequalities which can be applied to weighted estimators on networked variables. Based on these inequalities, we slightly improve both the Janson inequalities [36] and the concentration bounds for general U-statistics [32, 2]. Comparing with techniques that have been used in the past to prove concentration bounds for unweighted average estimators of networked sample, our proof methods do not decompose the convex sum of networked random variables into several independent parts. From these inequalities, we also obtain a very efficient way to find nearly optimal weights, which is called an FMN (Fractional Matching Number) weighting scheme. An FMN weighting scheme also achieves a small variance.

Learning from networked examples (Chapter 6)

In Chapter 6, we present a statistical learning theory of networked examples.

The main goal of supervised learning is to learn a function $f : \mathbb{X} \mapsto \mathcal{Y}$ from a set of training examples $Z = \{z_i | i = 1, \ldots, n\}$ with $z_i = (x_i, y_i) \in \mathbb{X} \times \mathcal{Y}$, and to predict labels for unseen examples. For an i.i.d. sample, there are many learning approaches, such as the empirical risk minimization principle

$$\min_f \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2,$$

which can lead to good generalization bounds. We generalize the empirical risk minimization principle to the networked case by allowing weighted empirical

risk

$$\sum_{i=1}^{n} w_i (f(x_i) - y_i)^2.$$

Based on inequalities derived in Chapter 5, we present generalization bounds for the networked examples.

1.3 Main publications

1. Christos Pelekis, Jan Ramon, and Yuyi Wang. On the Bernstein-Hoeffding method. *Annales de l'Institut Henri Poincaré (B) Probabilités et Statistiques,* Submitted.

2. Jan Ramon, Constantin Comendant, Mostafa Haghir Chehreghani, and Yuyi Wang. Graph and network pattern mining. In Marie-Francine Moens, Juanzi Li, and Tat-Seng Chua, editors, *Social Media and Social Computing.* CRC Press, 2014. Accepted.

3. Jan Ramon, Yuyi Wang[1], and Zheng-Chu Guo. Learning from networked examples. *Journal of Machine Learning Research*, Accepted.

4. Yuyi Wang and Jan Ramon.An efficiently computable support measure for frequent subgraph pattern mining. In *Proceedings of ECMLPKDD 2012,* pp. 362–377.

5. Yuyi Wang, Jan Ramon, and Thomas Fannes. An efficiently computable subgraph pattern support measure: Counting independent observations. *Data Mining and Knowledge Discovery, 27*(3):444–477, 2013.

[1]The first two authors are listed in lexicographical order.

Chapter 2

Background

In this chapter, we begin by reviewing some classical and simple cases of data mining and machine learning to first convey an intuition of what mining and learning are. These classical cases are extended to and compared to networked cases in Section 2.2. To make this thesis self-contained, we review basic graph theory concepts and notations in Section 2.3.

2.1 Classical cases

Data mining and machine learning is such a broad field that it is impossible to exhaustively list all tasks in this field. We give some tasks that are closely related to the main topics of this thesis.

Frequent itemset mining

One of the most popular and well studied instances of data mining is frequent itemset mining [5], which is also a basic step for several other data mining tasks, e.g., the first step of association rule mining [1] is to find (all) frequent itemsets in a database.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of n binary attributes called items. Let $D = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. An itemset $X \subseteq I$ is frequent if and only if the support of X is not less than

$s_{\min} \times |D|$ where $s_{\min}$ is a threshold specified by the user. Here, the support is the number of occurrences. The task of frequent itemset mining is to find (all) frequent itemsets in D.

To illustrate these concepts, we use a small example from the market basket analysis. The set of items is $I = \{\text{beer}, \text{candy}, \text{diaper}, \text{egg}, \text{milk}\}$ and a small database containing the items (1 and 0 code respectively presence and absence of an item in a transaction) is shown in Table 2.1. If we set $s_{\min} = 0.4$, the frequent itemsets are $\{\text{beer}\}, \{\text{diaper}\}, \{\text{milk}\}$ and $\{\text{beer}, \text{diaper}\}$.

transaction ID	beer	candy	diaper	egg	milk
1	0	1	1	0	1
2	0	0	0	1	1
3	1	0	0	0	1
4	0	0	1	1	1
5	1	0	1	0	0
6	1	0	0	0	1
7	1	0	1	0	0
8	1	1	1	0	0
9	1	1	1	0	1

Table 2.1: Example database with 5 items and 9 transactions

Note that the example in Table 2.1 is extremely small. In practice, datasets often contain thousands or millions of transactions and an itemset needs a support of more than several hundred transactions before it can be considered frequent. Many techniques (see e.g., [5]) have been proposed to accomplish the task of frequent itemset mining for real-world problems. Almost all these techniques are based on the fact that if an itemset is not frequent then its supersets are neither frequent, which is called anti-monotonicity.

Regression and Classification

Regression, in particular linear regression, is probably one of the oldest topics in mathematical statistics dating back to about two hundred years ago [73] when machine learning (or even the modern computer) had not been invented yet.

Given a set of data points $D = \{(x_1, y_1,), (x_2, y_2), \ldots, (x_n, y_n)\}$ where (x_i, y_i) is in $\mathbb{R}^2$, linear regression aims to find a straight line $\hat{f}(x) = ax + b$ that optimally approximates a function $f : \mathbb{R} \mapsto \mathbb{R}$ presumed to be implicit in this dataset, as in Fig. 2.1.

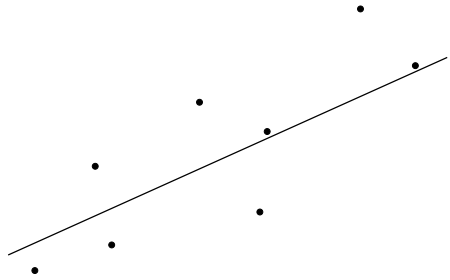


Figure 2.1: Linear regression

One often solves this problem by minimizing the empirical risk with regard to the square loss

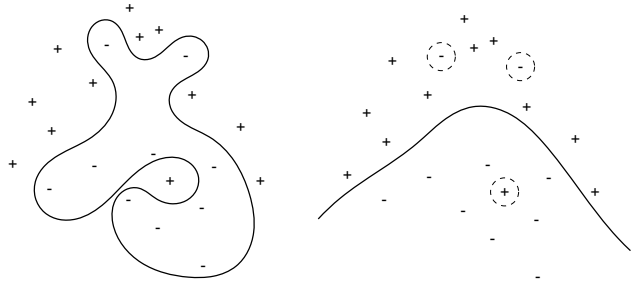
$$L(\hat{f}) = \sum_{i=1}^{n}(y_i - ax_i - b)^2,$$

which leads to the solution

$$a = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad b = \bar{y} - a\bar{x}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$.

Classification is similar to regression but it usually classifies examples into several groups rather than output a real value.



(a) zero training error case (b) non-zero training error case

Figure 2.2: Binary classification

Binary classification is the most common case. The output of binary classification is 0 (for positive examples) or 1 (for negative examples). See Fig. 2.2, given a set of training data points $D = \{(x_1, y_1,), (x_2, y_2), \ldots, (x_n, y_n)\}$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{0, 1\}$, a binary classification algorithm learns a

classification rule $\hat{f} : \mathbb{R}^2 \mapsto \{0, 1\}$. In Fig. 2.2a, we find a rule that is consistent with this data but is complicated. The classification rule in Fig. 2.2b does not classify all training data points correctly but is simpler. Intuitively, one may believe that the classification rule in Fig. 2.2b performs better when these two models are used to predict outputs of new data points, because it is likely that the first classification rule has been overfit. In order to find a good classification rule, one usually minimizes the empirical risk with regard to some loss (e.g., the 0-1 loss) plus an extra term $||\hat{f}||$ that measures the complexity of the rule

$$\sum_{i=1}^{n} I(y_i \neq \hat{f}(x_i)) + ||\hat{f}||$$

where $I(expr)$ equals to 1 if $expr$ is true, 0 if $expr$ is false. This principle, called structural risk minimization, is widely used in classification and regression.

Concentration bounds

The goal of frequent itemset mining is to collect interesting knowledge which can be applied to guide our future actions. Remember the result of the aforementioned market basket analysis example, $\{\text{beer}, \text{diaper}\}$ is a frequent itemset in that database, which reflects that people often buy beer and diaper together so that supermarkets may take advantage of this appearance to increase their profit with promotion. However, a database is only a finite sample, which may be not sufficiently representative for the underlying population. On the one hand, it is possible that an itemset is frequent in the database but infrequent in this population, thus this false positive would mislead our actions. On the other hand, an itemset may be frequent in the population but infrequent in the database, thus we miss some interesting knowledge because of this false negative. Therefore, one might ask:

- Under which conditions the frequency of an itemset in a database is close to the real frequency?

Before answering this question, let us introduce an important assumption that all the transactions in the database drawn from the population are independent and identically distributed (i.i.d.). There is a class of results in statistics stating that, if the sample is i.i.d. and the sample size is large enough, then some estimator will converge to the real value. These results can be used to answer the question above.

For a given itemset X and a database $D = \{t_1, t_2, \ldots, t_m\}$, let $\xi_i = I(X \in t_i)$. The frequency of X in D is

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} \xi_i,$$

and the real frequency is the expected value of $\hat{\mu}$

$$\mu = \mathbb{E}[\hat{\mu}].$$

The question that we need to answer becomes whether the estimator $\hat{\mu}$ is close to μ under the i.i.d. assumption. Several theories were developed to answer this type of questions, e.g., variance analysis, laws of large numbers, exponential inequalities, central limit theorems and large deviations, etc. If different criteria are applied, the answers are various as well.

First, let us consider variance. Suppose the variance of ξ_i is σ^2, then

$$\mathbb{E}\left[(\hat{\mu} - \mu)^2\right] = \frac{\sigma^2}{m}.$$

Next, we consider exponential inequalities, e.g., the one-side and two-side Chernoff-Hoeffding bounds:

- For any $\epsilon > 0$,

$$\Pr\left(\hat{\mu} - \mu \geq \epsilon\right) \leq e^{-2m\epsilon^2}$$

$$\text{and} \quad \Pr\left(|\hat{\mu} - \mu| \geq \epsilon\right) \leq 2e^{-2m\epsilon^2}.$$

All these results demonstrate that, when the sample size $m = |D|$ is large enough, the distance between $\hat{\mu}$ and μ is small, more precisely, the probability that they have a big distance is small.

For regression and classification, models trained from a finite set of examples are used to predict outputs for new data points, so a similar question can be asked:

- Under which conditions learning and generalizing from finite training examples is possible?

Statistical learning theory (SLT) deals with finding such conditions by showing generalization error bounds stating that the learned model would be close

to the optimal one if one has enough i.i.d. training examples[1] and properly chooses the hypothesis space in which the trained model is selected. To derive a generalization error bound, the aforementioned Chernoff-Hoeffding-style bounds play crucial roles.

2.2 Networked cases

In the previous section, we reviewed some classical cases in data mining and machine learning. This section introduces networked correspondences of these tasks.

Frequent subgraph pattern mining

Frequent subgraph pattern mining concerns the problem of looking for subgraph patterns that occur frequently in a collection of graphs or in a single large graph. In this thesis, we mainly consider the single-graph setting, and we call the large graph D containing all data the database graph. Referring to the many applications, such as social networks, the Internet, chemical and biological interaction networks, traffic networks and citation networks, the database graph is also often called the network.

The task of frequent subgraph pattern mining is to find (all) frequent subgraph patterns which satisfy some specified constraints (e.g., connected) in the database graph. Similar to the classical frequent itemset mining, a subgraph pattern is said to be frequent if its support is not less than a threshold specified by the user. To fully accomplish this task, a graph miner needs three main components, though sometimes there are no clear boundaries among them

- A subgraph pattern generator: it generates an initial subgraph pattern and extends this pattern to larger ones. For example, a generator could bring about a pattern as in Fig. 2.3b and extend this small pattern to a larger one as in Fig. 2.3c.
- A matching algorithm: it searches for copies of a subgraph pattern in a database graph. For example, a matching algorithm should be able to list all occurrences of the pattern in Fig. 2.3b (or Fig. 2.3c) in the database graph in Fig. 2.3a.

[1] Actually, in the classical results, testing data points are also assumed to be independent and have the same distribution as training examples.

- A support measure: it counts how frequently a pattern occurs in a database graph (usually after the matching operator finds all the occurrences). For example, a support measure should be able to measure the support of the pattern in Fig. 2.3b (or Fig. 2.3c) in the database graph in Fig. 2.3a.

The first two components, subgraph pattern generator and matching algorithm, are not relevant for the work described in this thesis; the interested reader can find information about them in [51]. The third component, support measure, is trivial in the classical frequent itemset mining; one only needs directly count the occurrences. However, in the networked case, to define a proper support measure becomes complicated. Should we ignore the overlaps between occurrences? If we directly use the number of the occurrences, the pattern in Fig. 2.3b has 4 occurrences in the database graph in Fig. 2.3a (See Fig. 2.3d), while that in Fig. 2.3c has 6 occurrences (See Fig. 2.3e). This violates the property of anti-monotonicity that allows mining algorithms to efficiently prune the search space.

We will discuss this problem in Chapter 3 by explaining an efficiently computable support measure that has the property of anti-monotonicity.

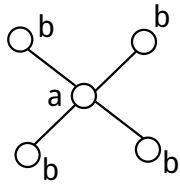
Learning from networked examples

Comparing to classical regression and classification which have been extensively studied, learning from networked examples is a relatively new topic. In networked cases, two or more training examples may share some information.

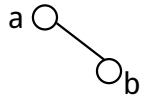
To build a concrete mental image, we consider a simple example of movie rating, which will be refined in following chapters. Every training example in this problem involves two objects, a person and a movie. In Fig. 2.4a, the vertices labeled P_i represent personal features, e.g., age and gender, and the vertices labeled M_i represent features of movies, e.g., length, actor popularity, genre, etc. An edge connecting a person vertex and a movie vertex represents a training example. If one likes the movie, the corresponding edge is labeled 1; otherwise, the edge is labeled 0. Since several edges may share a common person vertex or a common movie vertex, these training examples are not independent.

Suppose every person has a one-dimensional feature (e.g., age) and every movie also has a one-dimensional feature (e.g, length)[2] , then these examples can be plotted in a two-dimensional space as in Fig. 2.4b. In Fig. 2.4a, there are only 6

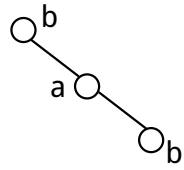
[2]In practice, we usually exploit more features of an object, not only the age of a person and the length of a movie. In order to plot these examples in a two-dimensional space, we simplify this problem by only considering one feature of an object.



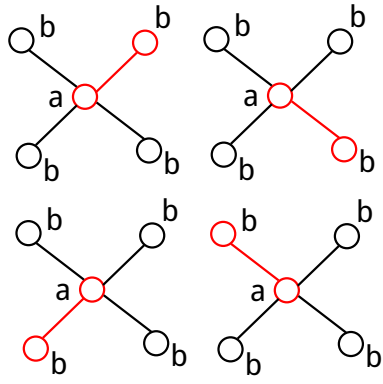
(a) A database graph



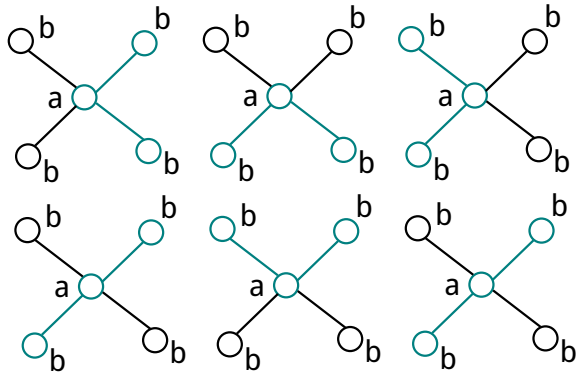
(b) A smaller subgraph pattern



(c) A larger subgraph pattern



(d) The occurrences of the smaller pattern

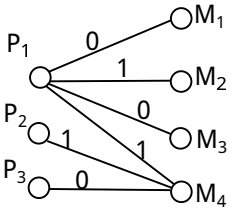


(e) The occurrences of the larger pattern

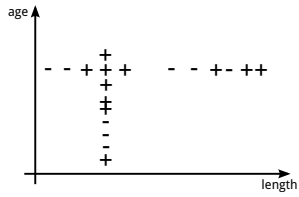
Figure 2.3: Frequent subgraph pattern mining

training examples. If we add more training examples which involve the person P_1 or the movie M_4, in that way these training examples in the two dimensional

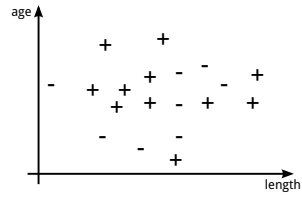
feature space should be similar to Fig. 2.4b. These networked examples lie on two orthogonal lines. If we assume that we have an i.i.d. training sample of the same size, then the corresponding two dimensional image looks like Fig. 2.4c. Intuitively, the i.i.d. examples should be more informative than the networked examples.



(a) Bipartite examples for the movie rating problem. Every example involves the person P_1 or the movie M_4.



(b) A networked training sample. The ratings given by P_1 are in a horizontal line; the ratings for M_4 are in a vertical line.



(c) An i.i.d. set of ratings. This training set has the same sample size as the networked one, but this sample is "more informative".

Figure 2.4: Networked training examples

A challenge of learning from networked examples is to answer the question whether the aforementioned principles, such as empirical risk minimization and structural risk minimization, still work in this setting. We will discuss the empirical risk minimization principle of learning from networked examples in Chapter 6.

Networked statistics

Like the classical cases, for learning from networked data, the following question could be asked

- Under which conditions the model trained from a networked sample is guaranteed to have good generalization ability?

Remember that in the classical cases with the i.i.d. assumption, the answer is related to the quality of some estimator $\hat{\mu}$ of the mean μ. The quality of the estimator can be measured by the variance $\mathbb{E}\left[(\hat{\mu}-\mu)^2\right]$ or the confidence level $\Pr(\hat{\mu}-\mu \geq \epsilon)$. Under the i.i.d. assumption, the variance analysis is straightforward, and the confidence level can be bounded by the classical Chernoff-Hoeffding inequalities.

In networked cases, since we drop the independence assumption, the above classical results cannot be directly applied. In Chapter 4, we will bring in the concept of networked random variables and analyze the variance of estimators defined on these networked random variables. In Chapter 5, we will focus on these estimators by extending Chernoff-Hoeffding style inequalities.

2.3 Basic graph theory concepts

In this section, we provide basic notions and notations of graph theory that are used in the following chapters.

The notion of a graph was introduced by Leonhard Euler in his paper on the Seven Bridges of Königsberg. It did not only solve this specific problem, but also gave necessary and sufficient conditions to find a walk through a general or even an imaginary city (graph), which consists of several islands (vertices), that would cross each bridge (edge) once and only once.

Because graphs naturally capture relations between entities, they are widely used to express structured and semistructured data nowadays.

Graphs

An *undirected graph* G is an ordered pair (V, E) where V is a set of *vertices* $[n] = \{1, \ldots, n\}$ and E is a set of *edges* $E \subseteq \{\{u, v\} | u, v \in V, u \neq v\}$. For example, the set V could be $[4] = \{1, 2, 3, 4\}$, and E could be $\{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{3, 4\}\}$. Graphs can be naturally visualized as in Fig. 2.5.

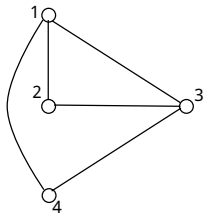


Figure 2.5: An undirected graph

Given two vertices u, v in a graph, if $e = \{u, v\} \in E$, then the two vertices u and v are said to be *adjacent*, and they are also said to be *incident* with e. The

neighborhood of a vertex v, denoted by N_v, is the set of vertices adjacent to v:

$$N_v \equiv \{u \in V | \{u, v\} \in E\}.$$

The cardinality of N_v is called the *degree* of v, denoted by d_v. In Fig. 2.5, vertex 4 has degree 2. The *maximum degree* of a graph G, denoted ω_G, is defined as:

$$\omega_G \equiv \max_{v \in V_G} d_v.$$

The maximum degree of the graph in Fig. 2.5 is 3 because $d_1 = 3$ and no vertex has a larger degree.

Replacing the edge set with a set of *arcs* (ordered pairs of vertices) $E \subseteq \{(u, v) | u, v \in V, u \neq v\}$, we obtain a *directed graph* (Fig. 2.6).

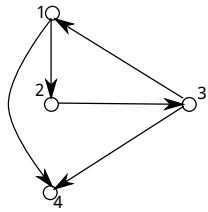


Figure 2.6: A directed graph

By $\mathcal{G}$, we denote the class of all graphs; by $\mathcal{G}^{\leftrightarrow}$ ($\mathcal{G}^{\rightarrow}$), the restriction to undirected (directed) graphs. For a graph G, we use V_G and E_G to denote its vertex set and edge set respectively.

Actually, the graph (Fig. 2.7) that Euler used to describe the Seven Bridges of Königsberg requires E to be a multiset which allows repeated elements. Such graphs are called *multigraphs*. Besides, graphs with self-loops and/or infinite vertices and edges are also used in many problems. In this thesis we focus on finite, simple graphs: those without self-loops or multiple edges.

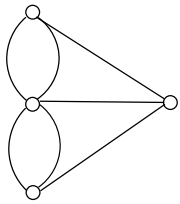


Figure 2.7: The graph corresponding to the Seven Bridges of Königsberg

By assigning labels to vertices and/or edges, we obtain labeled graphs $G = (V, E, \Sigma, \lambda)$ where Σ is an alphabet and $\lambda : V \cup E \mapsto \Sigma$ is a labeling function.

For example, V is a set of persons, edges in E are relationships between these persons, Σ might be $\{m, f, friends, couples\}$ in which m and f are assigned to vertices to represent their genders and $friends$ and $couples$ are assigned to edges to illustrate their relationships (Fig. 2.8). Note that an unlabeled graph can be also considered as a labeled graph in which all vertices and all edges are labeled with the same symbol.

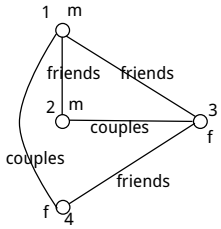


Figure 2.8: A labeled graph

For a labeled graph G, we use λ_G to stand for the labeling function. By $\mathcal{G}_\lambda$ ($\mathcal{G}_\bullet$), we denote the class of labeled (unlabeled) graphs. In this thesis, we combine notations, e.g., $\mathcal{G}_\bullet^{\rightarrow}$ for directed, unlabeled graphs.

A graph g is said to be a *subgraph* of G if and only if $V_g \subseteq V_G$ and $E_g \subseteq E_G$, and write $g \subseteq G$. Fig. 2.9 shows a subgraph of Fig. 2.5.

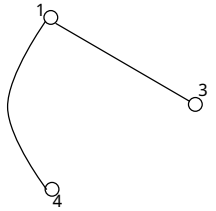


Figure 2.9: A subgraph of Fig. 2.5

For $G \in \mathcal{G}^{\leftrightarrow}$, $\overline{G} \equiv (V_G, V_G^2 \setminus (E_G \cup \{\{v, v\} | v \in V_G\}))$ denotes the *complement graph* of G. Fig. 2.10 shows the complement graph of Fig. 2.5.

By $K_n \in \mathcal{G}^{\leftrightarrow}$ we denote the *complete graph* on n vertices (Fig. 2.11), i.e., $K_n \equiv ([n], \{\{u, v\} | 1 \leq u < v \leq n\})$.

For an undirected graph G, if there exist two disjoint sets $V^{(1)}$ and $V^{(2)}$ such that $V_G = V^{(1)} \cup V^{(2)}$ and $E_G \subseteq \{\{u, v\} | u \in V^{(1)}, v \in V^{(2)}\}$, then this graph is a *bipartite graph* (Fig. 2.12).

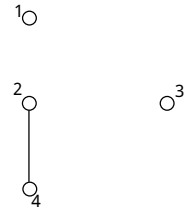


Figure 2.10: A complement graph of Fig. 2.5

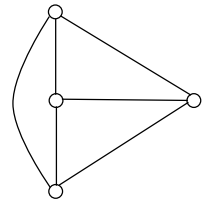


Figure 2.11: A complete graph on 4 vertices

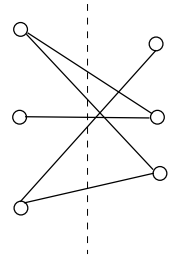


Figure 2.12: A bipartite graph

Morphisms

The following concepts defined in terms of $\mathcal{G}_\lambda^{\rightarrow}$ are also valid for undirected and/or unlabeled graphs by dropping the direction of the edges and/or the labels.

A *homomorphism* ψ from $G \in \mathcal{G}_\lambda^{\rightarrow}$ to $G' \in \mathcal{G}_\lambda^{\rightarrow}$ is a mapping from V_G to $V_{G'}$ such that $\forall (u,v) \in E_G : (\psi(v), \psi(w)) \in E_{G'}$. We say that G is *homomorphic* to G' (Fig. 2.13).

If $\forall v \in V_G : \lambda_G(v) = \lambda_{G'}(\psi(v))$, we call ψ *label-preserving*. We always implicitly assume that ψ is label-preserving when $G, G' \in \mathcal{G}_\lambda$.

A homomorphism ψ is called *vertex-surjective* if $\forall v' \in V_{G'} : \exists v \in V_G : \psi(v) = v'$.

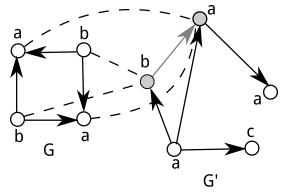


Figure 2.13: A homomorphism from G to G' shown by dashed lines: the vertices with label a (b) are mapped to a vertex with label a (b). The non-shaded vertices of G' are not involved in this homomorphism.

We call ψ *edge-surjective* if $\forall(u', v') \in E_{G'} : \exists(u, v) \in E_G : \psi(u) = u' \wedge \psi(v) = v'$ and call it *surjective* if it is both vertex- and edge-surjective (Fig. 2.14).

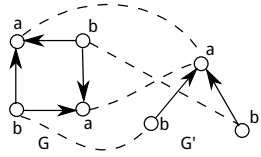


Figure 2.14: A surjective homomorphism from G to G' shown by dashed lines.

An *isomorphism* from G to G' is a bijective homomorphism ψ from G to G', i.e., the homomorphism ψ is surjective and for all $u, v \in V_G, v \neq u$ implies $\psi(v) \neq \psi(u)$. In that case, we say that G is *isomorphic* to G' and write $G \cong G'$ (Fig. 2.15).

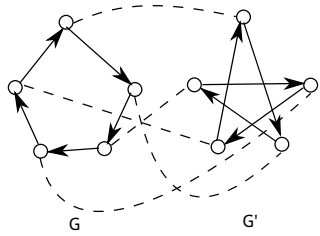


Figure 2.15: An isomorphism from G to G' shown by dashed lines.

We use $G \preceq G'$ to denote that $G \cong g$, for some subgraph g of G'. This is the same as saying that there exists a *subgraph isomorphism* from G to G' (Fig. 2.16).

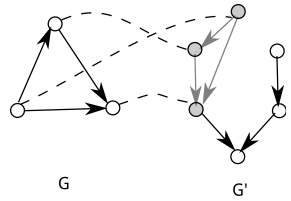


Figure 2.16: A subgraph isomorphism from G to G': the shaded part of G' is isomorphic to G.

Hypergraphs

Allowing edges to be arbitrary subsets (instead of only pairs) of vertices, i.e., $E \subseteq 2^V$, gives us *hypergraphs* (Fig. 2.17), and these edges are called *hyperedges*. The class of all hypergraphs is denoted by $\mathcal{H}$.

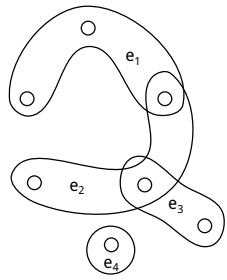


Figure 2.17: A hypergraph

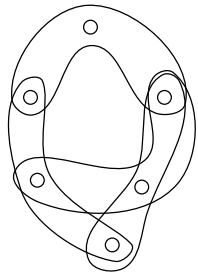


Figure 2.18: A 3-uniform hypergraph

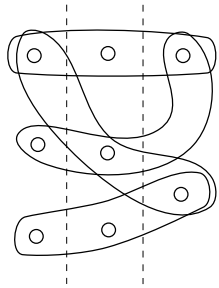


Figure 2.19: A 3-partite hypergraph

For a hypergraph $H \in \mathcal{H}$, if each edge in E_H has a same cardinality k, then this hypergraph is called a *k-uniform hypergraph* (Fig. 2.18). It is noticed that simple undirected graphs are 2-uniform hypergraphs.

For a k-uniform hypergraph H, if $V_H = \bigcup_{i=1}^{k} V^{(i)}$ where $V^{(i)}$ and $V^{(j)}$ are pairwise disjoint for all different i and j, and $E_H \subseteq \{\{v^{(1)}, \ldots, v^{(k)}\} | v^{(i)} \in V^{(i)}, i = 1, \ldots, k\}$, then this hypergraph is called a *k-partite hypergraph* (Fig. 2.19).

Chapter 3

Graph support measures

Graph mining is a subfield of structured data mining. An important task in graph mining is *frequent subgraph pattern mining*, which concerns the problem of finding subgraph patterns that occur frequently in a collection of graphs or in a single large graph. In this chapter, we consider the single-graph setting, and we will call the large graph containing all data the *database graph*.

In order to precisely define a frequent subgraph pattern mining problem, a *support measure* (also called a *frequency measure*) is needed. In the problem setting where patterns are mined in a set of transactions (e.g., itemset mining [1]), a simple support measure is to count the number of transactions in which the subgraph pattern occurs. However, in the context of a single large graph, the issue is less straightforward as several articles have demonstrated [9, 10, 22, 65].

A drawback of just using the number of occurrences of a subgraph pattern (either embeddings or images, see Definition 3.1) as its support is that this support does not have the important property of *anti-monotonicity*. An important class of anti-monotonic support measures relies on *overlap graphs*. However, existing overlap graph based support measures (OGSM) are expensive to compute.

In this chapter, we propose a new support measure ν^*[1] that is based on bounding the value of all occurrences of a subgraph pattern that share a particular part of the database graph, and we show that this support measure can be computed efficiently using a linear program (LP). The measure ν^* is not a traditional OGSM, because its output does not merely depend on the overlap graph. We introduce the notion of an *overlap hypergraph*, and represent ν^* as an overlap hypergraph based support measure (OHSM). We prove that ν^* is anti-monotonic

[1] In the original paper [70], this support measure was denoted as s.

and normalized. Furthermore, we show that all normalized anti-monotonic OHSMs are bounded between two extreme support measures. Our empirical analysis shows that this approach yields the first support measure which is both overlap based and computationally feasible.

3.1 Support measures

In this section, we review the concepts and properties of support measures and overlap graphs, and present a new overlap graph based support measure which is similar to the Lovász ϑ value but closer to the Maximum Independent Set (MIS) support measure. As this new measure is mainly of theoretical interest, it is not further explored in this chapter. In Section 3.3 we introduce the ν^* measure and show that it can be computed using algorithms with much lower asymptotic complexity.

Basic concepts and properties

We first review a number of concepts related to the support of a pattern.

Definition 3.1 (Image and embedding)**.** An *iso-image* (*homo-image*) g of a subgraph pattern $P \in \mathcal{G}$ in a database graph $D \in \mathcal{G}$ is a subgraph $g \subseteq D$ for which there exists an isomorphism (surjective homomorphism) ψ from P to g. The subgraph g is called the iso-image (homo-image) through ψ. An individual isomorphism (surjective homomorphism) ψ from P to g is called an *iso-embedding* (*homo-embedding*) of P in D.

In this chapter, we only consider iso-images, although the measure ν^* can be generalized for other matching operators such as homomorphism. We subsequently use the term *image* instead of iso-image, and denote by $\mathrm{Img}(D, P)$ the set of all images of P in D. We denote by $\mathrm{Emb}(D, P)$ the set of all embeddings of P in D. Given two patterns P' and P, two images $g \in \mathrm{Img}(D, P)$ and $g' \in \mathrm{Img}(D, P')$, if g is a subgraph of g', then we call g a *subimage* of g' and g' a *superimage* of g.

Definition 3.2 (Support)**.** A *support measure* is a function $f : \mathcal{G} \times \mathcal{G} \mapsto \mathbb{R}$ that maps pairs (D, P) to a non-negative number $f(D, P)$, where P is called the subgraph pattern, D the database graph and $f(D, P)$ the *support* of P in D.

For efficiency reasons, most graph miners generate subgraph patterns from smaller subgraph patterns to larger ones [11], exploiting the anti-monotonicity property of their support measure to prune the search space [41].

Definition 3.3 (Anti-monotonicity)**.** A support measure f is *anti-monotonic* if for all P, P', D in $\mathcal{G} : P \subseteq P' \Rightarrow f(D, P') \leq f(D, P)$.

However, anti-monotonicity is not sufficient. For example, a support measure that just returns a constant is anti-monotonic, but not informative.

The support measure should also be able to account for the independence of the occurrences of the subgraph patterns. *Overlap* can be defined in different ways (see [10] and later in this chapter Section 3.4). Popular definitions are, for instance as *vertex-overlap*, i.e., two images g_1 and g_2 overlap if $V_{g_1} \cap V_{g_2} \neq \emptyset$, and as *edge-overlap*, i.e., two images g_1 and g_2 overlap if $E_{g_1} \cap E_{g_2} \neq \emptyset$. Edge-overlap implies vertex-overlap. In this chapter, we use 'overlap' to mean vertex-overlap, although our results are also applicable in the edge-overlap setting.

While one can argue about the value a support measure should have for patterns with overlapping embeddings, things are clearer when embeddings do not overlap. Hence, we will use non-overlapping cases as reference points.

From a statistical point of view, the more independent examples are, the more valuable this set of examples. Calders et al. [10] proposed using a situation in which images (or embeddings) of a subgraph pattern occur independently (i.e., they do not overlap according to some notion of overlap) as a reference. In particular, the notion of a normalized graph support measure was defined: a support measure is *normalized* if every subgraph pattern which only has non-overlapping occurrences in a database graph has a support in this database graph that equals the number of occurrences.

Definition 3.4 (Normalized support)**.** A support measure f is *normalized* if for all P, D in $\mathcal{G} : f(D, P) = |\text{Img}(D, P)|$ when there do not exist two distinct images g_1 and g_2 in $\text{Img}(D, P)$ satisfying $V_{g_1} \cap V_{g_2} \neq \emptyset$.

Overlap graphs

The notion of an overlap graph plays an important role in the design and computation of anti-monotonic support measures in networks.

Definition 3.5 (Overlap graph)**.** Given a subgraph pattern P and a database graph D, the *overlap graph* of P in D is an unlabeled undirected graph G_P^D. Every vertex of G_P^D is an image of P in D, that is, $V_{G_P^D} = \text{Img}(D, P)$. Two distinct vertices u and v are adjacent in G_P^D if they overlap, i.e., if we use vertex-overlap, $V_u \cap V_v \neq \emptyset$. Though an image always overlaps with itself, we assume that overlap graphs do not contain any self-loop.

An overlap graph therefore indicates how often a subgraph pattern occurs in the database graph, and how independent these occurrences are. Vanetik et al. [66] define the induced support measure $f(D, P) = f'(G_P^D)$ where f' is a measure on overlap graphs. We call the induced support measure an overlap graph based support measure (OGSM). A major advantage of overlap graph based approaches is that they combine anti-monotonicity with counting the occurrences of a subgraph pattern which are independent according to certain criteria.

Existing overlap graph based support measures

The first normalized anti-monotonic OGSM which has been proposed was the size of the maximum independent set (MIS) of the overlap graph [65]. To precisely define these supports, we first present the concept of independent sets, cliques and clique partitions.

Definition 3.6 (Independent set)**.** Given a graph $G \in \mathcal{G}^{\leftrightarrow}$, a subset $I \subseteq V_G$ is an *independent set* if for every pair of vertices $u, v \in I$, $\{u, v\} \notin E_G$.

Definition 3.7 (Clique)**.** Given a graph $G \in \mathcal{G}^{\leftrightarrow}$, a subset $C \subseteq V_G$ is a *clique* if for every pair of vertices $u, v \in C$, $\{u, v\} \in E_G$.

Definition 3.8 (Clique partition)**.** Given a graph $G \in \mathcal{G}^{\leftrightarrow}$, a *clique partition* of G is a partition of the vertices of G into cliques.

Definition 3.9 (MIS support)**.** Given a database graph D and a subgraph pattern P the *MIS support* of P in D is

$$MIS(G_P^D) = \max\{|I| \mid I \text{ is an independent set of } G_P^D\}.$$

This MIS support is intuitively appealing since it measures how often we observe a subgraph pattern occurring independently. Unfortunately, computing the MIS of an overlap graph is NP-hard [26]. Moreover, it has been shown that MIS cannot be approximated even within a factor of $n^{1-o(1)}$, where n is the number of vertices of the overlap graph, in polynomial time, unless P=NP [21, 30, 19].

Later, Calders et al. [10] proposed two normalized anti-monotonic OGSMs, the size of a minimum clique partition (MCP) of the overlap graph and the Lovász theta value (ϑ) of the overlap graph.

Definition 3.10 (MCP support)**.** Given a database graph D and a subgraph pattern P the *MCP support* of P in D is

$$MCP(G_P^D) = \min\{|Q| \mid Q \text{ is a clique partition of } G_P^D\}.$$

The Lovász ϑ function is a well-known function sandwiched between MIS and MCP which can be computed in polynomial time. The concept of Lovász feasible matrix is used in the definition of the Lovász ϑ support measure.

Definition 3.11 (Lovász feasible matrix)**.** Given an overlap graph G_P^D, a *Lovász feasible matrix* A for G_P^D is a symmetric positive semidefinite matrix with (i) $A_{u,v} = 0$ for all u and v such that $\{u, v\} \in E_{G_P^D}$ and (ii) $Tr(A) = 1$ where $Tr(.)$ is the sum of the elements on the main diagonal (the trace).

Definition 3.12 (Lovász ϑ support measure)**.** Given a database graph D and a subgraph pattern P the *Lovász ϑ support measure* of P in D is

$$\vartheta(G_P^D) = \max \left\{ \sum_{i,j} A_{i,j} \mid A \text{ is a Lovász feasible matrix of } G_P^D \right\}.$$

These OGSMs are also very expensive to compute. In particular, MCP is NP-hard to compute while the Lovász ϑ value is the solution of a semidefinite program (SDP) which takes $O(|V_{G_P^D}|^{6.5})$ to solve for a general graph. An SDP primal-dual algorithm for approximating ϑ with a multiplicative error of $(1 + \epsilon)$ has been proposed [12]; it has a running time of $O(\epsilon^{-2} n^5 \log n)$. Iyrngar et al. [34] considered subgradient methods for approximating ϑ; they run in time $O(\epsilon^{-2} \log^3(\epsilon^{-1}) n^4 \log n)$ in the worst case. Unfortunately, even these approximation methods are still computationally too expensive for our purposes.

Operations on overlap graphs

Vanetik et al. [66] provide a way to prove the anti-monotonicity of OGSMs. The result is based on three operations defined on overlap graphs.

For any graph $G = (V, E) \in \mathcal{G}^{\leftrightarrow}$, we define

- Vertex Addition (VA): for a new vertex v, $VA(G, v) = (V \cup \{v\}, E \cup \{\{v, u\} | u \in V\})$.
- Edge Removal (ER): for a given edge $e \in E$, $ER(G, e) = (V, E \setminus \{e\})$.
- Clique Contraction (CC): for a given clique $K \subseteq V$ and a new vertex k, $CC(G, K, k) = (V \setminus K \cup \{k\}, E \setminus \{e | e \cap K \neq \emptyset\} \cup \{\{k, v\} | \forall u \in K : \{v, u\} \in E\})$.

Theorem 3.13 (Vanetik et al. [66])**.** *An overlap graph based support is anti-monotonic if and only if the corresponding overlap graph measure does not decrease when we perform VA, ER and CC on overlap graphs.*

The anti-monotonicity of OGSMs mentioned in Section 3.1 can be proved by this theorem. We extend these results to overlap hypergraphs (see Section 3.3).

The Schrijver graph measure as a support measure

The material in this section is not essential for the sequel of this chapter and can be easily skipped by readers eager to get to the main contribution.

The first function which was shown to be a normalized anti-monotonic OGSM computable in polynomial time was the Lovász ϑ value of the overlap graph. In the graph theory literature, many other measures on graphs are studied which could be of interest from the point of view of support measures. Often, the literature also shows relations to the size of the maximum independent set and other important measures. We believe that it may be valuable for the data mining community to further explore this literature. As an example, we point out that the Schrijver graph measure [56], which is similar to the Lovász ϑ value, can also be interpreted as a normalized anti-monotonic OGSM. The Schrijver graph measure has nearly the same computational complexity as the Lovász ϑ value, and it is closer to the MIS support measure. The latter can be an advantage for certain statistical tasks in which we want to stay as close as possible to an independent set of images.

The Schrijver graph measure is defined on a Lovász feasible matrix with nonnegative elements.

Definition 3.14 (Schrijver feasible matrix)**.** Given an overlap graph G_P^D, a *Schrijver feasible matrix* A for G_P^D is a Lovász feasible matrix which is nonnegative, i.e., for all i and j, $A_{i,j} \geq 0$.

Given a graph G, we denote by $\mathsf{sfm}(G)$ the set of all Schrijver feasible matrices of G.

Definition 3.15 (Schrijver graph support measure)**.** Given a database graph D and a subgraph pattern P the *Schrijver graph measure (SGM)* support of P in D is

$$SGM(G_P^D) = \max\left\{ \sum_{i,j} A_{i,j} \mid A \in \mathsf{sfm}(G_P^D) \right\}.$$

From the definition, we can easily see that for any overlap graph G_P^D, it holds that

$$SGM(G_P^D) \leq \vartheta(G_P^D) \tag{3.1}$$

since any Schrijver feasible matrix is a Lovász feasible matrix. In addition, the Schrijver graph measure can also be modeled as an SDP which has similar variables and constraints as the Lovász ϑ value, so it is almost equally expensive to compute the Schrijver graph measure of an overlap graph as to compute its Lovász ϑ value.

Now, we prove the following result using Theorem 3.13:

Theorem 3.16. *The SGM support measure is a normalized anti-monotonic OGSM.*

Proof. First, we prove SGM is normalized. This is equivalent to proving that $SGM(\overline{K_n}) = n$ where $\overline{K_n}$ is a graph with n isolated vertices. According to Inequality (3.1), we have $SGM(\overline{K_n}) \leq \vartheta(\overline{K_n})$. It is known that for any graph G, $\vartheta(G) \leq MCP(G)$ (see e.g., [40]). Therefore, $SGM(\overline{K_n}) \leq n$.

Now we show that n is attainable. The matrix whose every element $1/n$ is a Schrijver feasible matrix of $\overline{K_n}$. Then, $SGM(\overline{K_n}) = n$.

Next, we prove that SGM is anti-monotonic using the conditions of Theorem 3.13, i.e., the SGM support measure is non-decreasing under the three operations VA, ER and CC.

- Let $G' = VA(G, v)$ and $A \in \mathsf{sfm}(G_P^D)$ such that $SGM(G_P^D) = \sum_{i,j} A_{i,j}$. The matrix $A' = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$ is a Schrijver feasible matrix of G'. Then, $SGM(G') \geq \sum_{i,j} A'_{i,j} = \sum_{i,j} A_{i,j} = SGM(G)$.
- Let $G' = ER(G, e)$, and $A \in \mathsf{sfm}(G_P^D)$ such that $SGM(G_P^D) = \sum_{i,j} A_{i,j}$. The matrix $A' = A$ is a Schrijver feasible matrix of G'. Then, $SGM(G') \geq \sum_{i,j} A'_{i,j} = \sum_{i,j} A_{i,j} = SGM(G)$.
- Let $G' = CC(G, K, k)$, and $A \in \mathsf{sfm}(G_P^D)$ such that $SGM(G_P^D) = \sum_{i,j} A_{i,j}$. Without loss of generality, we assume that the vertices in K are the last vertices (i.e., they correspond to the bottom rows and rightmost columns). Suppose $|V_G| = n$, for any real vector $x = (x_1, \ldots, x_n)$, $x^T A x \geq 0$ because A is positive semidefinite. Let A' be a symmetric matrix of size $(n - |K| + 1) \times (n - |K| + 1)$ that $A'_{i,j} = A_{i,j}$ for every $1 \leq i, j \leq n - |K|$, $A'_{n-|K|+1,j} = \sum_{n-|K|+1 \leq i \leq n} A_{i,j}$ for every $1 \leq j \leq n - |K|$, $A'_{i,n-|K|+1} = \sum_{n-|K|+1 \leq j \leq n} A_{i,j}$ for every $1 \leq i \leq n - |K|$, and $A'_{n-|K|+1,n-|K|+1} = \sum_{n-|K|+1 \leq i \leq n} A_{i,i}$. For every real vector $y = (y_1, \ldots, y_{n-|K|+1})$, there is a vector x, $x_i = y_i$ for $1 \leq i \leq n - |K|$ and $x_i = y_{n-|K|+1}$ for $n - |K| \leq i \leq n$, such that

$y^T A'y = x^T Ax \geq 0$. Therefore, A' is a Schrijver feasible matrix of G' and $SGM(G') \geq \sum_{i,j} A'_{i,j} = \sum_{i,j} A_{i,j} = SGM(G)$.

□

According to Inequality (3.1) and the bounding theorem in [10], the following inequalities hold for any overlap graph G_P^D:

$$MIS(G_P^D) \leq SGM(G_P^D) \leq \vartheta(G_P^D) \leq MCP(G_P^D).$$

3.2 Related work

Besides the existing overlap graph based support measures described in the previous section, several previous studies have explored the support of a subgraph pattern in a database graph. In this section, we review a widely used support measure called the min-image based measure as well as another definition of overlap, which ignores those overlaps that do not harm the anti-monotonicity of the MIS support. The definitions given in this section are not needed for the main results in this chapter (Section 3.3).

Min-image based support

In [9], the authors proposed an anti-monotonic support measure named *min-image based support*.

$$minImage(D, P) = \min_{v \in V_P} |\{\psi(v) \mid \psi \text{ is a subgraph isomorphism from } P \text{ to } D\}| \tag{3.2}$$

This support counts the minimum number of images of subgraph pattern vertices. Although the anti-monotonicity of this support is obvious, and it can be computed very efficiently, it has several drawbacks.

For example, from a statistical point of view, *minImage* overestimates the evidence. In particular, as Fig. 3.1 shows, a vertex can be counted arbitrarily many times.

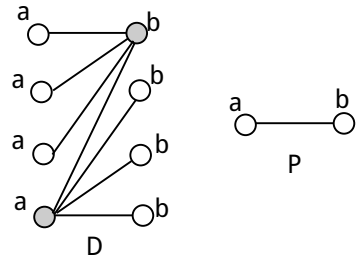


Figure 3.1: Database graph D contains two independent images of the subgraph pattern P. However, $minImage(D, P) = 4$ (and we can make this value arbitrarily large by adding more vertices with label b (resp. a) and link them to the bottom-left vertex with label a (resp. top-right vertex with label b). As a consequence, if we remove just a single vertex (the bottom-left or top-right one) the support of the pattern in the network can suddenly drop to one.

Harmful overlap

Fiedler and Borgelt [22] observed that, when using edge-overlap, some of the overlaps can be disregarded without harming the anti-monotonicity of the MIS support measure. The authors introduce the notion of *harmful overlap support* which relies on the nonexistence of equivalent ancestor embeddings.

Definition 3.17 (Harmful overlap)**.** Given a database graph D and a subgraph pattern P, there is harmful overlap between two embeddings ψ_1 and ψ_2 of P in D if $\exists v \in V_P : \psi_1(v), \psi_2(v) \in V_{g_1} \cap V_{g_2}$, where g_1 (resp. g_2) is the image of P in D through ψ_1 (resp. ψ_2).

The notion of harmful overlap makes sense, especially in situations where objects (vertices) in the database graph play different roles in different embeddings, i.e., if one vertex in the database graph participates in two embeddings as images of two different vertices of the pattern, this is not considered as a dependency between the two embeddings.

3.3 The ν^* support measure

In this section we introduce our new support measure ν^*. We start by introducing overlap hypergraphs, which are overlap graphs carrying additional information on the cause of the overlap.

Overlap hypergraphs

As we are using vertex-overlap, each vertex v in a database graph D determines a clique in the overlap graph G_P^D in which P is a subgraph pattern. That is, suppose v is a vertex in D, then $\text{Img}_v(D,P) = \{g \in \text{Img}(D,P) \mid v \in V_g\}$ induces a clique in G_P^D since these images overlap at the vertex v.

Definition 3.18 (Overlap hypergraph)**.** Given a subgraph pattern P and a database graph D, the *overlap hypergraph* of P in D is a hypergraph H_P^D. Every vertex of H_P^D is an image of P in D, that is, $V_{H_P^D} = \text{Img}(D,P)$. The hyperedge set of H_P^D is $E_{H_P^D} = \{\text{Img}_v \mid v \in V_D\}$.

In an overlap hypergraph H_P^D, we say that a hyperedge e is *dominated* by another hyperedge e' if $e \subset e'$, and a hyperedge e is *dominating* if it is not dominated by any other hyperedge. For any D and P, we define the reduced overlap hypergraph $\tilde{H}_P^D$ to be the hypergraph for which $V_{\tilde{H}_P^D} = V_{H_P^D}$ and $E_{\tilde{H}_P^D}$ is the set of all dominating hyperedges of H_P^D. In the sequel we only refer to $\tilde{H}_P^D$. We will abuse terminology and simply call $\tilde{H}_P^D$ the overlap hypergraph. See Fig. 3.2 for an example.

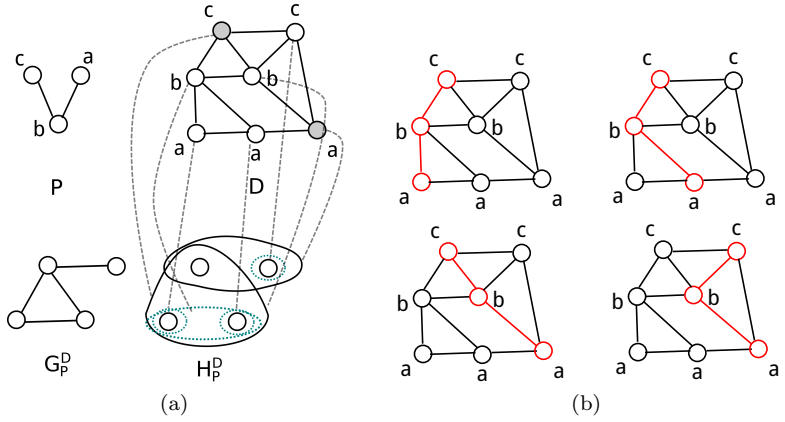


Figure 3.2: Overlap graphs and overlap hypergraphs. Figures in (a) show a subgraph pattern P (top left), a database graph D (top right), the overlap graph G_P^D (bottom left) and the overlap hypergraph H_P^D (bottom right). The two shaded vertices in D determine two dominating hyperedges (solid curves) in H_P^D; other vertices in D determine several dominated hyperedges (blue dotted curves) in H_P^D. Figures in (b) show all images of the pattern P in the database graph D.

We henceforth refer to the overlap hypergraph measures, which we denote by $f'(\tilde{H}_P^D)$, instead of referring to the induced support measure $f(D, P)$. Such induced support measures are called overlap hypergraph based support measures (OHSM). We call OHSMs and OGSMs overlap based support measures.

Origin and definition

Given an overlap hypergraph $\tilde{H}_P^D$, we can derive the corresponding overlap graph G_P^D by replacing every hyperedge with a clique. Therefore, we can rephrase the definition of the MIS measure using overlap hypergraphs. Suppose D is a database graph and P is a subgraph pattern:

$$MIS(D, P) = MIS(\tilde{H}_P^D) = \max |\{I \subseteq V_{\tilde{H}_P^D} \mid \forall e \in E_{\tilde{H}_p^D} : |e \cap I| \leq 1\}|, \quad (3.3)$$

i.e., the MIS measure is the maximum number of vertices, any two of which do not belong to one hyperedge. The MIS measure requires that a vertex of an overlap (hyper)graph is either in the independent set I or not, so it is the solution to a binary integer program. Our new measure ν^* is a linear relaxation of the MIS support measure by allowing for counting vertices of an overlap hypergraph only partially.

Let $\tilde{H}_P^D$ be an overlap hypergraph. We start by assigning to each vertex v of $\tilde{H}_P^D$ a variable x_v. We then consider vectors $x \in \mathbb{R}^{|V_{\tilde{H}_P^D}|}$ of variables where for every $v \in V_{\tilde{H}_P^D}$, x_v denotes the variable (component of x) corresponding to v.

Definition 3.19 (ν^* support measure)**.** The measure ν^* is defined by

$$\nu^*(\tilde{H}_P^D) = \max_{x \in \mathfrak{R}(\tilde{H}_P^D)} \sum_{v \in V_{\tilde{H}_P^D}} x_v \quad (3.4)$$

where $\mathfrak{R}(\tilde{H}_P^D)$ is the feasible region of x, and x is *feasible* if and only if it satisfies

$$\begin{array}{ll} \text{(i)} & \forall v \in V_{\tilde{H}_P^D} : 0 \leq x_v \\ \text{(ii)} & \forall e \in E_{\tilde{H}_p^D} : \sum_{v \in e} x_v \leq 1. \end{array} \quad (3.5)$$

Clearly, ν^* is the solution to a linear program.

We will call an element $x \in \mathfrak{R}(\tilde{H}_P^D)$ which makes $\sum_{v \in V_{\tilde{H}_P^D}} x_v$ maximal a *solution* to the LP of ν^*.

There are very effective methods for solving LPs, including the simplex method which is efficient in practice although its complexity is exponential, and the

more recent interior-point methods [6]. The interior-point method solves an LP in $O(n^2m)$ time, where n (here $\min\{|V_{\tilde{H}_P^D}|, |E_{\tilde{H}_P^D}|\}$) is the number of variables, and m (here $|V_{\tilde{H}_P^D}| + |E_{\tilde{H}_P^D}|$) is the number of constraints. Usually, subgraph patterns are not large, so the LPs for computing ν^* are sparse. Almost all LP solvers perform significantly better for sparse LPs.

An intuitive interpretation

So far, the ν^* support has been explained as a relaxation of the MIS support. In more detail, the basic ideas underlying the ν^* measure are the following:

1. *x_v is the contribution of the image v, and $x_v \geq 0$ for all v.* By finding an additional image (corresponding to a 'Vertex Addition' in the overlap (hyper)graph), it should positively contribute to the total support.

2. *The ν^* support measure is the sum of contributions of all images.* This idea addresses the question of how to use the contributions to define the total support. We can simply sum the contributions of all images, which is intuitive and is one of the main factors in the nice mathematical properties of ν^* including its additivity (the ν^* of a graph is the sum of the ν^* of its connected components).

3. *Every image should contribute as much as possible (but not more than 1).* This principle implies that ν^* is obtained through a process of maximization. In fact, in this maximization process we identify a set of images of the pattern which is most interesting / informative.

4. *All images which share a common vertex cannot contribute more than 1 in total, i.e., the sum of contributions of images in a hyperedge ≤ 1.* If only the first three principles applied and if no additional constraints were imposed on the contributions of the individual images, this would yield a trivial support which just counts the number of images. As pointed out earlier, this support is not anti-monotonic because overlapping images are counted independently. When discussing the *minImage* support measure, we also pointed out that, ideally, any vertex in a database graph cannot contribute more than 1 to the support measure. This is equivalent to saying that, all images sharing a common vertex cannot jointly contribute more than 1 in total.

The ν^* measure therefore has several uses. It is statistically meaningful in its own right (see Chapter 5), and as a relaxation of the maximal independent set, it can give a quick approximation of the MIS value. Still, if overlap needs

to be avoided at all cost (such as in [18]), the maximum independent set size will need to be computed or approximated from below if that is an acceptable alternative.

Conditions for anti-monotonicity

Vanetik et al. [66] described necessary and sufficient conditions for anti-monotonicity of OGSMs on labeled graph using edge-overlap. In [10], this result was generalized to any OGSM on labeled or unlabeled, directed or undirected graphs using edge overlap or vertex overlap and isomorphism, homomorphism or homeomorphism. Our conditions for anti-monotonicity are similar but based on overlap hypergraphs. In particular, we show that an OHSM is anti-monotonic if and only if it is non-decreasing under three operations on the overlap hypergraph.

We begin by defining these three operations on any overlap hypergraph, which we will then use in our conditions for anti-monotonicity. These operations are different from those used in [66, 10], but play a similar role. As mentioned in these earlier papers, the motivation for the operations defined below is that it is often easier to show that an OHSM satisfies the conditions of the theorem (being non-decreasing under the three operations), than to directly demonstrate the anti-monotonicity of a measure.

Definition 3.20 (Hypergraph operators)**.** For $H \in \mathcal{H}$, we define:

- Vertex Addition (VA): A new vertex v is added to every existing hyperedge: $VA(H, v) = (V_H \cup \{v\}, \{e \cup \{v\} \mid e \in E_H\})$.
- Subset Contraction (SC): Let $K \subseteq V_H$ be a set of vertices of the hypergraph such that $\exists e \in E_H : K \subseteq e$. Then, the subset contraction operation contracts K into a single vertex k, which remains in only those hyperedges that are supersets of K. Formally, $SC(H, K, k) = (V_H \setminus K \cup \{k\}, E_1 \cup E_2)$ where $E_1 = \{e \setminus K \cup \{k\} \mid e \in E_H \text{ and } K \subseteq e\}$ and $E_2 = \{e \setminus K \mid e \in E_H \text{ and } K \nsubseteq e\}$).
- Hyperedge Split (HS): This operation splits a size k hyperedge into k hyperedges of size $(k-1)$ each: $HS(H, e) = (V_H, E_H \setminus \{e\} \cup \{e \setminus \{v\} \mid v \in e\})$, where $e \in E_H$.

For example (see Fig. 3.3), suppose H_0 is a hypergraph with $V_{H_0} = \{v_1, v_2, v_3, v_4\}$, and E_{H_0} contains two hyperedges $\{v_1, v_2, v_3\}$ and $\{v_1, v_4\}$. Let $H_1 = VA(H_0, v_5)$, then $V_{H_1} = \{v_1, v_2, v_3, v_4, v_5\}$ and E_{H_1} contains hyperedges $\{v_1, v_2, v_3, v_5\}$ and $\{v_1, v_4, v_5\}$. Let $H_2 = SC(H_1, \{v_1, v_3\}, v_6)$, then $V_{H_2} = \{v_2, v_4, v_5, v_6\}$ and E_{H_2} contains hyperedges $\{v_2, v_5, v_6\}$ and $\{v_4, v_5\}$. Let

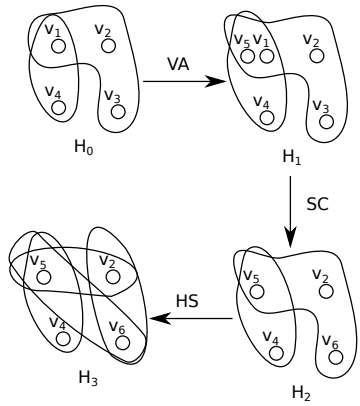


Figure 3.3: An example of the three operations defined on hypergraphs

$H_3 = HS(H_2, \{v_2, v_5, v_6\})$, then $V_{H_3} = V_{H_2}$ and E_{H_3} contains four hyperedges $\{v_2, v_5\}$,$\{v_2, v_6\}$, $\{v_5, v_6\}$ and $\{v_4, v_5\}$.

Sufficient condition

We present a sufficient condition for support measure anti-monotonicity in terms of the three operations on overlap hypergraphs that we have defined.

Theorem 3.21. *Let $f : \mathcal{H} \to \mathbb{R}$ be a hypergraph measure and $f' : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ be an overlap hypergraph based measure such that $f'(D, P) = f(\tilde{H}_P^D)$. If f is non-decreasing under VA, SC and HS, then f' is an anti-monotonic support measure.*

Proof. Suppose D is a database graph, and P and P' are two subgraph patterns such that P' is a subgraph of P. We prove that $\tilde{H}_{P'}^D$ can be obtained from $\tilde{H}_P^D$ by applying only the operations VA, SC and HS. It follows then that $f'(D, P) = f(\tilde{H}_P^D) \leq f(\tilde{H}_{P'}^D) = f'(D, P')$ for any D, P and P', proving the theorem.

Let $<$ be an arbitrary total order defined on $V_{\tilde{H}_{P'}^D}$. For $v \in V_{\tilde{H}_{P'}^D}$, we define the set $\Pi_v = \{u \in V_{\tilde{H}_P^D} \mid v \preceq u \text{ and } \forall w < v : w \not\preceq u\}$. Remind, $\preceq$ stands for subgraph isomorphism, hence Π_v consists of each image of P that contains v, an image of P', as far as it has not been part of some Π_w with $w < v$. Hence the sets Π_v are pairwise disjoint; moreover, every image of P is an element of some Π_v, i.e., $\{\Pi_v\}_{v \in V_{\tilde{H}_{P'}^D}}$ is a partition of $V_{\tilde{H}_P^D}$.

We point out that there may exist vertices v for which $\Pi_v = \emptyset$. We divide $V_{\tilde{H}^D_{P'}}$ into two sets $V_0 = \{v \mid \Pi_v = \emptyset\}$ and $V_1 = \{v \mid \Pi_v \neq \emptyset\}$.

Let H be a hypergraph initially equal to $\tilde{H}^D_P$. We will perform operations VA, SC and HS on H, until it is finally equal to $\tilde{H}^D_{P'}$.

First, H is modified by a sequence of VA operations. For each $v \in V_0$, we do $H := VA(H, v)$. Now, $\forall e \in E_H : V_0 \subseteq e$.

Then, for each $v \in V_1$, we perform $H := SC(H, \Pi_v, v)$. The operations are valid because for $v \in V_1$ each vertex $u \in \Pi_v$ stands for a superimage of the same v, i.e., $v \preceq u$ and hence $\exists e \in E_H : \Pi_v \subseteq e$. Note that a vertex is created for every $v \in V_0$ and for every $v \in V_1$. Moreover, contraction removes the vertices of the original H^D_P. As every vertex was part of exactly one Π_v, they are all removed and hence now the vertices of H are exactly the vertices of $H^D_{P'}$.

It remains to show that a sequence of HS operations can reduce the edges of H to the edges of $H^D_{P'}$. As all hyperedges are dominating, it suffices to show that, for every hyperedge $e \in H^D_{P'}$, there is a hyperedge in H that is a superset of e. Consider the vertex set $e_x = \{v \in V^D_{P'} | x \in V_D \text{ and } x \in v\}$, i.e., x is a vertex of the database D and x is part of the images that form the set. If e_x is not a hyperedge of $H^D_{P'}$, it is because another hyperedge dominates e_x. Let e'_x be the dominating hyperedge, i.e., $e'_x \in H^D_{P'}$ and $e_x \subseteq e'_x$.

We can divide e'_x into two disjoint parts, namely $e'_x \cap V_0$ and $e'_x \cap V_1$. The former is part of every hyperedge; so it remains to show that there is a hyperedge in H that contains $e'_x \cap V_1$.

In H^D_P, there is a hyperedge e''_x that is equal to or dominates the vertex set $\{v \in V^D_P | x \in V_D \text{ and } x \in v\}$. Now consider a vertex $v \in e'_x \cap V_1$; it is part of V_1 hence $\Pi_v \subseteq e''x$ and there is a contraction that introduces v in e''_x. This holds for each vertex in $e'_x \cap V_1$ hence H has a hyperedge that contains $e'_x \cap V_1$ and hence e'_x.

□

Theorem 3.22. *$\nu^*(D, P) = \nu^*(\tilde{H}^D_P)$ is a normalized anti-monotonic support measure.*

Proof. First, we prove that ν^* is normalized. If the subgraph pattern P only has non-overlapping images in the database graph D, every hyperedge in $E_{\tilde{H}^D_P}$ contains only one vertex, then setting $x_v = 1$ for every $v \in V_{\tilde{H}^D_P}$ is a feasible assignment and is clearly maximal. That is, ν^* equals the number of non-overlapping images. Therefore, ν^* is normalized.

Then, we prove ν^* is anti-monotonic using Theorem 3.21. Suppose H is an overlap hypergraph and x^* is an optimal solution to the LP of ν^*_H. Let H_1 be the overlap hypergraph $VA(H, v)$, and let $x_u = x^*_u$ for all vertices $u \neq v$ and $x_v = 0$. The vector x is a feasible solution for the LP of $\nu^*_{H_1}$, so $\nu^*_{H_1} \geq \sum_v x_v = \nu^*_H$. Let H_2 be the overlap hypergraph $SC(H, K, k)$, and let $x_u = x^*_u$ for all vertices $u \neq k$ and $x_k = \sum_{v \in K} x^*_v$. The vector x is a feasible solution for the LP of $\nu^*_{H_2}$, so $\nu^*_{H_2} \geq \sum_v x_v = \nu^*_H$. Let H_3 be the overlap hypergraph $HS(H, e)$. The vector x^* is also a feasible solution for the LP of $\nu^*_{H_3}$, so $\nu^*_{H_3} \geq \nu^*_H$. □

Necessary condition

We show that the condition for anti-monotonicity mentioned above is not only a sufficient but also a necessary condition.

Theorem 3.23. *Let $f : \mathcal{H} \to \mathbb{R}$ be a hypergraph measure and $f' : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ be an overlap hypergraph based measure such that $f'(D, P) = f(\tilde{H}^D_P)$. If f' is anti-monotonic, then f is non-decreasing under VA, SC and HS.*

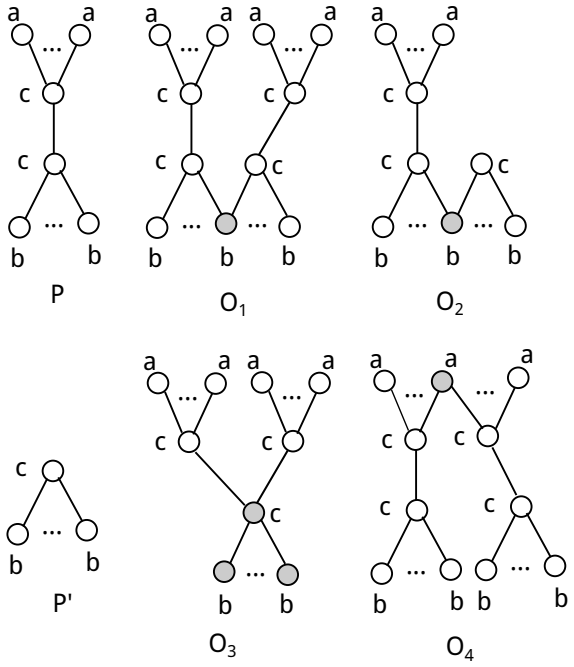


Figure 3.4: Two patterns and different types of overlap. The highlighted parts show the ways two images overlap.

Proof. Let H_P be any hypergraph and $H_{P'}$ a hypergraph obtained by performing VA, SC or HS on H_P. We show that there exists a database graph D and subgraph patterns P and P' such that $\tilde{H}_P^D = H_P$ and $\tilde{H}_{P'}^D = H'_P$ (it is not harmful to assume that H_P and $H_{P'}$ only contain dominating hyperedges). It follows then that $f(H_P) = f'(D, P) \leq f'(D, P') = f(H_{P'})$, which proves the theorem. For convenience, we only show the theorem for undirected labeled graphs, but the proof can be generalized.

In Figure 3.4, we give two subgraph patterns P and P' ($P' \preceq P$), and list 4 different possible types of overlap. The numbers of vertices with label a (called a-vertex) and b (called b-vertex) in P and P' are not fixed, but the pattern P always has 2 vertices labeled c (called c-vertex) while the pattern P' always has only one c-vertex. We construct the database graph D by combining multiple copies of the patterns P and P', overlapping in different ways. We name the different types of overlap O_1, O_2, O_3 and O_4. We say two or more copies of the pattern P have overlap type O_1 if they share a common b-vertex. Two or more copies of P and a copy of pattern P' have overlap type O_2 if they share a b-vertex. Two or more copies of the pattern P have overlap type O_3 if they share a subgraph isomorphic to P' (i.e., the bottom part of P in the figure). Two or more copies of the pattern P have overlap type O_4 if they share an a-vertex. As in our construction of the database graph D we do not use other types of overlap (e.g., we will not need copies of P that share c-vertices), every copy of P in D (whether overlapping with another copy or not) gives rise to one vertex in the overlap hypergraph H_P^D and one vertex in $H_{P'}^D$ and every copy of P' in D gives rise to one vertex in $H_{P'}^D$.

Let us first consider the case $H_{P'}^D = VA(H_P^D, v)$. Recall, the effect of VA is to create a new node v and to add it to all existing edges. Let m be the number of vertices and n be the number of edges in H_P^D. Then the pattern P has n b-vertices and no a-edges. Also the pattern P' has n b-edges, i.e., one b for each hyperedge of H_P^D. The database has n copies of P and one copy of P', so the overlap graphs H_P^D and $H_{P'}^D$ have respectively n and $n+1$ vertices as desired. We denote the j^{th} b in the i^{th} copy of P as $b_{i,j}$ and the j^{th} b in the copy of P' as b_i. As for the sharing of vertices in the database, consider the i^{th} hyperedge e_i of H_P^D and assume $e_i = \{v_{e_1}, \ldots, v_{e_k}\}$ of H_P^D. To get this hyperedge in H_P^D we let $b_{e_1,i} = \ldots = b_{e_k,i}$, i.e., the copies $e_j (1 \leq j \leq k)$ of P in D share their i^{th} b as in case O_1 of Figure 3.3. This sharing gives rise to the edge e_i in H_P^D as desired. Moreover we set also $b_{e_1,i} = b_i$, i.e., each copy $e_j (1 \leq j \leq k)$ of P share their i^{th} b with the i^{th} b of the copy of P' as in case O_2 of Figure 3.3. This gives rise to the edge $\{v_{e_1}, \ldots, v_{e_k}, v\}$ in H_p^D, hence $H_{P'}^D = VA(H_P^D, v)$ as desired.

Next, we consider the case $\tilde{H}_{P'}^D = SC(\tilde{H}_P^D, K, k)$. We let the pattern P have

$|E_{\tilde{H}^D_{P'}}|$ b-vertices and $|E_{\tilde{H}^D_{P'}}|$ a-vertices. We let the pattern P' also have $|E_{\tilde{H}^D_{P'}}|$ b-vertices. The database graph D consists of $|V_{\tilde{H}^D_P}|$ P-images, such that $|K|$ of them (the ones corresponding to the contracted subset) share a subgraph (corresponding to k) isomorphic to P' (i.e., their overlap type is O_3). Next to these overlaps, there are overlaps determined by $\tilde{H}^D_P$: for every hyperedge $e \in E_{\tilde{H}^D_{P'}}$ that satisfies $e \cap K \neq \emptyset$ and $K \nsubseteq e$, we let the P-images corresponding to vertices participating in e share the a-vertex corresponding to e (i.e., their overlap type is O_4) and the P-images corresponding to vertices participating in $e \setminus K$ share a b-vertex (i.e., their overlap type is O_1). Then, the hyperedge $e \setminus K$ appears in $E_{\tilde{H}^D_{P'}}$. For every hyperedge $e \in E_{\tilde{H}^D_{P'}}$ that satisfies $e \cap K = \emptyset$, we let the P-images corresponding to vertices participating in e share a b-vertex (i.e., their overlap type is O_1). These O_1 overlaps imply that e appears in $E_{\tilde{H}^D_{P'}}$ when $e \cap K = \emptyset$. For every hyperedge $e \in E_{\tilde{H}^D_{P'}}$ that satisfies $K \subseteq e$, we let the P-images corresponding to vertices participating in $e \setminus K$ and the P'-image which corresponds to k share a b-vertex (i.e., their overlap type is O_2). These O_2 overlaps imply that $e \setminus K \cup \{k\}$ appears in $E_{\tilde{H}^D_{P'}}$ when $K \subseteq e$. This construction satisfies $\tilde{H}^D_{P'} = SC(\tilde{H}^D_P, K, k)$.

Finally, we consider the case $\tilde{H}^D_{P'} = HS(\tilde{H}^D_P, e)$. We let the pattern P have $|E_{\tilde{H}^D_{P'}}|$ b-vertices, and one a-vertex. We let the pattern P' also have $|E_{\tilde{H}^D_{P'}}|$ b-vertices. The database graph D consists $|V_{\tilde{H}^D_P}|$ P-images, each corresponding to a vertex of $\tilde{H}^D_P$, with the overlaps among images determined by $\tilde{H}^D_P$ as follows. We let the P-images corresponding to vertices participating in e share their a-vertex (i.e., their overlap type is O_4), and for every $v \in e$ we let the P-images corresponding to vertices participating in $e \setminus \{v\}$ share a b-vertex (i.e., their overlap type is O_1). These overlaps make sure that e appears in $E_{\tilde{H}^D_{P'}}$ and, for all $v \in e$, $e \setminus \{v\}$ appears in $E_{\tilde{H}^D_{P'}}$. For every other hyperedge e' in $E_{\tilde{H}^D_{P'}}$, we let the P-images corresponding to vertices participating in e' share the b-vertex which determines e' (i.e., their overlap type is O_1). These O_1 overlaps imply that e' also appears in $E_{\tilde{H}^D_{P'}}$. This constructions satisfies $\tilde{H}^D_{P'} = HS(\tilde{H}^D_P, e)$. □

Bounding theorem

In [10], the authors showed that all normalized anti-monotonic OGSMs are bounded (between the maximum independent set size (MIS) and the minimum clique partition size (MCP)). Similarly, we prove that all normalized anti-monotonic OHSMs are also bounded. We first introduce another OHSM, the size of a minimum set cover (MSC) of overlap hypergraphs:

$$MSC(D,P) = MSC(\tilde{H}_P^D) = \min |\{S \subseteq E_{\tilde{H}_P^D} \mid \bigcup_{e \in S} e = V_{\tilde{H}_P^D}\}| \tag{3.6}$$

It is not difficult to verify that MSC is normalized and anti-monotonic. Computing MSC is an NP-hard problem. The maximum independent set size (Eq. (3.3)) and minimum set cover (Eq. (3.6)) are the minimally and maximally possible normalized anti-monotonic OHSMs.

Theorem 3.24. *Given a database graph D, and a subgraph pattern P, it holds that $MIS(D,P) \leq f(D,P) \leq MSC(D,P)$ for every normalized anti-monotonic OHSM $f(D,P) = f'(\tilde{H}_P^D)$.*

Proof. We use Theorem 3.23 to show the minimality of MIS and the maximality of MSC.

On the one hand, let $I = \{v_1, v_2, \cdots, v_k\}$ be a maximum independent set of $\tilde{H}_P^D$. Starting from the hypergraph $H_I = (\{v_1, v_2, \cdots, v_k\}, \{\{v_1\}, \{v_2\}, \cdots, \{v_k\}\})$, we can get $\tilde{H}_P^D$ by adding vertices $V_H \setminus I$ using VA first and then splitting hyperedges by a sequence of HS. Since f is normalized, $f'(H_I) = k$. Because f is anti-monotonic, f' cannot decrease after each step and $f(D,P)$ is larger than or equal to $k = MIS(D,P)$.

On the other hand, let $\{e_1, e_2, \cdots, e_k\}$ be a minimum set cover for $\tilde{H}_P^D$ and let $H_{sc} = SC(\ldots SC(SC(H, e_1, v_{e_1}), e_2, v_{e_2}) \cdots, e_k, v_{e_k})$. H_{sc} only has the hyperedges with exactly one vertex in each of them. Because f is anti-monotonic, f' does not decrease under SC and thus $f(D,P) \leq f'(SC(H, e_1)) \leq \cdots \leq f'(H_{sc}) = MSC(D,P)$. □

Relaxation of the OGSM MIS

One may ask whether the ν^* support can be defined by relaxing the OGSM MIS instead of the OHSM MIS. In other words, is the concept of overlap hypergraphs really necessary?

Our answer is that the concept of overlap hypergraph is needed for the definition of the ν^* support measure because it carries additional information on the overlap graph. In particular the hyperedges show which overlaps have a common cause. If we did not have this information, we would not be able to reconstruct it. For instance (see Fig. 3.5), if we see a triangle in an overlap graph, we do not know whether this triangle originates from one vertex shared by the three images or from three vertices, each shared by two of the images. This

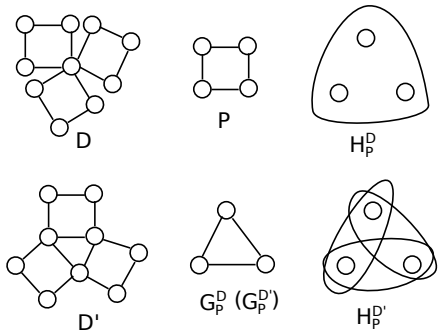


Figure 3.5: An example of the case that two different overlap hypergraphs correspond to the same overlap graph

additional information is needed for the definition of ν^*, and for its mathematical properties.

3.4 Properties of the ν^* support measure and discussion

In this section, we discuss several aspects of our approach, pointing out interesting properties and possible future extensions.

Phase transition from frequent to infrequent

Large real-world networks are known to satisfy properties similar to random graphs. A well-known result is that properties that can be expressed in first order logic are satisfied either by almost all graphs or by almost none (0-1 law, see [20]). For random graphs, it is either very easy to embed a given subgraph pattern P in the network, or very difficult (see also our experiments below). This leads to another 0-1 property: the frequency of many subgraph patterns is either very low or very high (for our ν^* measure, nearly equal to the network size). Consider for instance a social network and the subgraph pattern "X is a friend of Y and Y is a friend of Z". Since most people have at least two friends, such a subgraph pattern will match about everywhere. This holds more generally for many tree and path subgraph patterns. In fact, most such subgraph patterns are overly general and not very interesting.

The notion of overlap

Overlap can be defined in several ways, for instance as vertex-overlap or as edge-overlap. In [66] and [10], the authors showed that overlap-based support measures are anti-monotonic for the vertex-overlap setting. Furthermore, they claimed that the results are also valid for the edge-overlap setting, even though this should be treated more carefully. Consider, for instance, an unlabeled database graph D which satisfies $|E_D| > |V_D|$, a subgraph pattern P_1 which consists of a single vertex and another subgraph pattern P_2 which consists of a single edge. If we use edge-overlap, the support of P_1 in D ($|V_D|$) is smaller than that of P_2 in D ($|E_D|$). Still, this violation of the anti-monotonicity property only applies to these small patterns and does not affect larger patterns. As such, it does not significantly inhibit the use of support measures to prune the pattern mining search.

Definitions of overlap can be generalized. For example, the following notions of overlap could also be considered:

- *two-vertex (edge) overlap:* two images overlap if and only if they share two or more common vertices (edges);
- *label-specific overlap:* two images overlap if and only if they share a common vertex (edge) which has a label in a certain set;
- *distance-based overlap:* Two images u and v overlap if u has a vertex x and v has a vertex y such that the distance between x and y is smaller than a specified constant *min_dist*.

In order to study the anti-monotonicity of the two-vertex (edge) overlap and the label-specific overlap, we need the following definition.

Definition 3.25 (S-overlap)**.** Given a set S of graphs $\{g_i\}_{i=1}^{|S|}$ (some of them may be unconnected), a database graph D and a pattern P, two images $u, v \in \mathrm{Img}(D, P)$ overlap if and only if there exists at least one pattern $g \in S$ such that u and v share an image of g.

The vertex overlap, the edge overlap, the two-vertex (edge) overlap and the label-specific overlap are essentially S-overlap. For instance, two-vertex overlap is an S-overlap where S only contains the graph with two isolated vertices.

For S-overlap, we can show the following result.

Theorem 3.26. *Given a set S of graphs $\{g_i\}_{i=1}^{|S|}$, a database graph D and two patterns P, p where $p \subseteq P$, if we use S-overlap and there exists $g \in S$ such that $g \subseteq p$, then H_p^D can be obtained by performing VA, SC or HS on H_P^D.*

For the distance-based overlap, a similar result holds.

Theorem 3.27. *Given a database graph D and two patterns P, p where $p \subseteq P$, if we use the distance-based overlap, then H_p^D can be obtained by performing VA, SC or HS on H_P^D.*

The proofs of the two theorems above are similar to the proof of Theorem 3.21.

3.5 Implementation

To conduct the experiments in Section 3.6, we implemented the ν^* support measure in a pattern mining system[2]. In this section, we discuss a number of implementation-related issues.

As we mentioned in Chapter 2, graph miners have three components: a pattern matcher, a candidate generator and a support measure. We briefly describe the basic pattern matcher and the simple candidate generator used in our experiments. We also present methods to handle patterns with too many embeddings (images) and to optimize the computation of overlap (hyper)graph based anti-monotonic support measures.

Pattern matching

As this chapter focuses on the computation of a support measure, efficient pattern matching is not our main concern. We implemented a simple pattern matcher with a few optimizations. In particular, patterns are searched by backtracking. Images for the various vertices of a pattern are selected on the basis of a heuristic. Before processing a pattern P of k vertices, we compute the embeddings of all subpatterns of $k-1$ vertices. We store the time cost of the pattern matching for all these parents and chose the 'best' parent p, i.e., the subgraph of size $k-1$ with the smallest number of embeddings and the least time spent on pattern matching. Then, for the pattern matching of P, we order the vertices as they were ordered for p, and the vertex of P which is not in p is put at the end of the sequence. Thus, we expect to be able to prune the search space at an early stage.

[2]This system is part of the MIPS project, available from `http://people.cs.kuleuven.be/~jan.ramon/MiGraNT/MIPS/`

Handling patterns with many embeddings

An important problem in constructing a graph pattern miner for a single network is that some patterns have an extremely large number of embeddings. As explained in Section 3.4, we believe that such patterns are not very interesting, and we do not calculate their exact supports. In particular, we impose a threshold τ on the number of embeddings to be found and stop searching when the threshold is reached. If a pattern has more embeddings than than the threshold, we only compute a lower bound of its support using these found embeddings.

To avoid that all embeddings we found share the same first vertex, we designed our pattern matcher to perform a cyclical breadth-first search: whenever we find an embedding of a pattern, we try another image for the first vertex of the pattern next. When we have tried to find an embedding mapping the first pattern vertex on every vertex of the network, we start over again. Thus, we ensure that embeddings are distributed evenly over the network.

This approach yields three different types of subgraph patterns.

- Interesting subgraph patterns, i.e., subgraph patterns which are frequent enough to pass the minimum support threshold while the number of embeddings remains under the maximum embedding threshold $\{P \mid \nu^*(D,P) \geq \sigma \text{ and } |\text{Emb}(D,P)| \leq \tau\}$.
- Infrequent subgraph patterns, i.e., subgraph patterns which are not frequent enough to pass the minimum support threshold while the number of embeddings remains under the maximum embedding threshold $\{P \mid \nu^*(D,P) < \sigma \text{ and } |\text{Emb}(D,P)| \leq \tau\}$.
- Overly frequent subgraph patterns, i.e., subgraph patterns of which the number of embeddings is larger than the predefined maximum #embedding threshold $\{P \mid |\text{Emb}(D,P)| > \tau\}$. Still, in practice these patterns are usually frequent, and the lower bounds are close to the real supports.

We find that, in practice, a number of embeddings equal to at most ten times the number of vertices in the graph is sufficient to discover all frequent patterns.

The pattern miner

We follow the classical level-wise pattern mining approach, with some modifications for patterns with an extremely large number of embeddings

(as discussed above). Our miner contains a candidate generator, a pattern matching algorithm (as discussed above) and a module for support measure computation (the main topic of this chapter).

The algorithm maintains a list of candidate patterns and a list of solutions (frequent patterns). It starts with an empty pattern in the candidate list. It processes the list of candidate patterns one by one. First, it calculates the frequency of the pattern. If the pattern is frequent, it is added to the list of solutions, and all its extensions (see the candidate generation explanation) are added to the end of the candidate list. As an initialization, the empty candidate is considered frequent.

Candidate generation is performed by adding a single vertex to a frequent graph and the new vertex must be connected to some existing vertex. This choice is preferred above the more common choice of building patterns edge by edge. Indeed, sparser graphs have a larger number of embeddings and there is a higher risk of reaching the upper bound on the number of embeddings to compute. Hopping from dense pattern to dense pattern guarantees accurate results for the dense (and hence more interesting) patterns.

Even though we use a canonical form, we do not use canonical refinement. We generate a candidate if there is at least one parent that is frequent in the sense that we have an exact value of or lower bound on its support, higher than the minimum support threshold.

Image elimination

Although the linear program can be solved very efficiently, the number of images may grow exponentially with the size of subgraph patterns. Therefore, effective optimizations can help us compute the support. In this section, we demonstrate that some images can be shown to contribute 0 to the support without being included in the linear program. The following theorem holds for every OHSM.

Theorem 3.28. *Suppose H is an overlap hypergraph and there exist two vertices v and w in V_H for every edge $e \in E_H$ such that $v \in e$ implies $w \in e$. Let $H' = H[V_H \setminus \{w\}]$ be the induced sub-hypergraph of H on the vertex set $V_H \setminus \{w\}$, then $f(H) = f(H')$ if f is any anti-monotonic OHSM.*

Proof. We use the fact that the anti-monotonic OHSM cannot decrease under VA (Vertex Addition), HS (Hyperedge Split) and SC (Subset Contraction). Let H be an overlap hypergraph and v and w as defined above.

On the one hand, let $H'' = VA(H', w)$, then $f(H'') \geq f(H')$ because f cannot decrease under VA. Because H can be obtained by a series of HS from H'', we have that $f(H) \geq f(H'')$. According to the two inequalities, $f(H) \geq f(H')$.

On the other hand, because $H' = SC(G, \{v, w\}, v)$ and f cannot decrease under SC, we have that $f(H') \geq f(H)$. Thus, $f(H) = f(H')$. □

This theorem tells us that if the hyperedges containing image v are a subset of the hyperedges containing image w, then we can eliminate the image w without changing any anti-monotonic OHSM. We call the image w an *unnecessary image*.

In our experiments, we found that many images are only contained in a single dominating hyperedge. For these images, we can use the following corollary.

Corollary 3.29. Suppose $H = (V, E)$ is an overlap hypergraph, let $v \in V_H$ be an image which is only contained in a single hyperedge and $N(v)$ be the set of vertices that are adjacent to v. If f is an anti-monotonic OHSM, then $f(H) = f(H[V \setminus \{v\} \setminus N(v)]) + 1$.

We can show similar results for any OGSM.

Theorem 3.30. *Suppose G is an overlap graph and there exist two adjacent vertices v and w in V_G such that $N(v) \setminus \{w\} \subseteq N(w) \setminus \{v\}$, i.e., every neighbour of v (apart from w) is also a neighbour of w, let $G' = G[V_G \setminus \{w\}]$ be the induced subgraph of G on the vertex set $V_G \setminus \{w\}$, then $f(G) = f(G')$ if f is any anti-monotonic OGSM.*

Proof. We use the fact that the anti-monotonic OGSM cannot decrease under VA (Vertex Addition), ER (Edge Removal) and CC (Clique Contraction). Let G be an overlap graph and v and w as defined above.

On the one hand, let $G'' = VA(G', w)$, then $f(G'') \geq f(G')$ because f cannot decrease under VA. We can consider G is obtained by a series of ER from G'', we have that $f(G) \geq f(G'')$. According to the two inequalities, $f(G) \geq f(G')$.

On the other hand, because $G' = CC(G, \{v, w\}, v)$ and f cannot decrease under CC, we have that $f(H') \geq f(H)$. Thus, $f(G) = f(G')$. □

If an overlap graph has m edges and a maximal degree d it only takes $O(md)$ time to eliminate all the unnecessary images from the OGSM. Hence, eliminating these unnecessary images first can speed up the computation of support measures, especially for the more expensive ones such as the Lovász ϑ and the SGM support measures.

In our experiments, we found that almost half the images in every overlap graph are unnecessary. This may inspire us to find more efficient ways to reduce the size of overlap graphs, enabling us to use the Lovász ϑ support measure and the SGM support measure in practice.

3.6 Experiments

This section provides experimental results, illustrating the practical potential of our new measure ν^*.

Experimental setup

Our experiments address the following experimental questions:

- Q1 How does the computational cost of the ν^* measure compare to other existing overlap based support measures, for instance, the Lovász ϑ value?
- Q2 How high is the cost of computing the ν^* measure?
- Q3 Is it feasible to mine all ν^*-frequent subgraph patterns in moderately sized networks?
- Q4 What can we learn about the phase transition between frequent and infrequent patterns and the randomness of the real-world dataset?

All experiments were run on an Intel Core i7-2600 CPU (3.4Gz) with 8Gb RAM.

Data

In our experiments, we consider both synthetic and real-world data. We first give an overview of the datasets used.

Synthetic datasets

We constructed random hypergraphs RndHyp-V-E; where V is the number of vertices and E is the number of hyperedges. In these random hypergraphs, for every vertex v and hyperedge e with probability 0.05, v is an element of e.

We also generated scale-free networks of different sizes, using the linear preferential attachment processes described in [3]. We denote such networks with 10^N vertices BARABASI-N. The network begins with an initial network of $m_0 \geq 2$ vertices (here, $m_0 = 5$) which are fully connected. New vertices are added to the network one at a time until the size of the network reaches 10^N. Each new vertex is connected to m existing vertices (in this case, $m = 5$) with a probability proportional to the number of edges that these existing nodes already have. Formally, the probability p_i that the new vertex is connected to vertex i is

$$p_i = k_i / \sum_j k_j$$

where k_j is the degree of the vertex j. Vertices are randomly labeled by 4 different labels (the labels do not affect the network generating procedure).

Real-world datasets

We use two labeled real-world datasets: Facebook and DBLP. Table 3.1 shows all real-world datasets with elementary statistics.

In the Facebook network, vertices represent users while edges represent friendship relations. The label of a vertex represents the privacy setting of the corresponding user. We consider two Facebook samples[3]: a uniform sample (FACEBOOK-UNIFORM) and a Metropolis-Hastings random walk sample (FACEBOOK-MHRW) [28].

We use two DBLP co-authorship networks (DBLP-0305 showing co-authorships from 2003 to 2005 and DBLP-0507 showing co-authorships from 2005 to 2007) [4] in which vertices represent authors. If an author i co-authored a paper with author j, the networks contain an undirected edge $\{i, j\}$. The vertices are unlabeled, whereas the edges are labeled with an integer indicating the year in which the edge first appeared.

Table 3.1: Overview of real-world datasets.

Dataset	Version	# vertices	# edges	labels
FACEBOOK	UNIFORM	984830	371021	4 vertex labels
FACEBOOK	MHRW	957359	3584376	4 vertex labels
DBLP	0305	109944	228461	3 edge labels
DBLP	0507	135516	290363	3 edge labels

[3] http://odysseas.calit2.uci.edu/doku.php/public:online_social_networks

Results

The Lovász ϑ support measure vs. the ν^* support measure

We compared the time needed to compute the Lovász ϑ function with the time needed to compute the ν^* measure on random hypergraphs. In particular, we consider random hypergraphs RndHyp-V-E with $V \in \{20, 40, 60, \ldots, 200\}$ and $E \in \{20, 40, 60, 80, 100\}$. We evaluate ν^* directly on the hypergraphs, while in order to evaluate the Lovász ϑ function we first convert the hypergraphs to graphs by replacing every hyperedge with a clique.

Fig. 3.6 shows the time cost of computing the ν^* measure and the Lovász ϑ measure for these graphs. The symbols θ-E and ν^*-E mean that there are E hyperedges in that hypergraph.

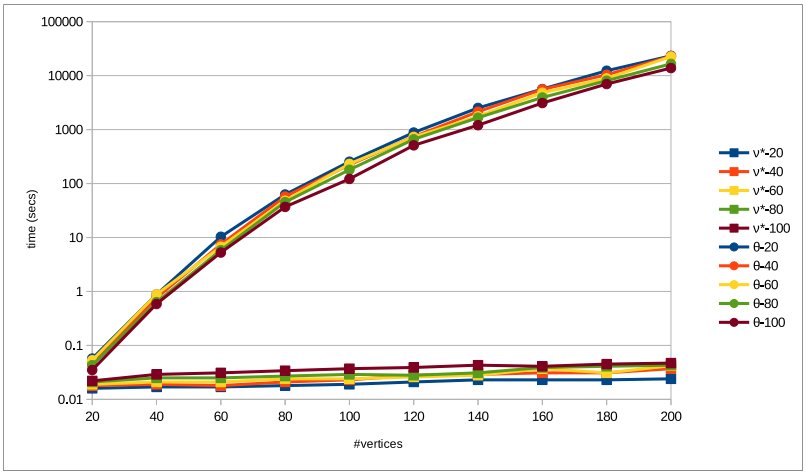


Figure 3.6: Time required for computing the Lovász ϑ and ν^*.

Mining patterns in real-world data

Table 3.2 gives the results of our algorithm for mining frequent subgraph patterns in the Facebook networks with a minimal frequency threshold of 0.2%. Table 3.3 provides the same information for DBLP mined with a frequency threshold of 1%.

As described in Section 3.5, for patterns with a huge number of embeddings we only collect a limited number of them in our mining algorithm. Here, we put

the threshold at 100000 embeddings. Only few patterns have more than that number of embeddings, e.g., in FACEBOOK-MHRW we discovered 6 of them (all are path patterns). All patterns with more than 100000 embeddings were found to be frequent (thanks to our breadth-first way of searching for embeddings), so we didn't miss any frequent pattern.

Table 3.2: Frequent subgraph pattern mining in FACEBOOK-UNIFORM network (left) and FACEBOOK-MHRW network (right) with minimal support 0.2%. Lev = level (number of vertices), Cand = # candidate subgraph patterns, Freq = # frequent subgraph patterns, T_{map} = average time per pattern needed to compute images, T_{ν^*} = average time needed to compute the support measure. Time units are in seconds.

Lev	Cand	Freq	T_{map}	T_{ν^*}
1	3	3	0.01	0.00
2	2	2	0.03	1.88
3	7	6	0.26	1.96
4	41	10	0.26	0.48
5	67	11	0.15	0.40
6	72	16	0.36	0.99
7	168	26	0.31	1.63
8	337	36	0.39	2.32
9	515	43	1.63	4.23
10	847	64	0.25	5.60
11	1268	85	0.47	11.12
12	1849	25	0.83	15.12

Lev	Cand	Freq	T_{map}	T_{ν^*}
1	2	2	0.01	0.00
2	2	2	0.57	1.72
3	7	5	0.79	12.43
4	40	27	0.75	10.87
5	220	101	1.19	8.17
6	480	193	1.71	4.46
7	126	80	0.19	0.62
8	104	66	4.58	0.91
9	93	30	22.73	1.20
10	29	8	12.85	0.89
11	18	0	50.97	0.01

Table 3.3: Frequent subgraph pattern mining in DBLP-0305 network (left) and DBLP-0507 network (right) with minimal support 1%.

Lev	Cand	Freq	T_{map}	T_{ν^*}
1	1	1	0.00	0.00
2	3	3	0.05	0.91
3	24	21	0.09	2.21
4	559	167	0.24	20.59
5	10746	496	0.08	0.16
6	28737	1694	1.73	0.36

Lev	Cand	Freq	T_{map}	T_{ν^*}
1	1	1	0.00	0.00
2	3	3	0.14	2.16
3	24	17	0.17	2.56
4	204	93	0.17	13.8
5	689	420	0.21	0.44

Mining patterns in synthetic data

In the BARABASI-N datasets, all tree patterns are very frequent. As this is in line with the predictions of random graph theory, we only report statistics for

the non-tree subgraph patterns. We set the frequency threshold to 0.1%.

Tables (3.4)-(3.6) give the results of these experiments, in which BARABASI-N (with N $\in \{2, 3, 4, 5, 6\}$) are mined for frequent non-tree subgraph patterns up to level 6 (except for the network which has 10^6 vertices). Given that we only report non-tree patterns, no patterns smaller than 4 are reported.

Table 3.4: Frequent non-tree subgraph pattern mining in the BARABASI-2 network (left) and BARABASI-3 network (right) with minimal support 0.1%.

BARABASI-2				
Lev	Cand	Freq	T_{map}	T_{ν^*}
4	20	16	0.014	0.383
5	191	182	0.015	0.388
6	2083	2033	0.018	0.394

BARABASI-3				
Lev	Cand	Freq	T_{map}	T_{ν^*}
4	20	20	0.02	0.38
5	215	215	0.03	0.43
6	2430	2422	0.13	0.48

Table 3.5: Frequent non-tree subgraph pattern mining in the BARABASI-4 network (left) and BARABASI-5 network (right) with minimal support 0.1%.

BARABASI-4				
Lev	Cand	Freq	T_{map}	T_{ν^*}
4	20	20	0.09	0.39
5	215	215	0.28	0.41
6	2430	2349	3.14	1.88

BARABASI-5				
Lev	Cand	Freq	T_{map}	T_{ν^*}
4	20	5	2.91	0.30
5	99	9	12.44	0.67
6	758	648	354.19	24.41

Table 3.6: Frequent non-tree subgraph pattern mining in the BARABASI-6 network with minimal support 0.1%.

BARABASI-6				
Level	Candidates	Frequent	T_{map}	T_{ν^*}
4	20	0	216.20	0.43
5	55	12	1565.13	0.99
6	-	-	-	-

Discussion

Based on the results presented above, we can answer the experimental questions as follows:

Q1 Figure 3.6 shows that, for all randomly generated (hyper)graphs, ν^* can be computed in a very short period of time (< 0.01 seconds), while the

time required to compute ϑ increases rapidly when the number of vertices increases. Clearly, for larger (hyper)graphs on which the ν^* measure can be computed efficiently, it is extremely difficult to compute the ϑ value in a reasonable time by solving the corresponding SDP using existing methods. Therefore, ν^* outperforms ϑ value in terms of efficiency.

Q2 The cost of computing ν^* mainly depends on the number of embeddings. This cost is low for the smallest patterns (having only a small number of embeddings with little overlap), and also, in several cases, for the largest patterns, probably because they are less frequent on average than the smaller ones.

Q3 The ν^* measure, allows us to mine frequent subgraph patterns of moderate size in a reasonable amount of time. The main bottleneck for DBLP is the explosive growth of the set of frequent patterns. In practical applications where it is more obvious which types of patterns are of interest, this problem could probably be overcome by selecting a smaller pattern language. For Facebook, only one label is really important, and the number of patterns increases much more slowly. In contrast to earlier approaches using the MIS or ϑ measures, where the computation of the support measure was the bottleneck, here both pattern matching and support measure computation contribute a significant fraction to the runtime. As we mainly selected the linear programming library (GLPK) based on its ease of interfacing with the other C code rather than on efficiency, we expect that the time required for support measure computation can be further reduced.

Q4 For the synthetic data, we found that cyclic (non-tree) subgraph patterns were fairly infrequent, as we needed a frequency threshold of 0.1% to mine them. This suggests that whereas in standard random graph models nodes choose their neighbors randomly, in real-world data the connections between candidate neighbors have an important influence.

3.7 Summary

In this chapter, we have studied the problem of measuring how frequently a given subgraph pattern occurs in a given database graph. We have proposed a new overlap based support measure ν^*, which unlike existing overlap based support measures can be computed efficiently. We have shown that this overlap based measure is anti-monotonic and normalized. The experimental results demonstrate that it is practical to use and effective in pruning the search space.

Our proposed measure is flexible, in the sense that it is possible for a user to plug in his own definition of overlap. It also makes it possible to measure the frequency of a subgraph pattern in a more sound statistical way. There are, however, other challenges related to subgraph pattern mining in networks. The major one in our experiments was subgraph pattern matching. However, we anticipate that here too we will be able to make considerable advances in finding tractable solutions. In particular, we intend to integrate our approach with recent results concerning efficient subgraph pattern matching operators based on arithmetic circuits [39].

In this chapter, we mention that the support measure should be larger if there are more independent observations. Starting from the next chapter, we make this point clearer from a statistical point of view.

Chapter 4

Networked variance

In many domains, e.g., scientific research and businesses, statistics and statistical machine learning play important roles. A crucial assumption made by many approaches in the field of statistics and machine learning is that observations are independently drawn. Most existing estimators are designed based on this assumption. However, this assumption does not hold for observations extracted from the same network since two or more observations may share some information.

In this chapter, we define networked examples and networked random variables, and propose a weaker independence assumption. The model of networked examples will be used in Chapter 6 for learning. The model of networked random variables is mainly used in statistics. We consider mean value estimators on networked random variables because many learning tasks and statistics involve mean value estimation.

Bias and variance are widely used to measure the quality of estimators. We restrict to unbiased weighted estimators in this chapter, and exploit Hoeffding's decomposition to minimize the variance. This variance minimization algorithm is efficient.

4.1 Networked examples

In this section, we introduce networked examples and a framework to represent them. The basic intuition is that a networked example combines a number of objects, each of which can carry some information (features). However, each of

these objects can be shared with other examples, such that we get a network. The sharing of information also makes explicit the dependencies between the examples.

Structure of networked examples

We use a hypergraph $G = (V, E)$ to represent a set of networked examples. The vertices $V = \{v_1, v_2, \ldots, v_m\}$ represent objects, and the hyperedges $E = \{e_1, e_2, \ldots, e_n\}$ represent examples grouping a number of objects. In what follows, we will often abuse terminology, identify vertices and objects and use the terms interchangeably. We consider two variants of networked examples: general networked examples and tuple networked examples. Tuple networked examples are special cases of general networked examples. For problems with tuple networked examples, the hypergraphs are restricted to be k-partite. For indexing, we use the common notation $[k] = \{i \in \mathbb{N} \mid 1 \leq i \leq k\}$. The following running examples illustrate both types of networked examples.

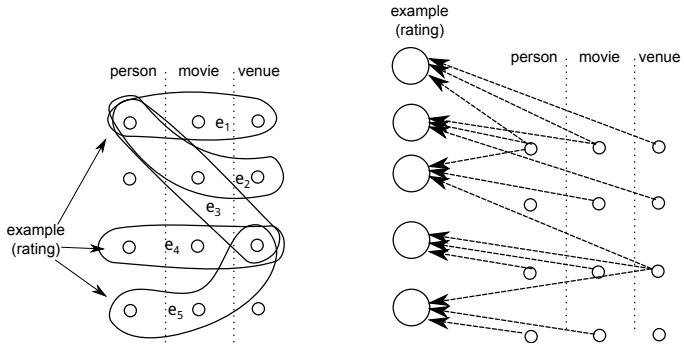
Example 4.1 (club)**.** Consider a dataset of clubs, i.e., every club is an example. A club is related to one or more themes and has two or more members. In our network representation, we have nodes for persons and for themes. A club is represented with a hyperedge containing the persons who are member of the club and the themes to which the club is related. Clubs can share members and/or themes. ⧫

The following example has been discussed in [64].

Example 4.2 (binary classification)**.** In a binary classification problem, we want to predict the relationship between some objects. For example, given two persons, a question is whether they are friends or not. In network analysis, this task is also called link prediction. We would have a vertex set V of persons and every hyperedge contains two persons. ⧫

Example 4.3 (court)**.** In court, lawsuits involves litigants (clients), lawyers and judges. We want to analyze the satisfaction of clients in their lawsuits. We can construct a network with a vertex set $V^{(1)}$ of litigants, a vertex set $V^{(2)}$ of lawyers and a set $V^{(3)}$ of judges. The hyperedges (training examples) of interest are then tuples $(litigant, lawyer, judge)$. ⧫

Example 4.4 (movie rating)**.** Consider the problem of predicting the rating by a person who watched a movie in a particular cinema. We can construct a network with a vertex set $V^{(1)}$ of persons, a vertex set $V^{(2)}$ of movies and a set $V^{(3)}$ of cinemas. Every hyperedge contains a person vertex, a movie vertex and a cinema vertex. Figure 4.1a illustrates this setup. ⧫



(a) represented by a hypergraph

(b) represented by a Bayesian network

Figure 4.1: Networked examples of movie rating

Features and target values

We denote with $\mathcal{X}$ the space of features of objects. Objects can be represented with feature vectors or other suitable data structures.

As usual in supervised learning, an example is a pair of an input and a target value. The input of an example is a collection of objects and is represented by a multiset of elements of $\mathcal{X}$. We denote with $\mathbb{X} = \mathbb{N}^{\mathcal{X}}$ the space of all possible examples.

Each example also has a target value, e.g., a class label or regression value. We denote with $\mathcal{Y}$ the space of all possible target values.

Example 4.5 (club)**.** Clubs have several members and several themes. Each member may be represented by a feature vector containing demographic data and a set of interests. Each theme may be described by a Wikipedia page on the theme. Suppose we want to model how active each club is. Then we could attach to each club as target value the frequency of its meetings. ♦

Example 4.6 (binary classification)**.** In the problem of predicting whether two persons are friends, every person has his/her own feature vector (containing e.g., interests, demographical info, education, ...) in the space $\mathcal{X}$, while an example (an element of $\mathbb{X}$) is a pair of such vectors. The target value of an example is a boolean value to show whether they are friends, i.e., $\mathcal{Y} = \{0, 1\}$. ♦

Example 4.7 (movie rating)**.** In the movie rating example, a movie (a vertex in $V^{(2)}$) can be described by a feature vector from $\mathcal{X}^{(2)}$ describing genre, actor popularity, ... A person (a vertex in $V^{(1)}$) who watches movies can be described

by a feature vector from $\mathcal{X}^{(1)}$ describing gender, age, nationality, ... A venue (a vertex in $V^{(3)}$) can be described by a feature vector from $\mathcal{X}^{(3)}$ containing cinema size, city, ... Then, $\mathbb{X} = \mathcal{X}^{(1)} \times \mathcal{X}^{(2)} \times \mathcal{X}^{(3)}$ is the space of feature vectors of complete examples, consisting of a concatenation of a movie feature vector, a person feature vector and a cinema feature vector. The target value of an example is the rating the person gave to the movie in the concerned venue, e.g., the space $\mathcal{Y}$ can be the set of integers $\{1, 2, \ldots, 10\}$. ♦

When representing a dataset, we use a labeled hypergraph where vertices are labeled with the descriptions (feature vectors) of the objects they represent and the hyperedges are labeled with the target values of the examples they represent.

Definition 4.8 (labeled hypergraph)**.** A labeled hypergraph is a 5-tuple $(G, \Sigma_V, \phi, \Sigma_E, \lambda)$ where G is a hypergraph, Σ_V is a vertex label alphabet, Σ_E is a hyperedge label alphabet, $\phi : V(G) \to \Sigma_V$ is a vertex labeling function and $\lambda : E(G) \to \Sigma_E$ is a hyperedge labeling function.

The two labeling functions defined above are used to assign features to every object (vertex) and to assign a target value to every example (hyperedge). Therefore, we use $\Sigma_V = \mathcal{X}$ and $\Sigma_E = \mathcal{Y}$.

Independence assumption

Though networked examples are not independent, we still need to assume some weaker form of independence of the examples. If we would not make any assumption, the dependence between examples could be so strong that they perfectly correlate (and hence are all identical). In such situation, it is not possible to generalize or learn.

The independence assumption we propose here is a relaxation of the classical i.i.d. assumption. While it is still not fully general and is not satisfied in all applications, we believe it is a useful first step. The idea we adopt here is that we explicitly model information shared by several examples and in this way also explicitly model the dependencies between the examples. We do not model the dependencies in detail and our analysis works for any possible dependency of the examples on the shared information.

We consider a labeled hypergraph $(G, \mathcal{X}, \phi, \mathcal{Y}, \lambda)$, where the labels assigned by ϕ and λ are drawn randomly from a probability distribution ρ. We make the following assumptions:

- The objects (and their features) assigned to vertices are independent from the hyperedges in which they participate, i.e., there is a function $\rho_{\mathcal{X}} : \mathcal{X} \mapsto [0,1]$ such that for every $q \in \mathcal{X}$ and $v \in V(G)$, $\rho(\phi(v) = q) = \rho_{\mathcal{X}}(q) = \rho_{\mathcal{X}}(\phi(v) = q \mid E(G))$.
- Moreover, every hyperedge (example) gets a target value drawn identically and independently given the features attached to the objects (vertices) incident with the hyperedge, i.e., there is a probability measure $\rho_{\mathcal{Y}|\mathbb{X}} : \mathcal{Y} \times \mathbb{X} \mapsto [0,1]$ such that for all $e \in E(G)$, $\rho(\lambda(e) = y \mid \phi|_e) = \rho_{\mathcal{Y}|\mathbb{X}}(y, \phi|_e) = \rho(\lambda(e) = y \mid \phi, E(G))$. Here, $\phi|_e$ is ϕ restricted to e, i.e., $\phi|_e = \{(v, \phi(v)) \mid v \in e\}$. Even if the hyperedges share vertices, still their target values are sampled i.i.d. from $\rho_{\mathcal{Y}|\mathbb{X}}$ based on their (possibly identical) feature vector.
- One can choose freely which vertices participate in which hyperedges, and which hyperedges belong to the training set and the test set, as long as this hyperedge and training set selection process is completely independent from the drawing of objects for the vertices and the drawing of target values for the hyperedges.

From the above assumptions, we can infer that

$$\rho(\phi, \lambda) = \prod_{v \in V(G)} \rho_{\mathcal{X}}(\phi(v)) \prod_{e \in E(G)} \rho_{\mathcal{Y}|\mathbb{X}}(\lambda(e), \phi|_e).$$

Our analysis holds no matter what the distribution ρ is, as long as the above assumptions are met.

It is possible that the empirical distribution of the training and/or test set deviates from ρ, but we show that we can bound the extent to which this is possible based on the assumptions.

As a special case, we often consider a k-partite setting. We can see this is a special case as follows. Let $(G, \mathcal{X}, \phi, \mathcal{Y}, \lambda)$ be a labeled k-partite hypergraph with $V = \cup_{i=1}^{k} V^{(i)}$ the vertex partition and $\mathcal{X} = \cup_{i=1}^{k} \mathcal{X}^{(i)}$ the feature space partition. Let $\mathbb{X} = \mathcal{X}^{([k])} = \mathcal{X}^{(1)} \times \ldots \times \mathcal{X}^{(k)}$ be the cartesian product of k feature spaces. Then, writing $\rho_{\mathcal{Y}|\mathbb{X}}$ as $\rho_{\mathcal{Y}|\mathbb{X}}(y, \phi|_e) = f(y, \phi(e \cap V^{(1)}), \ldots, \phi(e \cap V^{(k)}))$ for some function $f : \mathcal{Y} \times \mathcal{X}^{([k])} \mapsto [0,1]$ ensures that for vertices of $V^{(i)}$ only the part of the features from $\mathcal{X}^{(i)}$ is relevant. Learning $\rho_{\mathcal{Y}|\mathbb{X}}(y, \phi|_e)$ is then equivalent to assigning (independently) to every vertex of $V^{(i)}$ a set of features from $\mathcal{X}^{(i)}$ and learning the function $f : \mathcal{X}^{([k])} \mapsto \mathcal{Y}$.

These assumptions may not yet hold in all real-world situations but are already a better approximation than the classic i.i.d. assumptions. It may be instructive to consider real-world situations where our assumptions are satisfied and variants where they are not.

Example 4.9 (movie rating)**.** In our movie rating example, it may or may not be realistic that these assumptions hold. In particular, if ratings are obtained from visitors of a cinema, then probably some visitors will already have a preference and will not choose movies randomly. On the other hand, if ratings are obtained during a sneak preview, experiment or movie contest where a number of participants or jury members are asked to watch a specific list of movies, one could randomize the movies to increase fairness. In this way our assumptions would be satisfied. ⧫

Example 4.10 (court)**.** In our court example, if every litigant is assisted by a pro-deo lawyer and pro-deo lawyers and judges are randomly assigned to cases (e.g., for reasons of fairness), then our assumption holds. Else, if litigants can choose their prefered lawyers or judges specialize on particular topics, then our independence assumption may not hold. ⧫

Generating synthetic data

In many machine learning studies, synthetic data is used to evaluate approaches or testing hypotheses. Therefore, it is useful to investigate how one can generate synthetic data following a specified model. This is straightforward for i.i.d. data as one can draw each example independently from a fixed distribution. In the most general case generating data is much more difficult, e.g., in Markov logic networks [52] and exponential random graph models [53] sampling data is nontrivial.

Generating data according to the assumptions explained above is reasonably straightforward. One approach consists of the following steps:

- Choose any hypergraph G whose hyperedges determine the examples. The choice is completely free, and can be inspired by the type of experiments one intends to conduct, e.g., a powerlaw graph or other network satisfying topological properties of the targeted real-world application, or experimental designs (see also Example 5.8).
- Randomly sample the features for each vertex in this hypergraph according to the distribution $\rho_{\mathcal{X}}$ which also can be freely chosen. If G is a k-partite hypergraph, the distributions can be different for different partitions of vertices.
- Randomly sample the label for each hyperedge according to the features of vertices inside this hyperedge and the distribution $\rho_{\mathcal{Y}|\mathbb{X}}$ encoding the concept to be learned. One can choose this distribution freely.

A relation to learning probabilistic logical models

The problem described above is also equivalent to one of the fundamental tasks faced when learning directed probabilistic models such as Probabilistic Relational Models [25], Logical Bayesian Networks [23], Relational Bayesian Networks [35] and other directed models in the field of Statistical Relational Learning [27], namely learning the conditional probability function of a dependency template.

For instance, a dependency template may state that the rating a person gives to a movie depends on the interests of the person, the genre of the movie, the production cost of the movie and the size of the cinema theatre. A classic logic-based notation for such template(s) is

$$\begin{array}{lcl} rating(M,P,V) & | & interest(P) \\ rating(M,P,V) & | & genre(M) \\ rating(M,P,V) & | & cost(M) \\ rating(M,P,V) & | & size(V) \end{array}$$

The semantics of such template is that for every instantiation (also called grounding) of the template (i.e., substitution of the variable M with a movie, of P with a person and of V with a cinema) the corresponding conditional probability function describes the dependency of the random variable representing the rating on the random variables representing the interests, genre, cost and size.

When we consider all ground dependencies, we get a ground Bayesian network, as illustrated in Fig. 4.1b.

A classic approach to learn such conditional probability function is to collect a training set where every example consists of the features of a specific grounding and the corresponding target value. Then, a classifier is learned using a supervised learning algorithm, e.g., a decision tree learner. However, often the grounding of the template, the training examples share common information (properties of persons, movies, cinemas), and in fact the problem is equivalent to the problem of learning from networked examples introduced above. Therefore, to the extent our independence assumptions better capture the domain than the i.i.d. assumptions made by the classic algorithms, we expect it will be beneficial for the accuracy of the models to plug in one of the weighting schemes discussed in the next section. In fact, it is an advantage of our proposal that the example weighting schemes can be easily applied to allmost any existing supervised learning algorithm.

4.2 Mean estimation

Let $f : \mathbb{X} \mapsto \mathbb{R}$ be a real-valued function on some space $\mathbb{X}$. A fundamental problem is to estimate the expected value $\mu_f = \mathbb{E}_x[f(x)]$. For instance, the solution to many learning problems can be written in terms of such expected values of functions of the distribution of examples.

Example 4.11. Consider the problem of linear regression. We are given a set of examples $\{(x_i, y_i)\}_{i=1}^n$ drawn identically and independently from some fixed but unknown distribution D which first draws an x_i from a first distribution, then an r_i from a second zero-mean distribution and then computes $y_i = \alpha + \beta x_i + r_i$ for some fixed but unknown parameters α and β. We can recover these parameters α and β from expected values of functions applied to the data pairs by

$$\beta = \frac{\mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y]}{\mathbb{E}[x^2] - (\mathbb{E}[x])^2} \text{ and } \alpha = \mathbb{E}[y] - \frac{\mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y]}{\mathbb{E}[x^2] - (\mathbb{E}[x])^2}\mathbb{E}[x]$$

In this case, the parameter values of the model are combinations of the expected values of the functions x, y, x^2, y^2 and xy. One can approximate these expected values from the sample $\{(x_i, y_i)\}_{i=1}^n$, e.g., approximating $\mathbb{E}[x]$ with $\sum x_i/n$. ♦

Example 4.12. Similarly, in the context of learning decision trees, the frequencies of (a suitable subset of) all itemsets form sufficient statistics (see e.g., [48]). Such frequency of an itemset Z is equal to the expected value of the indicator function which returns 1 if Z is contained in a given example and 0 otherwise. ♦

One typically approximates $\mu_f = \mathbb{E}_x[f(x)]$ by averaging the value of f over an independent sample of the distribution, in particular, given a set $\{(x_i, y_i)\}_{i=1}^n$ of instances x_i drawn i.i.d. from some distribution $\rho_{\mathbb{X}}$, we could estimate

$$\hat{\mu}_f = \frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{4.1}$$

In order to study the networked setting, we first define networked random variables.

Definition 4.13 (G-networked random variables)**.** Given a hypergraph $G = (V, E)$, we call $(\xi_i)_{i=1}^n$ G-networked random variables where $n = |E|$ if there exist a distribution $\rho_{\mathcal{X}}$ on some feature space $\mathcal{X}$ and a function f mapping multisets of elements of $\mathcal{X}$ on real numbers such that $\xi_i = f(\{\Phi_v \mid v \in e_i\})$ and $\mathbb{E}[\xi_i] = \mu_f$ where $\{\Phi_v\}_{v \in V}$ is a set of independent $\rho_{\mathcal{X}}$-distributed random variables indexed by the vertices of G.

Note that in the definition above, the random variables are completely determined by the features of hyperedges. This definition can be generalized to the case that networked random variables also depend on the labels of hyperedges and some random factors.

According to the definition of G-networked random variables, we do not have an independent sample but rather a set of vertices V with independently drawn features and a fixed hyperedge structure E. In such case, what is the optimal way to combine the observations to get a good estimate? As we will see, the answer depends on the criterion one chooses to measure the quality of the estimate. Two popular quality criteria are variance and concentration bounds. For an unbiased estimator $\hat{\mu}$ of μ, i.e., an estimator $\hat{\mu}$ for which $\mathbb{E}[\hat{\mu}] = \mu$, the variance is defined as

$$\mathrm{var}(\mu) = \mathbb{E}\left[(\hat{\mu} - \mu)^2\right] \tag{4.2}$$

A concentration bound on $\hat{\mu} - \mu$ is a statement of the form

$$\forall \epsilon > 0, \Pr\left(|\hat{\mu} - \mu| \geq \epsilon\right) \leq \delta(\epsilon) \tag{4.3}$$

where δ is a (typically monotonically decreasing) function mapping positive reals on positive reals.

We limit ourselves to weighted average estimators

$$\hat{\mu}_f = \frac{\sum_{i=1}^{n} w_i f(x_i)}{\sum_{i=1}^{n} w_i}.$$

In the i.i.d. case, Formula (4.1), which is the above formula with uniform weights, is normally optimal. For the networked case, we study more alternatives:

- EQW: all examples get EQual Weights, i.e., for all i, $w_i = 1$.
- IND: a maximum-size set $E_{IND} \subseteq E$ of pairwise disjoint examples is selected, i.e., $\forall e_1, e_2 \in E_{IND} : e_1 \cap e_2 = \emptyset$. Examples in E_{IND} get weight 1, examples not in E_{IND} get weight 0.
- MinVar: a weighting scheme that improves the variance of estimators.
- FMN: a weighting scheme that improves the concentration bound guarantee of the EQW weighting scheme.

We first introduce the MinVar estimator in this chapter and discuss the FMN estimator in the next chapter.

4.3 Networked variance minimization

In this section, we analyze the variance of the statistics on k-partite networks (tuple networked examples and $\mathbb{X} = \mathcal{X}^{([k])}$). This analysis will result in a convex quadratically constrained linear program that minimizes the variance of the worst case. This shows that, for a weighted average, one can compute the weights that minimize its variance in an efficient way.

First, we define a decomposition of functions defined on the hyperedges. This decomposition comes from the analysis of variance (ANOVA, see e.g., [55]) that partitions the variance in a particular variable into components attributable to different sources of variation. This technique is also called Hoeffding's decomposition (see e.g., [49]).

Given a k dimensional vector space $\mathcal{X}^{([k])} = \times_{i=1}^{k} \mathcal{X}^{(i)}$ and an index set $S \subseteq [k]$, we define $\mathcal{X}^{(S)} := \times_{i \in S} \mathcal{X}^{(i)}$ (where the Cartesian product is taken in increasing order of i). For a vector $x \in \mathcal{X}^{([k])}$, $x^{(i)}$ is the projection of x on its i-th component and $x^{(S)} = \times_{i \in S} x^{(i)}$.

Consider a product distribution $\rho_{\mathbb{X}}$ on $\mathcal{X}^{([k])}$. We denote the marginal distribution over $\mathcal{X}^{(S)}$ with $\rho^{(S)}$, i.e., for $x \in \mathcal{X}^{([k])}$, $\rho^{(S)}(x^{(S)}) = \prod_{i \in S} \rho^{(i)}(x^{(i)})$. Let f be a function defined on $\mathcal{X}^{([k])}$, and let $x \in \mathcal{X}^{([k])}$. We first define $\mu = \mu_{\emptyset}(x) = \mathbb{E}_{x \sim \rho_{\mathbb{X}}}[f(x)]$, and then recursively define for every non-empty set $S \subseteq [k]$,

$$\mu_S\left(x^{(S)}\right) := \mathbb{E}_{x^{([k]\setminus S)} \sim \rho^{([k]\setminus S)}}\left[f(x) | x^{(S)}\right] - \sum_{T \subset S} \mu_T\left(x^{(T)}\right). \tag{4.4}$$

Example 4.14. Consider a function $f_{ex} : \mathcal{X}^{(1)} \times \mathcal{X}^{(2)} \times \mathcal{X}^{(3)} \mapsto \mathbb{R}$ where $\mathcal{X}^{(1)} = \mathcal{X}^{(2)} = \{1, 2\}$, $\mathcal{X}^{(3)} = \{1, 2, 3\}$. The values of f_{ex} are given in Table 4.1. Suppose that (x_1, x_2, x_3) is uniformly distributed over $\mathcal{X}^{(1)} \times \mathcal{X}^{(2)} \times \mathcal{X}^{(3)}$. Then, Table 4.2 gives $\mu_S\left(x^{(S)}\right)$ for all S and $x^{(S)}$. For example, we can compute:

$$\mu = \sum_{x_1, x_2, x_3} f_{ex}(x_1, x_2, x_3)/12 = 3/4$$

$$\mu_{\{1\}}(1) = \sum_{x_2, x_3} f_{ex}(1, x_2, x_3)/6 - \mu = 1/12$$

♦

From Eq. (4.4), we can easily derive that

$$f(x) = \sum_{S \subseteq [k]} \mu_S\left(x^{(S)}\right). \tag{4.5}$$

	$x_1 = 1$ $x_2 = 1$	$x_1 = 1$ $x_2 = 2$	$x_1 = 2$ $x_2 = 1$	$x_1 = 2$ $x_2 = 2$
$x_3 = 1$	1	1	0	1
$x_3 = 2$	0	1	1	1
$x_3 = 3$	1	1	1	0

Table 4.1: Function values of f_{ex} for Example 4.14

μ	$3/4$	$\mu_{\{2,3\}}(1,1)$	$-1/6$
$\mu_{\{1\}}(1)$	$1/12$	$\mu_{\{2,3\}}(1,2)$	$-1/6$
$\mu_{\{1\}}(2)$	$-1/12$	$\mu_{\{2,3\}}(1,3)$	$1/3$
$\mu_{\{2\}}(1)$	$-1/12$	$\mu_{\{2,3\}}(2,1)$	$1/6$
$\mu_{\{2\}}(2)$	$1/12$	$\mu_{\{2,3\}}(2,2)$	$1/6$
$\mu_{\{3\}}(1)$	0	$\mu_{\{2,3\}}(2,3)$	$-1/3$
$\mu_{\{3\}}(2)$	0	$\mu_{\{1,2,3\}}(1,1,1)$	$1/3$
$\mu_{\{3\}}(3)$	0	$\mu_{\{1,2,3\}}(1,1,2)$	$-1/6$
$\mu_{\{1,2\}}(1,1)$	$-1/12$	$\mu_{\{1,2,3\}}(1,1,3)$	$-1/6$
$\mu_{\{1,2\}}(1,2)$	$1/12$	$\mu_{\{1,2,3\}}(1,2,1)$	$-1/3$
$\mu_{\{1,2\}}(2,1)$	$1/12$	$\mu_{\{1,2,3\}}(1,2,2)$	$1/6$
$\mu_{\{1,2\}}(2,2)$	$-1/12$	$\mu_{\{1,2,3\}}(1,2,3)$	$1/6$
$\mu_{\{1,3\}}(1,1)$	$1/6$	$\mu_{\{1,2,3\}}(2,1,1)$	$-1/3$
$\mu_{\{1,3\}}(1,2)$	$-1/3$	$\mu_{\{1,2,3\}}(2,1,2)$	$1/6$
$\mu_{\{1,3\}}(1,3)$	$1/6$	$\mu_{\{1,2,3\}}(2,1,3)$	$1/6$
$\mu_{\{1,3\}}(2,1)$	$-1/6$	$\mu_{\{1,2,3\}}(2,2,1)$	$1/3$
$\mu_{\{1,3\}}(2,2)$	$1/3$	$\mu_{\{1,2,3\}}(2,2,2)$	$-1/6$
$\mu_{\{1,3\}}(2,3)$	$-1/6$	$\mu_{\{1,2,3\}}(2,2,3)$	$-1/6$

Table 4.2: μ_S values for Example 4.14

We now review some properties of this Hoeffding decomposition. The proof of the following lemmas can be found in Appendix A.1.

Lemma 4.15. For any non-empty $S \subseteq [k]$, μ_S is zero-mean for every dimension, i.e., for all $i \in S$, $\mathbb{E}_{x^{(i)} \sim \rho^{(i)}} \left[\mu_S \left(x^{(S)} \right) \right] = 0$.

Lemma 4.16. For any $S \neq T$, the functions μ_S and μ_T are uncorrelated (orthogonal), i.e., they have zero covariance or $\operatorname{cov}(\mu_S, \mu_T) = \mathbb{E}\left[\mu_S \left(x^{(S)} \right) \mu_T \left(x^{(T)} \right) \right] = 0$.

Example 4.17. Consider again the function f_{ex} and the distribution of (x_1, x_2, x_3) in Example 4.14. We can verify that

$$\begin{aligned}
\mathbb{E}_{x^{(1)}}\left[\mu_{\{1\}}(x^{(1)})\right] &= \frac{1}{2}\left(\mu_{\{1\}}(1) + \mu_{\{1\}}(2)\right) = \frac{1}{2}\left(\frac{1}{12} - \frac{1}{12}\right) = 0 \\
\mathbb{E}_{x^{(3)}}\left[\mu_{\{1,3\}}(1, x^{(3)})\right] &= \frac{1}{3}\left(\mu_{\{1,3\}}(1,1) + \mu_{\{1,3\}}(1,2) + \mu_{\{1,3\}}(1,3)\right) \\
&= \frac{1}{3}\left(\frac{1}{6} - \frac{1}{3} + \frac{1}{6}\right) = 0
\end{aligned}$$

♦

Lemma 4.18 shows that the variance of a function can be decomposed into $2^k - 1$ values σ_T^2 which we call variance components.

Lemma 4.18. The variance of the function f is the sum of the variances of μ_S of all S, i.e., $\sigma^2 = \sum_{S\subseteq[k]} \sigma_S^2 - \mu^2 = \sum_{S\neq\emptyset\wedge S\subseteq[k]} \sigma_S^2$ where $\sigma^2 = \mathbb{E}_{x\sim\rho_{\mathbb{X}}}\left[(f(x) - \mu)^2\right]$ and $\sigma_S^2 = \mathbb{E}_{x^{(S)}\sim\rho^{(S)}}\left[\mu_S^2\left(x^{(S)}\right)\right]$.

Example 4.19. Consider again the function and distribution in Example 4.14. We can calculate the variance $\sigma^2 = 3/16$ and the variance components, e.g., $\sigma_{\{1,3\}}^2 = \frac{1}{6}(\mu_{\{1,3\}}^2(1,1) + \mu_{\{1,3\}}^2(1,2) + \mu_{\{1,3\}}^2(1,3) + \mu_{\{1,3\}}^2(2,1) + \mu_{\{1,3\}}^2(2,2) + \mu_{\{1,3\}}^2(2,3)) = \frac{1}{18}$. All the variance components are listed in Table 4.3. Then, we can check that $\sigma^2 = \sum_{T\subseteq[3]\wedge T\neq\emptyset} \sigma_T^2$. ♦

T	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$	Sum
σ_T^2	1/144	1/144	0	1/144	1/18	1/18	1/18	3/16

Table 4.3: Variance components in Example 4.19

We now analyze this variance decomposition in the context of networked examples. First, we introduce the concept of overlap index matrix.

Definition 4.20 (overlap index matrix)**.** For a vector of examples $E = \{e_i\}_{i=1}^n$, we define the overlap matrix of E, denoted J^E to be the $n \times n$ matrix with

$$J_{i,j}^E = \{l \in [k] \mid e_i^{(l)} = e_j^{(l)}\}.$$

Example 4.21. Consider the hypergraph in Fig. 4.1a. Its overlap index matrix is

$$\begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} \{1,2,3\} & \{1\} & \{1\} & \emptyset & \emptyset \\ \{1\} & \{1,2,3\} & \{1,2\} & \emptyset & \emptyset \\ \{1\} & \{1,2\} & \{1,2,3\} & \{3\} & \{3\} \\ \emptyset & \emptyset & \{3\} & \{1,2,3\} & \{3\} \\ \emptyset & \emptyset & \{3\} & \{3\} & \{1,2,3\} \end{pmatrix} \end{array}$$

♦

According to the definitions and properties above, and according to our assumption that vertex features are drawn i.i.d., for two examples (hyperedges) e_i and e_j the covariance of $f(\phi(e_i))$ and $f(\phi(e_j))$ is $\operatorname{cov}(f(\phi(e_i)), f(\phi(e_j))) = \sum_{T \subseteq J^E_{i,j}} \sigma_T^2$.

Let $E = \{e_i\}_{i=1}^n$ be a set of examples in a k-partite hypergraph. Let $F = (f(\phi(e_i)))_{i=1}^n$ be the vector of function values on the n hyperedges. Then, let $\Sigma = \operatorname{var}(F)$ be the covariance matrix of these function values, i.e., $\Sigma_{i,j} = \operatorname{cov}(f(\phi(e_i)), f(\phi(e_j))) = \sum_{T \subseteq J^E_{i,j}} \sigma_T^2$.

For a weight vector $w \in [0,1]^n$ with $\|w\|_1 = \sum_{i=1}^n w_i = 1$, the weighted sum $w^\top F = \sum_{i=1}^n w_i f(\phi(e_i))$ approximates $\mu = \mathbb{E}_{x \sim \rho_{\mathbb{X}}}[f(x)]$, as $\mathbb{E}\left[w^\top F\right] = \sum_{i=1}^n w_i \mathbb{E}_{x \sim \rho_{\mathbb{X}}}[f(x)] = \mu$. The variance of this weighted sum estimate, which we denote $\sigma_E\left(w, \{\sigma_T\}_{T \subseteq [k]}\right)$, is

$$\sigma_E\left(w, \{\sigma_T\}_{T \subseteq [k]}\right) = \operatorname{var}(w^\top F) = w^\top \Sigma w = \sum_{i \in [n]} \sum_{j \in [n]} \left(w_i w_j \sum_{T \in J^E_{i,j}} \sigma_T^2 \right).$$

When estimating the mean of a distribution, we usually prefer an estimator with a variance which is as small as possible. Given a k-partite hypergraph, we can see that the variance of a networked sample not only depends on the weights but also on the 2^k values of σ_T, one for each $T \subseteq [k]$. In practice, we usually do not know the values of the σ_T. Still, if the variance σ^2 of the function f (not the variance we intend to minimize) is a constant, i.e., the sum $\sum_{T \subseteq [k]} \sigma_T^2 = \sigma^2$ is a constant, then for every weight vector w one can find a tight upper bound for $\operatorname{var}(w^\top F)$ by maximizing $w^\top \Sigma w$ as a function of the variance components $\{\sigma_T\}_{T \subseteq [k]}$. We call the variance of this type the worst-case variance. An interesting question is now for what weight vector w the worst-case variance is minimal. We define the following game, played by a player MIN who attempts to minimize variance by choosing the weight vector

w and a player MAX who attempts to maximize the variance by choosing the random variable and hence the decomposition of its variance.

Definition 4.22 (MWCWSV game)**.** A minimum worst-case weighted-sum variance game (MWCWSV) is a game parameterized by an overlap index matrix J^E, with a player MIN who can choose a vector w satisfying

$$\sum_{i\in[n]} w_i = 1 \tag{4.6}$$

$$\forall i : w_i \geq 0 \tag{4.7}$$

and a player MAX who can choose $(\sigma_T)_{T\subseteq[k]}$ satisfying

$$\sum_{T\subseteq[k]} \sigma_T^2 = \sigma^2, \tag{4.8}$$

and with payoff function

$$\sigma_E\left(w, \{\sigma_T\}_{T\subseteq[k]}\right).$$

which MIN tries to minimize and MAX tries to maximize.

Hence, finding the optimal strategy of MIN is equivalent to computing

$$\min_w \max_{\{\sigma_T^2 : T\subseteq[k]\}} \sigma_E\left(w, \{\sigma_T\}_{T\subseteq[k]}\right)$$

subject to the constraints (4.6), (4.7) and (4.8).

Example 4.23. Let us consider the hypergraph in Fig. 4.1a again. The payoff function σ_E is

$$w^\top \begin{pmatrix} \sigma^2 & \sigma^2_{\{1\}} & \sigma^2_{\{1\}} & 0 & 0 \\ \sigma^2_{\{1\}} & \sigma^2 & \sigma^2_{\{1\}} + \sigma^2_{\{2\}} + \sigma^2_{\{1,2\}} & 0 & 0 \\ \sigma^2_{\{1\}} & \sigma^2_{\{1\}} + \sigma^2_{\{2\}} + \sigma^2_{\{1,2\}} & \sigma^2 & \sigma^2_{\{3\}} & 0 \\ 0 & 0 & \sigma^2_{\{3\}} & \sigma^2 & \sigma^2_{\{3\}} \\ 0 & 0 & \sigma^2_{\{3\}} & \sigma^2_{\{3\}} & \sigma^2 \end{pmatrix} w.$$

♦

We can see that the payoff function σ_E is convex in its first argument w and linear in its second argument $\{\sigma_T\}_{T\subseteq[k]}$. By Sion's minimax theorem [60], we get

$$\min_w \max_{\{\sigma_T^2 : T\subseteq[k]\}} \sigma_E\left(w, \{\sigma_T\}_{T\subseteq[k]}\right) = \max_{\{\sigma_T^2 : T\subseteq[k]\}} \min_w \sigma_E\left(w, \{\sigma_T\}_{T\subseteq[k]}\right)$$

which ensures that there exists an equilibrium (saddle point).

We now transform this game into an equivalent convex quadratically constrained linear program which can be efficiently solved using standard methods [6].

Lemma 4.24. For any MWCWSV game, there exists a saddle point $\left(w^*, \{\sigma_T^*\}_{T\subseteq[k]}\right)$ such that $\forall T \subseteq [k], |T| \geq 2 \Rightarrow \sigma_T^* = 0$.

Proof. Among the saddle points of the MWCWSV problem, let $\left(w^*, \{\sigma_T^*\}_{T\subseteq[k]}\right)$ be one of those minimizing the number of sets $T \subseteq [k]$ for which $|T| \geq 2$ and $\sigma_T^* \neq 0$. We prove the lemma by showing that if there is at least one $T \subseteq [k]$ for which $|T| \geq 2$ and $\sigma_T^* \neq 0$, then the solution is not optimal or the number of $T \subseteq [k]$ for which $|T| \geq 2$ and $\sigma_T^* \neq 0$ is not minimal.

Suppose that there is a specific $U \subseteq [k]$ such that $|U| \geq 2$ and $\sigma_U^* \neq 0$. Then, select an arbitrary $l \in U$ and define the weight vector w' and variance components $\{\sigma'_T\}_{T\subseteq[k]}$ as follows: (i) $w' = w^*$, (ii) $\sigma'_U = 0$, (iii) $\left(\sigma'_{\{l\}}\right)^2 = \left(\sigma^*_{\{l\}}\right)^2 + \left(\sigma_U^*\right)^2$ and (iv) for all $T \in 2^{[k]} \setminus \{U, \{l\}\}$, $\sigma'_T = \sigma_T^*$. We can see that $\left(w', \{\sigma'_T\}_{T\subseteq[k]}\right)$ is a feasible solution, because $\|w'\|_1 = \|w^*\|_1 = 1$, $\forall i : w'_i = w_i^* \geq 0$ and $\sum \left(\sigma'_T\right)^2 = \sum \left(\sigma_T^*\right)^2 = \sigma^2$. Moreover, we have

$$
\begin{aligned}
&\left(w'\right)^\top \Sigma' w' - \left(w^*\right)^\top \Sigma^* w^* \\
&= \left(\left(\sigma'_{\{l\}}\right)^2 - \left(\sigma^*_{\{l\}}\right)^2\right) \sum \{w_i w_j \mid l \in J^E_{i,j}\} \\
&\quad + \left(\left(\sigma'_U\right)^2 - \left(\sigma_U^*\right)^2\right) \sum \{w_i w_j \mid U \subseteq J^E_{i,j}\} \\
&= \left(\sigma_U^*\right)^2 \left(\sum \{w_i w_j \mid l \in J^E_{i,j}\} - \sum \{w_i w_j \mid U \subseteq J^E_{i,j}\}\right) \\
&= \left(\sigma_U^*\right)^2 \sum \{w_i w_j \mid l \in J^E_{i,j} \wedge U \not\subseteq J^E_{i,j}\} \\
&\geq 0
\end{aligned}
$$

Hence, $\left(w', \{\sigma'_T\}_{T\subseteq[k]}\right)$ is also a saddle point. The lemma follows by considering that the number of sets T for which $|T| \geq 2$ and $\sigma'_T \neq 0$ is smaller than the number of T for which $|T| \geq 2$ and $\sigma_T^* \neq 0$, leading to the announced contradiction. □

Example 4.25. According to Lemma 4.24, the payoff function σ_E in Example 4.23 can be simplified as

$$w^\top \begin{pmatrix} \sigma^2 & \sigma^2_{\{1\}} & \sigma^2_{\{1\}} & 0 & 0 \\ \sigma^2_{\{1\}} & \sigma^2 & \sigma^2_{\{1\}} + \sigma^2_{\{2\}} & 0 & 0 \\ \sigma^2_{\{1\}} & \sigma^2_{\{1\}} + \sigma^2_{\{2\}} & \sigma^2 & \sigma^2_{\{3\}} & 0 \\ 0 & 0 & \sigma^2_{\{3\}} & \sigma^2 & \sigma^2_{\{3\}} \\ 0 & 0 & \sigma^2_{\{3\}} & \sigma^2_{\{3\}} & \sigma^2 \end{pmatrix} w.$$

♦

Therefore, for any MWCWSV game, the solution of the following simplified game can be extended into a solution of the original MWCWSV game:

$$\begin{aligned} \min_w \max_{\{\sigma_l^2 : l \in [k]\}} \quad & \sum_{i\in[n]} \sum_{j\in[n]} w_i w_j \sum_{l \in J^E_{i,j}} \sigma_l^2 \\ \text{s.t.} \quad & \sum_{l\in[k]} \sigma_l^2 = \sigma^2 \\ & \sum_{i\in[n]} w_i = 1 \\ & \forall i : w_i \geq 0 \end{aligned}$$

For a fixed w, the inner part

$$\begin{aligned} \max_{\{\sigma_l^2 : l \in [k]\}} \quad & \sum_{i\in[n]} \sum_{j\in[n]} w_i w_j \sum_{l \in J^E_{i,j}} \sigma_l^2 \\ \text{s.t.} \quad & \sum_{l\in[k]} \sigma_l^2 = \sigma^2 \end{aligned}$$

is a linear program with decision variables $\{\sigma_l^2\}_{l\in[k]}$, so it reaches the maximum value when $\sigma_l = \sigma$ for some l (and $\sigma_{l'} = 0$ for all $l' \neq l$). Then, the inner part is equivalent to:

$$\max_{l\in[k]} \sum_{i\in[n]} \sum_{j\in[n]} w_i w_j \sigma^2 I(l \in J^E_{i,j})$$

where I is the indicator function.

Introducing a new decision variable t, one can rewrite the whole optimization problem as follows:

$$\begin{aligned} \min_{w;t} \ & t \\ \text{s.t. } & \forall l \in [k] : \sum_{i\in[n]} \sum_{j\in[n]} w_i w_j \sigma^2 I(l \in J^E_{i,j}) \leq t \\ & \sum_i w_i = 1 \\ & \forall i : w_i \geq 0 \end{aligned} \tag{4.9}$$

An optimal solution of this problem can be extended into a saddle point of the original MWCWSV game by choosing some l for which $\sum_{i\in[n]} \sum_{j\in[n]} w_i w_j \sigma^2 I(l \in J^E_{i,j}) = t$, and setting $\sigma_l = \sigma$ and $\sigma_T = 0$ for all $T \in 2^{[k]} \setminus \{\{l\}\}$. Conversely, it is straightforward to see that any optimal strategy of MIN can be mapped to an optimal solution of program (4.9). Therefore, there is a one-to-one mapping between optimal strategies of MIN and solutions of program (4.9).

Example 4.26. Continuing on Example 4.25, the final program is

$$\begin{aligned} \min_{w;t} \ & t \\ \text{s.t. } & (w_1 + w_2)^2 + (w_3 + w_4)^2 + w_5^2 \leq t \\ & w_1^2 + (w_2 + w_3)^2 + (w_4 + w_5)^2 \leq t \\ & (w_1 + w_5)^2 + w_2^2 + w_3^2 + w_4^2 \leq t \\ & w_1 + w_2 + w_3 + w_4 + w_5 + w_6 = 1 \\ & w_1, w_2, w_3, w_4, w_5, w_6 \geq 0 \end{aligned}$$

♦

4.4 Summary

In this chapter, we have introduced networked examples, networked random variables and a weaker independence assumption. These networked examples and networked random variables can be represented by hypergraphs.

We proposed the MinVar weighting scheme which is optimal from the worst-case variance point of view.

In the future, we want to consider independence assumptions that are more general than those investigated here. A first step in this direction would be to develop a measure to assess the strength of the dependency of the hyperedges on the features of the vertices.

Chapter 5

Concentration inequalities for networked random variables

To derive a generalization error bound in the next chapter, only a variance guarantee is not sufficient. One needs stronger concentration inequalities to show learnability results. For mutually independent random variables, there exists a class of concentration inequalities to bound $\Pr(\hat{\mu} - \mu \geq \epsilon)$ and $\Pr(|\hat{\mu} - \mu| \geq \epsilon)$, e.g., the Chernoff-Hoeffding concentration inequality. The assumption of random variables being mutually independent is pivotal in the proofs of these concentration inequalities.

In this chapter, we prove Chernoff-Hoeffding style concentration inequalities for weighted mean value estimators on networked random variables. We first discuss unweighted (or equally weighted) averages and Janson's inequalities which are Chernoff-Hoeffding style concentration inequalities for the EQW estimator on networked random variables. In order to prove a weighted version, we define the vertex-bounded weight vectors for hypergraphs. Based on this definition, we show a lemma related to moment generating functions. This lemma not only brings us Chernoff-Hoeffding style concentration inequalities for weighted mean value estimators on networked random variables, but also improves Janson's inequalities a bit.

According to these inequalities, we are able to minimize the bound if we properly choose a weight vector. This minimization can be done by solving a linear program as in Chapter 3.

As an example, we apply our concentration inequalities to U-statistics.

5.1 Unweighted averages and Janson's bound

In this section, we relate our work to Janson's result [36]. The definitions introduced below are only relevant for the discussion of Janson's result and are not required for the rest of the thesis.

Definition 5.1 (hyperedge-chromatic number)**.** The hyperedge-chromatic number χ of a hypergraph G is the smallest number of colors needed to color the hyperedges in E_G such that any two adjacent hyperedges have different colors.

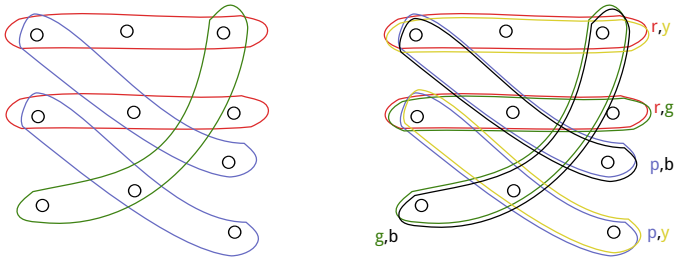
Definition 5.2 (b-fold hyperedge-coloring)**.** A b-fold hyperedge-coloring of a hypergraph G is an assignment of b colors to every hyperedge in E_G such that adjacent hyperedges have no color in common.

Definition 5.3 (b-fold hyperedge-chromatic number)**.** The b-fold hyperedge-chromatic number $\chi^{(b)}$ of a hypergraph G is the smallest number of colors needed to obtain a b-fold hyperedge-coloring of the hyperedges in E_G.

Note that, in the definition of hyperedge-chromatic numbers, we only do 1-fold hyperedge-coloring.

Definition 5.4 (Fractional hyperedge-chromatic number)**.** Let G be a hypergraph. The fractional hyperedge-chromatic number χ^* of G is

$$\chi^*_G = \inf_b \frac{\chi^{(b)}_G}{b}.$$



(a) Hyperedge coloring (b) 2-fold hyperedge coloring

Figure 5.1: An example for different hyperedge-chromatic numbers

Example 5.5. Consider the hypergraph in Fig. 5.1. Three colors are sufficient and necessary to color these hyperedges in such a way that no two incident

hyperedges have the same color (Fig. 5.1a). A 2-fold hyperedge-coloring of the hyperedges needs five colors (Fig. 5.1b). In fact, this 2-fold hyperedge-coloring also gives us the fractional hyperedge-chromatic number $\frac{5}{2}$. ♦

The following theorem, from [36], gives concentration bounds on the error between the expected value μ and the average of n networked samples ξ_i.

Theorem 5.6. *Let $(\xi_i)_{i=1}^n$ be G-networked random variables with mean $\mathbb{E}[\xi_i] = \mu$, variance $var(\xi_i) = \sigma^2$ and satisfying $|\xi_i - \mu| \leq M$. Then for all $\epsilon > 0$,*

$$\Pr\left(\frac{1}{n}\sum_{i=1}^{n}\xi_i - \mu \geq \epsilon\right) \leq \exp\left(-\frac{n\epsilon^2}{2\chi_G^* M^2}\right),$$

$$\Pr\left(\frac{1}{n}\sum_{i=1}^{n}\xi_i - \mu \geq \epsilon\right) \leq \exp\left(-\frac{8n\epsilon^2}{25\chi_G^*(\sigma^2 + M\epsilon/3)}\right),$$

where χ^* is the fractional hyperedge-chromatic number.

In some cases, the fractional hyperedge-chromatic number can be very large. One can always construct hypergraphs such that every pair of hyperedges intersects. In this case, the fractional hyperedge-chromatic number is equal to its hyperedge number, hence $\frac{n}{\chi_G^*} = 1$. In that way, the exponent does not decrease with sample size n and then the bounds in Theorem 5.6 do not decrease. As an example, we consider projective planes.

Definition 5.7 (projective plane)**.** For every integer $\kappa \geq 2$, a projective plane of order κ is a hypergraph G such that

- every pair of vertices determines one hyperedge
- every pair of hyperedges intersect at one vertex
- every hyperedge $e \in E_G$ contains $\kappa + 1$ vertices
- every vertex is contained in $\kappa + 1$ hyperedges

The last two properties together imply that $|E_G| = |V_G| = \kappa^2 + \kappa + 1$. Also, it is known that a projective plane of order κ exists when κ is a prime power [46]. Moreover, the fractional hyperedge-chromatic number of any subhypergraph of a projective plane is equal to its hyperedge number. Finally, a truncated projective plane (see e.g., [46]) of order κ, which is obtained from a projective plane of order κ by deleting a vertex v and the $\kappa + 1$ hyperedges incident to

v, is a $\kappa+1$-partite hypergraph; its fractional hyperedge-chromatic number is equal to κ^2, its hyperedge number.

So there are k-partite hypergraphs consisting of n ($n = O(k^2)$) tuple-networked examples for which $\frac{n}{\chi^*_G} = 1$. For such datasets, Janson's inequalities (Theorem 5.6) fail to offer useful bounds. A tighter bound will be derived in the next section (Corollary 5.16); it is significantly better in some datasets such as projective planes.

Projective planes and truncated projective planes are not only of theoretical interest. In fact, they are special cases of block designs as studied in the field of experimental design [16]. This field studies what points in a feature space to measure to maximize certain experimental objectives such as diversity and independence of training data.

Example 5.8. Consider the problem of solving a task by combining experts from four different disciplines D_1, D_2, D_3 and D_4. One may be interested in understanding how the quality of the result depends on the skills and points of view of the four experts. Suppose we can hire three experts from each of the four disciplines to perform a number of experiments. We denote the i-th expert ($i \in \{1,2,3\}$) of the j-th discipline with $v_{j,i}$. In each experiment, we choose one expert X_j from each discipline D_j ($j \in \{1,2,3,4\}$), and measure the function value $f(X_1, X_2, X_3, X_4)$. Measuring all $3^4 = 81$ combinations of values of $f : \{1,2,3\}^4 \to \mathbb{R}$ may be too expensive. We could therefore decide to measure only the 9 datapoints in Table 5.1 and then to fit a model.

Interesting properties are that (i) each feature-value pair $X_j = v_{j,i}$ (for

$(v_{1,1}, v_{2,1}, v_{3,1}, v_{4,1})$	$(v_{1,2}, v_{2,1}, v_{3,2}, v_{4,3})$	$(v_{1,3}, v_{2,1}, v_{3,3}, v_{4,2})$
$(v_{1,1}, v_{2,2}, v_{3,2}, v_{4,2})$	$(v_{1,2}, v_{2,2}, v_{3,3}, v_{4,1})$	$(v_{1,3}, v_{2,2}, v_{3,1}, v_{4,3})$
$(v_{1,1}, v_{2,3}, v_{3,3}, v_{4,3})$	$(v_{1,2}, v_{2,3}, v_{3,1}, v_{4,2})$	$(v_{1,3}, v_{2,3}, v_{3,2}, v_{4,1})$

Table 5.1: A truncated projective plane as experimental design.

$j \in \{1,2,3,4\}$ and $i \in \{1,2,3\}$) occurs in the same number (three) of examples and (ii) each combination of two feature-value pairs $X_j = v_{j,i}$ and $X_{j'} = v_{j',i'}$ (for $i, i' \in \{1,2,3\}$ and $j, j' \in \{1,2,3,4\}$ and $i \neq i'$) uniquely determines a datapoint and (iii) any two datapoints share exactly one feature-value pair. These properties are beneficial for avoiding bias in learned models. At the same time, when we view the experts $v_{j,i}$ as vertices and the datapoints as hyperedges, by definition, this dataset is a truncated projective plane of order 3. ♦

5.2 Concentration bounds for weighted networked random variables

In this section, we study concentration bounds of the form (4.3). Before stating the main result (Theorem 5.10), we first define vertex-bounded weight vectors and give the property whose generalization requires this concept.

A common key property used for proving basic exponential concentration inequalities is that all observations are independent. If $(\xi_i)_{i=1}^n$ are independent random variables, then the moment-generating function $\mathbb{E}[\exp(c\sum_{i=1}^n \xi_i)]$, where $c \in \mathbb{R}$, satisfies

$$\mathbb{E}\left[\exp\left(c\sum_{i=1}^{n}\xi_i\right)\right] = \prod_{i=1}^{n}\mathbb{E}\left[e^{c\xi_i}\right].$$

However, when considering networked random variables, the equality does not hold. Instead, we show a generalized property (Theorem 5.12). Based on this theorem, we derive exponential concentration inequalities. First, we define vertex-bounded weights of hypergraphs.

Definition 5.9 (vertex-bounded weights)**.** Given a hypergraph $G = (V, E)$ with $E = \{e_i\}_{i=1}^n$, a vertex-bounded weight vector w is a nonnegative vector $(w_i)_{i=1}^n$ defined on its hyperedges satisfying that for every vertex $v \in V$, $\sum_{i:v\in e_i} w_i \leq 1$. In other words, a weight vector is vertex-bounded if for every vertex the sum of the weights of the incident hyperedges is at most 1.

The following inequalities are networked analogues of the Bennett, Bernstein, Hoeffding inequalities (see e.g., Chapter 2 in [15]).

Theorem 5.10. *Let $(\xi_i)_{i=1}^n$ be G-networked random variables with mean $\mathbb{E}[\xi_i] = \mu$, variance $\sigma^2(\xi_i) = \sigma^2$, and satisfying $|\xi_i - \mu| \leq M$. Let w be a vertex-bounded weight vector of G and $|w| = \sum_{i=1}^n w_i$, then for all $\epsilon > 0$,*

$$\Pr\left(\frac{1}{|w|}\sum_{i=1}^{n} w_i\xi_i - \mu \geq \epsilon\right) \leq \exp\left(-\frac{|w|\epsilon}{2M}\log\left(1+\frac{M\epsilon}{\sigma^2}\right)\right), \tag{5.1}$$

$$\Pr\left(\frac{1}{|w|}\sum_{i=1}^{n} w_i\xi_i - \mu \geq \epsilon\right) \leq \exp\left(-\frac{|w|\epsilon^2}{2(\sigma^2+\frac{1}{3}M\epsilon)}\right), \text{ and} \tag{5.2}$$

$$\Pr\left(\frac{1}{|w|}\sum_{i=1}^{n} w_i\xi_i - \mu \geq \epsilon\right) \leq \exp\left(-\frac{|w|\epsilon^2}{2M^2}\right). \tag{5.3}$$

In order to prove this theorem, we first show some intermediate result.

Lemma 5.11. Let $\beta = (\beta_i)_{i=1}^k \in \mathbb{R}_+^k$ such that $\sum_{i=1}^k \beta_i \leq 1$. Then, the function $g(t)$ with $t = (t_i)_{i=1}^k \in \mathbb{R}_+^k$ defined by $g(t) = \prod_{i=1}^k t_i^{\beta_i}$, is concave.

The following theorem plays a very important role in proving Theorem 5.10, and the rest is essentially standard.

Theorem 5.12. *Given G-networked random variables $(\xi_i)_{i=1}^n$, if $w = (w_i)_{i=1}^n$ is a vertex-bounded weight vector of the hypergraph G, then*

$$\mathbb{E}\left[\exp\left(\sum_{i=1}^n w_i \xi_i\right)\right] \leq \prod_{i=1}^n \left(\mathbb{E}\left[e^{\xi_i}\right]\right)^{w_i}. \tag{5.4}$$

Proof. First, note that the expectation in the left hand side of inequality (5.4) is over the (independent) features $x_1, \ldots, x_m$ of the vertices of G, because these are the basic random variables of which the $(\xi_i)_{i=1}^n$ are composed. We prove this theorem by induction on $|V_G|$. For $|V_G| = 1$,

$$\mathbb{E}\left[\exp\left(\sum_{i=1}^n w_i \xi_i\right)\right] = \mathbb{E}_{x_1}\left[\prod_{i=1}^n e^{w_i \xi_i}\right].$$

Using Lemma 5.11 with $t = \left(e^{\xi_i}\right)_{i=1}^n$, $\beta = w$ and $g(t) = \prod_{i=1}^n e^{w_i \xi_i}$, we know that $g(t)$ is a concave function since w is a vertex-bounded weight vector. Given that $g(t)$ is concave, we have

$$\mathbb{E}\left[\exp\left(\sum_{i=1}^n w_i \xi_i\right)\right] = \mathbb{E}_{x_1}[g(t)] \leq g(\mathbb{E}_{x_1}[t]) = \prod_{i=1}^n \left(\mathbb{E}\left[e^{\xi_i}\right]\right)^{w_i}$$

which follows from Jensen's inequality [37]. Assume that the theorem is true for $|V_G| = 1, \ldots, m-1$, we now prove the theorem for $|V_G| = m$. We can write

$$\mathbb{E}\left[\exp\left(\sum_{i=1}^n w_i \xi_i\right)\right] = \mathbb{E}_{x_m}\left[\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[\prod_{i=1}^n e^{w_i \xi_i} \middle| x_m\right]\right]. \tag{5.5}$$

where the $\mathbb{E}[\cdot|\cdot]$ notation on the right hand side denotes a conditional expectation. We use the induction hypothesis on the right hand side of Eq. (5.5), yielding

$$\mathbb{E}_{x_m}\left[\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[\prod_{i=1}^n e^{w_i \xi_i} \middle| x_m\right]\right] \leq \mathbb{E}_{x_m}\left[\prod_{i=1}^n \left(\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[e^{\xi_i} \middle| x_m\right]\right)^{w_i}\right]. \tag{5.6}$$

We define two index sets A and B, partitioning hyperedges in G (and hence random variables ξ_i) into a part which is incident with v_m (dependent on x_m)

and a part which is not: $A := \{i | v_m \in e_i\}$ and $B := \{i | v_m \notin e_i\}$. Then, for all $i \in B$, ξ_i is independent of x_m. We can write this as

$$\mathbb{E}_{x_m}\left[\prod_{i=1}^{n}\left(\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[e^{\xi_i}\middle|x_m\right]\right)^{w_i}\right]$$

$$=\mathbb{E}_{x_m}\left[\prod_{i\in A}\left(\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[e^{\xi_i}\middle|x_m\right]\right)^{w_i}\right]\prod_{i\in B}\left(\mathbb{E}\left[e^{\xi_i}\right]\right)^{w_i}. \tag{5.7}$$

Let $t = \left(\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[e^{\xi_i}\middle|x_m\right]\right)_{i\in A}$, $\beta = (w_i)_{i\in A}$ and $g(t) = \prod_{i\in A}\left(\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[e^{\xi_i}\middle|x_m\right]\right)^{w_i}$. According to the definition of vertex-bounded weights and Lemma 5.11, we know that $g(t)$ is concave. Again, by Jensen's inequality, we have

$$\mathbb{E}_{x_m}\left[\prod_{i\in A}\left(\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[e^{\xi_i}\middle|x_m\right]\right)^{w_i}\right] \leq \prod_{i\in A}\left(\mathbb{E}_{x_m}\left[\mathbb{E}_{x_1,\ldots,x_{m-1}}\left[e^{\xi_i}\middle|x_m\right]\right]\right)^{w_i}$$

$$= \prod_{i\in A}\left(\mathbb{E}\left[e^{\xi_i}\right]\right)^{w_i}. \tag{5.8}$$

From Equations (5.5), (5.7) and Inequalities (5.6) and (5.8), we can see that this theorem is still true for $|V_G| = m$. □

Remark: Note that this theorem holds for any hypergraph (and its corresponding networked random variables). It therefore also holds for any k-partite hypergraph ($k \in \mathbb{Z}_+$).

Using Theorem 5.12, we are able to obtain exponential concentration inequalities of networked variables. The proofs of corresponding inequalities of independent random variables can be found in [17].

Theorem 5.13. *Let $(\xi_i)_{i=1}^n$ be G-networked random variables with mean $\mathbb{E}[\xi_i] = \mu$ and variance $\sigma^2(\xi_i) = \sigma^2$, such that $|\xi_i - \mu| \leq M$ with probability 1. Let $w = (w_i)_{i=1}^n$ be a vertex-bounded weight vector for G, and let $|w| = \sum_i w_i$, then for all $\epsilon > 0$,*

$$\Pr\left(\sum_i w_i\left(\xi_i - \mu\right) \geq \epsilon\right) \leq \exp\left(-\frac{|w|\sigma^2}{M^2}h\left(\frac{M\epsilon}{|w|\sigma^2}\right)\right)$$

where $h(a) = (1+a)\log(1+a) - a$ for any real number a.

Proof. Without loss of generality, we assume $\mu = 0$. Let c be an arbitrary positive constant which will be determined later. Then

$$I := \Pr\left(\sum_{i=1}^{n} w_i\xi_i \geq \epsilon\right) = \Pr\left(\exp\left(c\sum_{i=1}^{n} w_i\xi_i\right) \geq e^{c\epsilon}\right).$$

By Markov's inequality and Theorem 5.12, we have

$$I \leq e^{-c\epsilon}\mathbb{E}\left[\exp\left(c\sum_{i=1}^{n} w_i\xi_i\right)\right] \leq e^{-c\epsilon}\prod_i \left(\mathbb{E}\left[e^{c\xi_i}\right]\right)^{w_i}.$$

Since $|\xi_i| \leq M$ and $\mu = 0$, we have

$$\mathbb{E}\left[e^{c\xi_i}\right] = 1 + \sum_{p=2}^{+\infty}\frac{c^p\mathbb{E}[\xi_i^p]}{p!} \leq 1 + \sum_{p=2}^{+\infty}\frac{c^pM^{p-2}\sigma^2}{p!}$$

from the Taylor expansion for exponential functions. Using $1+a \leq e^a$, it follows that

$$\mathbb{E}\left[e^{c\xi_i}\right] \leq \exp\left(\sum_{p=2}^{+\infty}\frac{c^pM^{p-2}\sigma^2}{p!}\right) = \exp\left(\frac{e^{cM}-1-cM}{M^2}\sigma^2\right)$$

and therefore

$$I \leq \exp\left(-c\epsilon + \frac{e^{cM}-1-cM}{M^2}|w|\sigma^2\right).$$

Now choose the constant c to be the minimizer of the bound on the right hand side above:

$$c = \frac{1}{M}\log\left(1 + \frac{M\epsilon}{|w|\sigma^2}\right).$$

That is, $e^{cM} - 1 = \frac{M\epsilon}{|w|\sigma^2}$. With this choice,

$$I \leq \exp\left(-\frac{|w|\sigma^2}{M^2}h\left(\frac{M\epsilon}{|w|\sigma^2}\right)\right).$$

This proves the desired inequality. □

Theorem 5.14. *Let $(\xi_i)_{i=1}^n$ be G-networked random variables with mean $\mathbb{E}[\xi_i] = \mu$ and variance $\sigma^2(\xi_i) = \sigma^2$, such that $|\xi_i - \mu| \leq M$. Let $w = (w_i)_{i=1}^n$ be a vertex-bounded weight vector for G and let $|w| = \sum_i w_i$, then for all $\epsilon > 0$,*

$$\Pr\left(\sum_{i=1}^{n} w_i\left(\xi_i - \mu\right) \geq \epsilon\right) \leq \exp\left(-\frac{\epsilon}{2M}\log(1 + \frac{M\epsilon}{|w|\sigma^2})\right),$$

$$\Pr\left(\sum_{i=1}^{n} w_i\left(\xi_i - \mu\right) \geq \epsilon\right) \leq \exp\left(-\frac{\epsilon^2}{2(|w|\sigma^2 + \frac{1}{3}M\epsilon)}\right),$$

$$\Pr\left(\sum_{i=1}^{n} w_i\left(\xi_i - \mu\right) \geq \epsilon\right) \leq \exp\left(-\frac{\epsilon^2}{2|w|M^2}\right).$$

Proof. Without loss of generality, we assume $\mu = 0$. The first inequality follows from Theorem 5.13 and the inequality

$$h(a) \geq \frac{a}{2}\log(1+a),\ \forall a \geq 0.$$

The second inequality follows from Theorem 5.13 and the inequality

$$h(a) \geq \frac{3a^2}{6+2a},\ \forall a \geq 0.$$

To prove the third inequality, we use Theorem 5.12. As the exponential function is convex and $-M \leq \xi_i \leq M$, there holds

$$e^{c\xi_i} \leq \frac{c\xi_i - (-cM)}{2cM}e^{cM} + \frac{cM - c\xi_i}{2cM}e^{-cM}.$$

It follows from the assumption $\mu = 0$ and the Taylor expansion for the exponential function that

$$\mathbb{E}\left[e^{c\xi_i}\right] \leq \frac{1}{2}e^{-cM} + \frac{1}{2}e^{cM} = \frac{1}{2}\sum_{p=0}^{+\infty}\frac{(-cM)^p}{p!} + \frac{1}{2}\sum_{p=0}^{+\infty}\frac{(cM)^p}{p!} = \sum_{p=0}^{+\infty}\frac{(cM)^{2p}}{(2p)!}$$

$$= \sum_{p=0}^{+\infty}\frac{((cM)^2/2)^p}{p!}\prod_{j=1}^{p}\frac{1}{2j-1} \leq \sum_{p=0}^{+\infty}\frac{((cM)^2/2)^p}{p!} = \exp((cM)^2/2).$$

This, together with Theorem 5.12, implies

$$\begin{aligned}\Pr\left(\sum_{i=1}^{n} w_i\left(\xi_i - \mu\right) \geq \epsilon\right) &= \Pr\left(\exp\left(c\sum_{i=1}^{n} w_i\xi_i\right) \geq e^{c\epsilon}\right)\\ &\leq \exp\left(-c\epsilon + \mathbb{E}\left[c\sum_{i=1}^{n} w_i\xi_i\right]\right)\\ &\leq \exp\left(-c\epsilon + |w|(cM)^2/2\right).\end{aligned}$$

Choose $c = \epsilon/(|w|M^2)$. Then, $\Pr\left(\sum_{i=1}^{n} w_i\left(\xi_i - \mu\right) \geq \epsilon\right) \leq \exp\left(-\frac{\epsilon^2}{2|w|M^2}\right)$. □

Now we are ready to prove Theorem 5.10.

Proof of Theorem 5.10. We apply Theorem 5.14 to the variables $\xi_i' = \xi_i/|w|$ which satisfy $|\xi_i' - \mathbb{E}[\xi_i']| \leq M/|w|, \sigma^2(\xi_i') = \sigma^2/|w|^2$. □

From Theorem 5.10, we can see that the tighter bounds can be obtained by maximizing $|w|$. Given a hypergraph $G = (V, E)$, this can be achieved by solving the linear program (LP):

$$\begin{aligned} \max_w \quad & \sum_{i=1}^{n} w_i \\ \text{s.t.} \quad & \forall i : w_i \geq 0 \\ & \forall v \in V : \sum_{i:v\in e_i} w_i \leq 1 \end{aligned}$$

The optimal value of this linear program is called the fractional matching number (FMN) of the hypergraph G. We denote it as ν_G^*[1]. That is, ν_G^* is defined as the maximum of the sum of the weights, and a corresponding weight vector is called an FMN weight vector. Note that the linear program above and the same value ν^* are used in Chapter 3 to define a graph support measure. Decision variables in the linear program of the graph support measure are defined on vertices in overlap hypergraphs, but here decision variables (weights) are defined on hyperedges. In fact, if we replace an overlap hypergraph with its dual overlap hypergraph, then the two linear programs are exactly the same.

As we have shown, using an FMN weight vector we can achieve a good concentration inequality. We can also show an upper bound of the corresponding variance if G is a k-partite hypergraph:

Theorem 5.15. *If a weight vector w is an FMN weight vector of a k-partite hypergraph G, then the variance σ^2_{FMN} of the weighted sample mean can be bounded as follows:*

$$\sigma^2_{FMN} = var\left(\frac{1}{\nu^*}\sum_{i=1}^{n} w_i\xi_i\right) \leq \frac{\sigma^2}{\nu^*}.$$

Proof. Note that if the weight vector w (if it is feasible) is given in the program (4.9), the objective value t is the worst-case variance for this weight vector. Thus, for any positive number a, if a weight vector w satisfies

- $\forall l \in [k] : \sum_{i\in[n]} \sum_{j\in[n]} w_i w_j I(l \in J^E_{i,j}) \leq a$
- $\sum_{i\in[n]} w_i = 1$

[1]This number also appeared in [71] where we denoted it s.

- $\forall i : w_i \geq 0$

then the variance $\mathrm{var}\left(\sum_{i=1}^{n} w_i \xi_i\right)$ is at most $a\sigma^2$.

Now, we define $w' = \frac{w}{\nu^*}$, i.e., for every i, $w'_i = \frac{w_i}{\nu^*}$. Since w is a vertex-bounded weight vector, for all vertices v, $\sum_{i:v\in e_i} w_i \leq 1$ which implies

$$\left(\sum_{i:v\in e_i} w_i\right)^2 \leq \sum_{i:v\in e_i} w_i.$$

For every l, because every hyperedge meets $V^{(l)}$ exactly once, we get

$$\sum_{v:v\in V^{(l)}} \left(\sum_{i:v\in e_i} w_i\right)^2 \leq \sum_{i\in[n]} w_i. \tag{5.9}$$

The left hand side of Inequality (5.9) is equal to $\sum_{i\in[n]}\sum_{j\in[n]} w_i w_j I(l \in J^E_{i,j})$, and the right hand side is equal to ν^*. Then, we have $\forall l \in [k]$: $\sum_{i\in[n]}\sum_{j\in[n]} w'_i w'_j I(l \in J^E_{i,j}) \leq \frac{1}{\nu^*}$. It is easy to verify that $\sum_{i\in[n]} w'_i = 1$ and $\forall i : w'_i \geq 0$. Hence, w' satisfies (4.9) and $\sigma^2_{FMN} = \mathrm{var}(\sum_{i=1}^{n} w'_i \xi_i) \leq \frac{1}{\nu^*}\sigma^2$. □

Using Theorem 5.10, we can also improve existing concentration inequalities for the EQW weighting scheme [36]. Let w be a vertex-bounded weight vector and satisfy $w_1 = w_2 = \ldots = w_n$ (EQW). This requires that for all i, $0 < w_i \leq \frac{1}{\omega_G}$ where $\omega_G = \max_{v\in V_G} |\{e : v \in e\}|$ is the maximum degree of G. Let $w_1 = w_2 = \ldots = w_n = \frac{1}{\omega_G}$, we can get the following corollary.

Corollary 5.16. Let $(\xi_i)_{i=1}^n$ be G-networked random variables with mean $\mathbb{E}[\xi_i] = \mu$, variance $\sigma^2(\xi_i) = \sigma^2$, and satisfying $|\xi_i - \mu| \leq M$. Then for all $\epsilon > 0$,

$$\Pr\left(\frac{1}{n}\sum_{i=1}^{n}\xi_i - \mu \geq \epsilon\right) \leq \exp\left(-\frac{n\epsilon^2}{2\omega_G M}\log\left(1 + \frac{M\epsilon}{\sigma^2}\right)\right),$$

$$\Pr\left(\frac{1}{n}\sum_{i=1}^{n}\xi_i - \mu \geq \epsilon\right) \leq \exp\left(-\frac{n\epsilon^2}{2\omega_G(\sigma^2 + \frac{1}{3}M\epsilon)}\right),$$

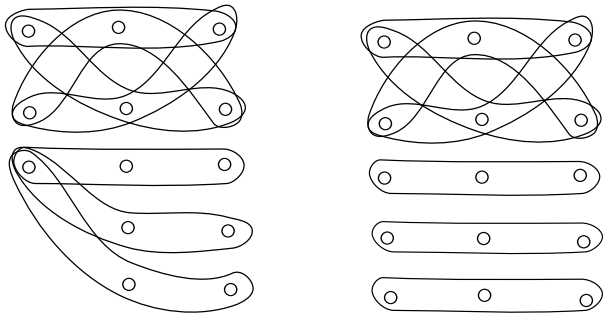
$$\Pr\left(\frac{1}{n}\sum_{i=1}^{n}\xi_i - \mu \geq \epsilon\right) \leq \exp\left(-\frac{n\epsilon^2}{2\omega_G M^2}\right).$$

We know that for every hypergraph G, it holds that the maximum degree of G is not larger than the fractional hyperedge-chromatic number of G, $\omega_G \leq \chi^*_G$. This

fact generally ensures the inequalities in Corollary 5.16 provide tighter bounds than those in Theorem 5.6. In addition, for any $r \geq 1$, there exist hypergraphs G such that $\frac{\chi_G^*}{\omega_G} > r$, and hence the improvement of Corollary 5.16 over Theorem 5.6 can be arbitrarily large. For example, consider (truncated) projective planes discussed already in the last section. The maximum degree of the projective plane of order κ ($\kappa \geq 2$) is $\kappa + 1$ while its fractional chromatic number is equal to the number of hyperedges $n = \kappa^2 + \kappa + 1$, so $\frac{\chi_G^*}{\omega_G} = \kappa + \frac{1}{\kappa+1} = O(|E_G|^{1/2})$. A similar result can be obtained for tuple networked examples. The maximum degree of the truncated projective plane of order κ is κ, while its fractional chromatic number is the same as the number of hyperedges $n = \kappa^2$, so $\frac{\chi_G^*}{\omega_G} = \kappa = O(|E_G|^{1/2})$.

It is possible that the size of the matching number of a hypergraph is smaller than $\frac{n}{\omega_G}$ (see Example 5.17), but it is also possible that it is larger (see Example 5.18). Therefore, concentration bounds using the IND weighting scheme (i.e., the classical concentration bounds applied to sets of independent examples) cannot be compared in strength with the above theorem. However, both $\frac{n}{\omega_G}$ and the size of a maximum independent set of hyperedges of G are smaller than ν_G^*. Therefore, the FMN weighting scheme always gives the best concentration bounds of these three weighting schemes.

In the following, we give some examples using tripartite hypergraphs which make the relationship between the three parameters (ν^*, ω and $|E_{IND}|$) clearer.



(a) A hypergraph with $|E_{IND}| < \frac{|E|}{\omega_G}$

(b) A hypergraph with $|E_{IND}| > \frac{|E|}{\omega_G}$

Figure 5.2: Two hypergraphs with different relationships between $|E_{IND}|$ and $\frac{|E|}{\omega_G}$.

Example 5.17. Consider the tripartite hypergraph in 5.2a. The three parameters of G satisfy the following inequality:

$$\nu_G^* = 3 > \frac{|E|}{\omega_G} = \frac{7}{3} > |E_{IND}| = 2.$$

⧫

Example 5.18. Consider the tripartite hypergraph in 5.2b. The three parameters of G satisfy the following inequality:

$$\nu_G^* = 5 > |E_{IND}| = 4 > \frac{|E|}{\omega_G} = 3.5.$$

⧫

5.3 Concentration inequalities for U-statistics

A statistical method, that generates unbiased estimators of minimum variance, involves the notion of U-statistics proposed by Hoeffding in [32]. A U-statistic is a class of estimators, which can usually be written as averages over functions on distinct samples of size r taken from an i.i.d. sample $\{x_1, x_2, \ldots, x_m\}$, e.g., the sample mean, sample variance, sample moments, Kendall-τ (see [38]), Wilcoxon's signed-rank sum (see [72]), etc. Hoeffding also derived concentration inequalities for U-statistics. Using results from Section 5.2, we can improve these concentration inequalities. As an example, we only consider one-sample U-statistics.

Definition 5.19 (One-sample U-statistics)**.** Let $\{x_i\}_{i=1}^m$ be independent random variables. For $m \geq r$ consider a random variable of the form

$$U = \sum_{\binom{m}{r}} \xi(x_{i_1}, \ldots, x_{i_r})$$

where the sum $\sum_{\binom{m}{r}}$ is taken over all subset $\{i_1, \ldots, i_r\}$ of distinct positive integers not exceeding m. The random variable U is called a one-sample U-statistic.

It is always possible to construct a hypergraph $G = (V, E)$ for a one-sample U-statistic. This graph has m vertices, and $E = \{S \subseteq V \mid |S| = r\}$. We consider the independent random variables $\{x_i\}_{i=1}^m$ as the features of the vertices. The statistic U is an equally weighted sample mean of the networked random variables of these hyperedges.

If the function ξ is bounded, $|\xi - \mathbb{E}[\xi]| \leq M$, [32] showed that for any $\epsilon > 0$,

$$\Pr\left(U - \mu \geq \epsilon\right) \leq \exp\left(-\frac{\lfloor \frac{m}{r} \rfloor \epsilon^2}{2M^2}\right)$$

where $\mu = \mathbb{E}[U]$.

Arcones showed a Bernstein-type bound [2] that if $\mathrm{var}(\xi) = \sigma^2$, then for any $\epsilon > 0$,

$$\Pr\left(U - \mu \geq \epsilon\right) \leq \exp\left(-\frac{\lfloor \frac{m}{r} \rfloor \epsilon^2}{2(\sigma^2 + \frac{M\epsilon}{3})}\right).$$

However, a corollary of our result shows that the operator $\lfloor \rfloor$ is not necessary, i.e.,

$$\Pr\left(U - \mu \geq \epsilon\right) \leq \exp\left(-\frac{m\epsilon^2}{2rM^2}\right)$$

and

$$\Pr\left(U - \mu \geq \epsilon\right) \leq \exp\left(-\frac{m\epsilon^2}{2r\left(\sigma^2 + \frac{M\epsilon}{3}\right)}\right).$$

To prove these inequalities, we just let $n = \frac{m!}{(m-r)!r!}$ and $\omega_G = \frac{(m-1)!}{(m-r)!(r-1)!}$ in Corollary 5.16.

5.4 Summary

Janson's inequality showed that ignoring the dependency relationships between random variables may result in poor estimations. We proposed the FMN weighting scheme to generalize existing Chernoff-Hoeffding style concentration inequalities. The weights in our weighting schemes can be computed efficiently.

While the MinVar weighting scheme is optimal for worst case variance, we do not have a proof that the FMN scheme is optimal for concentration bounds. Hence, it would be interesting to find methods to optimize the weights for concentration inequalities.

Chapter 6

Statistical learning theory on networked examples

Many practical approaches to supervised learning in networks ignore (at least partially) the problem of dependent data and learn models with classic machine learning techniques. While these work to some extent, they are not supported by a well-developed theory such as the one which provides generalization guarantees for the i.i.d. case as a function of the number of training examples. As a consequence, one may miss opportunities to learn due to the numerous dependencies between the training examples.

In the previous chapters, we have developed theory for statistics on networked data, in particular we focused on concentration and variance bounds. For the variance criterion, we determined the weighting scheme that minimizes the (worst case) variance among all weighted average estimators of a distribution mean. For what concerns concentration bounds, we showed that the FMN weighting scheme provides clearly better properties than classical approaches (even though we could not prove optimality of a weighting scheme for concentration bounds).

Statistical learning theory gives answers to fundamental questions about learning from examples. Which conditions ensure that a function (a concept) can be learned from examples? Why, in some cases, the measured performance on a dataset lead to guarantees on the generalization performance? How can we properly choose hypothesis spaces? How many examples are needed for training? These theoretical results are elegant and helpful, though the assumptions for the results to be valid are almost impossible to check for most (if not all) practical tasks. Exponential concentration inequalities (and to some extent also variance

bounds) form crucial tools in statistical learning theory. As an application, in this chapter, we use the results we obtained in the previous chapter to show generalization performance guarantees when learning from networked examples, making the same relaxed assumptions as in previous sections. We do this in the context of a specific framework (empirical risk minimization), but the same principles can be applied to many other paradigms in the field of learning theory.

6.1 Learning theory

We first review some basic concepts of statistical learning theory and empirical risk minimization, and then discuss the learning theory for networked examples.

Expected risk and empirical risk

The main goal of supervised learning is to learn a function $f : \mathbb{X} \mapsto \mathcal{Y}$ from a set of training examples $Z = \{z_i\}_{i=1}^n$ with $z_i = (x_i, y_i) \sim \rho$, and to predict labels for unseen examples. We define a loss function $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+$. The value $L(f(x), y)$ denotes the expected local error suffered from the use of f to predict y from x. We use the square loss function, that is $L(f(x), y) = (f(x) - y)^2$. Note that our analysis can easily be extended to general loss functions. We can measure the predictive ability of a learned model f approximating ρ by averaging the local error over all pairs (x, y) by integrating over $\mathcal{Z}$ with respect to ρ. More precisely, we define the *expected risk* as

$$\mathcal{E}(f) = \int_{\mathcal{Z}} (f(x) - y)^2 \rho(x, y) dx dy.$$

A natural idea is to find the minimizer $f_{\rho,\mathcal{F}}$ of $\mathcal{E}(f)$ over all functions, i.e.,

$$f_{\rho,\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathcal{E}(f),$$

where the minimization is taken over the set of all measurable functions $\mathcal{F}$. Unfortunately, because the probability distribution ρ is unknown, $f_{\rho,\mathcal{F}}$ cannot be computed directly. If examples in Z were mutually independent, by the law of large numbers, as the sample size n tends to infinity, the *empirical risk*

$$\mathcal{E}_Z(f) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2$$

converges to the expected risk $\mathcal{E}(f)$. Then we may get a good candidate $f_{Z,\mathcal{F}}$ to approximate the target function $f_{\rho,\mathcal{F}}$, where

$$f_{Z,\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \mathcal{E}_Z(f).$$

Empirical risk minimization principle

In order to avoid over-fitting, one usually does not take the minimization of the empirical risk over all the measurable functions. The main idea of the empirical risk minimization (ERM) principle [57] is to find the minimizer in a properly selected hypothesis space $\mathcal{H}$, i.e.,

$$f_{Z,\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathcal{E}_Z(f).$$

The performance of the ERM approach is commonly measured in terms of the *excess risk*

$$\mathcal{E}(f_{Z,\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{F}}).$$

If we define

$$f_{\rho,\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathcal{E}(f),$$

then the excess risk can be decomposed as

$$\mathcal{E}(f_{Z,\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{F}}) = [\mathcal{E}(f_{Z,\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{H}})] + [\mathcal{E}(f_{\rho,\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{F}})].$$

We call the first part the *sample error* $\mathcal{E}_S(Z) := \mathcal{E}(f_{Z,\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{H}})$, the second part the *approximation error* $\mathcal{E}_A(\mathcal{H}) := \mathcal{E}(f_{\rho,\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{F}})$.

The approximation error is independent of the sample and it is studied in [17]. It is an interesting question how to choose a proper hypothesis space. Intuitively, a small hypothesis space brings a large approximation error, while large hypothesis space results in over-fitting. Hence the hypothesis space must be chosen to be not too large or too small. It is closely related to the bias-variance problem. In this chapter, we concentrate on the sample error.

The complexity of the hypothesis space is usually measured in terms of covering number [75], entropy number [63], VC-dimension [67], etc. As an illustration of our approach, we use the covering numbers defined below to measure the capacity of our hypothesis space $\mathcal{H}$, and the hypothesis space $\mathcal{H}$ will be chosen as a subset of $\mathcal{C}(\mathbb{X})$ which is a Banach space of continuous functions on a compact metric space $\mathbb{X}$ with the norm $\|f\|_\infty = \sup_{x\in\mathbb{X}} |f(x)|$. However, our approach can be applied using other hypothesis space measures as well.

Before stating the existing results, we first introduce some notations and definitions.

Definition 6.1 (Covering number)**.** Let $\mathcal{H}$ be a metric space and $\tau > 0$. We define the *covering number* $N(\mathcal{H}, \tau)$ to be the minimal $\ell \in \mathbb{N}$ such that there exists ℓ disks in $\mathcal{H}$ with radius τ covering $\mathcal{H}$. When $\mathcal{H}$ is compact, this number is finite.

Definition 6.2 (M-bounded functions)**.** Let $M > 0$ and ρ be a probability distribution on $\mathcal{Z}$. We say that a set $\mathcal{H}$ of functions from $\mathbb{X}$ to $\mathcal{Y}$ is *M-bounded* when

$$\Pr_{(x,y)\sim\rho}\left(\sup_{f\in\mathcal{H}}|f(x)-y|\leq M\right)=1.$$

The following result for the i.i.d. case can be found in [17].

Theorem 6.3. *Let $\mathcal{H}$ be a compact and convex subset of $\mathcal{C}(\mathbb{X})$. If $\mathcal{H}$ is M-bounded, then for all $\epsilon > 0$,*

$$\Pr\left(\mathcal{E}_S(Z)\geq\epsilon\right)\leq N\left(\mathcal{H},\frac{\epsilon}{12M}\right)\exp\left(-\frac{n\epsilon^2}{300M^4}\right).$$

6.2 Learning theory for networked examples

Now, we provide statistical learning theory for learning from networked examples. We consider three weighting schemes having different upper sample error bounds which are related to different important parameters of hypergraphs. The first two weighting schemes are straightforward, but from the upper bound point of view, they waste the information provided by the networked examples. The third weighting scheme reaches a better sample error bound via solving the linear program discussed in Chapter 5.

The EQW weighting scheme

Let us first consider the EQW weighting scheme that learns from a set of networked examples in the same way as if they were i.i.d. (i.e., without weighting them as a function of the network structure). We can use Corollary 5.16 above to bound the sample error of EQW scheme:

Theorem 6.4. *Let $\mathcal{H}$ be a compact and convex subset of $\mathcal{C}(\mathbb{X})$, and Z be a G-networked sample. If $\mathcal{H}$ is M-bounded, then for all $\epsilon > 0$,*

$$\Pr\left(\mathcal{E}_S(Z)\geq\epsilon\right)\leq N\left(\mathcal{H},\frac{\epsilon}{12M}\right)\exp\left(-\frac{n\epsilon^2}{300\omega_G M^4}\right).$$

The result above shows that the bound of the sample error not only relies on the sample size but also depends on the maximum degree ω_G. That is, a larger sample may result in a poorer sample error bound since ω_G can also become larger.

Remark: In [64], based on Janson's inequalities, the authors provided a generalization bound for classifiers trained with equally weighted networked data. Using Corollary 5.16, their results can be improved.

The IND weighting scheme

A straightforward idea to learn from a G-networked sample Z is to find a (maximal) subset $Z_I \subseteq Z$ of training examples that correspond to a matching in G. Due to our assumptions, such set will be an i.i.d. sample. We can then perform algorithms on Z_I for learning. We can define the empirical risk

$$\mathcal{E}_{Z_I}(f) = \frac{1}{|Z_I|} \sum_{z_i \in Z_I} \left(f(x_i) - y_i\right)^2,$$

and the function we obtain by the ERM principle is

$$f_{Z_I,\mathcal{H}} = \arg\min_{f \in \mathcal{H}} \mathcal{E}_{Z_I}(f).$$

To bound the sample error of this weighting scheme, we can directly use Theorem 6.3, replacing n there by $|Z_I|$.

A key step in applying the IND weighting scheme is to find a large Z_I. The larger Z_I is, the more accurate f_{Z_I} we can guarantee. To find a large Z_I is equivalent to find a large matching in G. However, given a positive integer n_0, it is in general an NP-complete problem to decide whether there is a matching in G of size greater than n_0 [26]. Moreover, the maximum matching problem is also an APX-complete problem [21], so we would not expect an efficient algorithm to achieve a good approximation in practice.

The FMN weighting scheme

We now consider the FMN weighting scheme proposed in the last chapter. The ν^* value is a linear program relaxation of the maximum matching problem [44, 13], so it always holds that $\nu^*_G \geq |E_{IND}|$ where $|E_{IND}|$ is the matching number, i.e., the size of a maximum independent set of hyperedges.

For a G-networked sample Z, we denote the FMN weighted sample $Z_{\nu^*} = \{(z_i, w_i)\}$ where $(w_i)_{i=1}^n$ is an FMN weight vector. Now we can define a new empirical risk on the FMN weighted sample Z_{ν^*} that

$$\mathcal{E}_{Z_{\nu^*}}(f) = \frac{1}{\nu^*} \sum_{i=1}^{n} w_i \left(f(x_i) - y_i\right)^2.$$

Later, we show that the empirical risk $\mathcal{E}_{Z_{\nu^*}}$ converges to the expected risk $\mathcal{E}(f)$ as ν^* tends to infinity for fixed f.

We consider the ERM approach associated with Z_{ν^*}. As discussed in Section 6.1, the ERM approach aims to find a minimizer of the empirical risk in a proper hypothesis space $\mathcal{H}$ to approximate the target function, i.e.,

$$f_{Z_{\nu^*},\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathcal{E}_{Z_{\nu^*}}(f).$$

Then the performance of the ERM approach is measured by the excess risk

$$\mathcal{E}(f_{Z_{\nu^*},\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{F}}).$$

Recall the definition $f_{\rho,\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathcal{E}(f)$, the excess risk can be divided into two parts (sample error and approximation error) as follows

$$\mathcal{E}(f_{Z_{\nu^*},\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{F}}) = [\mathcal{E}(f_{Z_{\nu^*},\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{H}})] + [\mathcal{E}(f_{\rho,\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{F}})].$$

We focus on the sample error $\mathcal{E}_S(Z_{\nu^*}) := \mathcal{E}(f_{Z_{\nu^*},\mathcal{H}}) - \mathcal{E}(f_{\rho,\mathcal{H}})$. To this end, we use the probability inequalities with ν^* (see Theorem 5.10) to estimate the sample error $\mathcal{E}_S(Z_{\nu^*})$. The following is the main result of this section.

Theorem 6.5. *Let $\mathcal{H}$ be a compact and convex subset of $\mathcal{C}(\mathbb{X})$. If $\mathcal{H}$ is M-bounded, then for all $\epsilon > 0$,*

$$\Pr\left(\mathcal{E}_S(Z_{\nu^*}) \geq \epsilon\right) \leq N\left(\mathcal{H}, \frac{\epsilon}{12M}\right) \exp\left(-\frac{\nu^*\epsilon^2}{300M^2}\right).$$

Remark: We mainly consider the ERM principle as an example of applying our concentration results to statistical learning theory. Many other learning approaches can also be analyzed using these concentration inequalities. For example, by using the inequalities in the section of networked concentration inequalities, similar results (generalization bounds) can be obtained for the regularization method (see e.g.,[29, 61]) which is a way to deal with over-fitting. Besides, the technique we used to prove our concentration inequalities can be easily adapted to get a Chernoff-type inequality for networked Bernoulli random variables which is useful if we intend to have a PAC-Bayesian bound.

Effective sample size

An important aspect of the theory presented can be understood as a better estimation of the effective sample size of a dataset. Several slightly different definitions exist, but generally speaking one can define the effective sampling

size of a dataset G for a particular statistical approximation task F as the number of examples an i.i.d. dataset would need to allow for estimating F as accurately as can be done with the original dataset G.

In that light, for the sake of estimating a statistic with concentration guarantees as in the last chapter or learning with PAC-style bounds as in Theorem 6.5, the fractional matching number ν^* is the effective sample size we can achieve using our theory, while classic approaches based on unweighted averaging would only achieve smaller effective sample sizes such as the one provided by Theorem 5.6. As illustrated in Example 5.8, the difference between these effective sample sizes can be substantial.

When the objective is variance minimization, Chapter 4 provides an approach to compute the effective sample size when using the MinVar weighting scheme. The variance criterion is in general easier than the concentration bound criterion, i.e., for some datasets the effective sample size for estimating a statistic with minimal variance will be larger than the effective sample size for estimating a statistic satisfying a concentration bound. The reason for this can be found in the fact that for classic data, even though i.i.d. is a sufficient condition, it is stronger than needed to guarantee a certain variance. In particular, if $(\xi_i)_{i=1}^n$ is a set of random variables each having variance σ^2, it is sufficient to assume they are uncorrelated to conclude that their average will have variance σ^2/n. Random variables may be uncorrelated but not independent.

When we know the dataset, we can compute the effective sample sizes according to the criteria of interest, and if we want to be safe we can take the most conservative effective sample size ν^*. Then, in the vast majority of results for i.i.d. data we are able to obtain an equivalent for networked data by replacing the sample size n by the effective sample size ν_G^* of the network. We expect this does not only hold for the learning result of Theorem 6.5, but for virtually any statistical computation that can be expressed in terms of averages and concentrations.

6.3 Related Work

In this section, we provide some additional discussion of relations between our results and existing work.

Hypothesis tests

In [69], the authors consider a similar setting of networked examples. They use dependency graphs to represent the examples and their relations. While we assume a worst case over all possible dependencies, and allow to model explicitly causes of dependencies (represented with vertices which can be incident with more than two edges), this work assumes a bounded covariance between pairs of examples connected with an edge (excluding possible higher-order interactions). While we use our model to show learning guarantees, [69] shows corrections for the bias (induced by the dependencies between examples) on statistical hypothesis tests. It seems plausible that both models can be applied for both the learning guarantee and statistical testing tasks.

Mixing conditions

There is also some literature on learning from a sequence of examples where examples close in the sequence are dependent. In the community of machine learning, mixing conditions are usually used in time series analysis. For example, in [29], the learning performance of a regularized classification algorithm using a non-i.i.d. sample is investigated, where the independence restriction is relaxed to so-called α-mixing or β-mixing conditions. In [61], regularized least squares regression with dependent samples is considered under the assumption that the training sample satisfies some mixing conditions. In [47], the authors presented a Bernstein type inequality for stationary exponentially α-mixing processes, that is based on the effective number (less than the sample size). Our Bernstein type inequalities for dependent network data too assigns weights to examples. However, the assumptions for the training sample are different, and the main techniques are distinct. Moreover, in practice, it is not easy to check whether the training sample satisfies the mixing conditions. Our networked training examples certainly do not satisfy any of these mixing conditions. We refer interested readers to [7] and references therein for more details about the mixing conditions.

Statistical relational learning

Our theory is applicable to problems considered in the field of Statistical Relational Learning (SRL) [27], e.g., for learning local conditional probability functions for directed models such as Probabilistic Relational Models [25], Logical Bayesian Networks [23], Relational Bayesian Networks [35].

There is a huge literature in SRL for learning features and existence of edges in a graph, for which we refer the reader to the excellent survey of [54]. An important difference to many of these is that we do not assume that the distribution of connections for every vertex in the test set will be similar to what we have seen in the training set. This is tightly connected to our independence assumptions. Both the classic assumptions (where often testing examples are found in the same network) and ours have clear advantages which are more or less important depending on the application. A question of future research is how to combine both aspects. E.g., if one wants to build models for time-evolving networks where a significant amount of vertices are replaced over time, the importance will gradually move from the classical setting where all examples are in the same network towards a situation where the future examples are in a new network with new vertices and connections but where still the same underlying processes apply.

There are also methods that aim at addressing settings where training set and test set are different, e.g., transfer learning approaches such as [31] and [45]. An important difference with this direction of research is that our approach does not need to learn the distribution of the test set, e.g., using a sample of labeled or unlabeled examples.

6.4 Summary

In this chapter, we reviewed statistical learning theory for the i.i.d. case and extended it to the networked case by exploiting concentration inequalities in the last chapter. We considered three different weighting schemes in which the FMN weighting scheme achieves the best learning bound. Though we only considered a specific type of bound, the result in this chapter can be generalized to a large fraction of existing learning theory results, e.g., PAC-Bayes learning bounds.

Interesting questions for future work are whether this strategy is (nearly) optimal or not and how to obtain a minimax lower bound for learning from networked examples.

Chapter 7

Conclusion

In this chapter, we summarize this thesis, discuss potential impacts of this work, and conclude it by pointing out some future work.

7.1 Thesis summary

In this thesis, we considered the problem of mining and learning from networked data.

We first investigated how to define support measures of subgraph patterns in a large network. The challenge of this task comes from the fact that occurrences of a subgraph pattern may overlap in the large network. The main goal of this work has several aspects. First, some properties of support measures, such as anti-monotonicity, play important roles in mining algorithms, so it is reasonable to require support measures of subgraph patterns to have such properties. Second, support measures should be normalized, i.e., when occurrences do not overlap, support measures should output the number of occurrences. Most anti-monotonic and normalized support measures are based on the concept of overlap graphs. Third, a support measure should be robust, because it is very common that the collected data has some noise. When we remove only one vertex from the large network, the support of a pattern is not supposed to change a lot. Fourth, a useful support measure of subgraph patterns needs to be efficiently computable. However, these earlier overlap graph based support measures cannot be efficiently computed. Fifth, it would be good if a support measure has an elegant statistical interpretation, which is related to the second

part of our work.

In Chapter 3, we proposed a support measure of subgraph patterns based on a new concept of overlap hypergraphs. This new support measure fulfills the aforementioned requirements. We considered how to keep the properties of being normalized, anti-monotonic and robust for overlap graph based support measures. At the same time, we tried to reduce the computational expenses by using linear programs. This support measure also inspired us to understand the statistical power of networked data.

We also considered how to do statistics on networked random variables. In most applications, data is assumed to be i.i.d. We reviewed earlier results on doing statistics on non-i.i.d. data, but most of these results rely on other assumptions such as mixing conditions which are not applicable to our cases. We introduced a hypergraph based model and a new assumption to replace the classical i.i.d. assumption. Janson proved Chernoff-Hoeffding style inequalities for this type of random variables and some statistical learning theory results based on Janson's inequalities exist. However, these earlier works only studied the case that we ignore the dependencies between data points and equally weight every data points. We demonstrated that ignoring the dependencies between data points and using unweighted (or equally weighted, the EQW weighting scheme) estimators may result in a poor estimation. To design a better estimator, we can choose an independent subset of these networked random variables (the IND weighting scheme), but we would waste too much information in the data. We proposed two other schemes for weighting data points that allow for using the available training data to a large extent while mitigating the dependency problem. In particular, the MinVar weighting scheme described in Chapter 4 is optimal from the worst-case variance point of view, while the FMN weighting scheme described in Chapter 5 allows for generalizing a large fraction of existing Chernoff-Hoeffding style concentration inequalities. The weights in our weighting schemes can be computed efficiently. As an example, in Chapter 6, we applied our statistical concentration results to the problem of learning from networked examples, in particular, empirical risk minimization principle on networked examples.

7.2 Discussions

In this thesis, we have designed a support measure of subgraph patterns and related this support measure to statistics on networked examples.

This graph support measure is anti-monotonic so it can be used to prune the search space of a frequent subgraph pattern mining task. Unlike many other

anti-monotonic support measures, this support measure is efficiently computable. As our experimental results show, this support measure can be used to mine frequent subgraph patterns in real-world networks.

This support measure is potentially helpful in several applications such as motif detection [59], uncertain graph mining [77] and graph mining with privacy [58]. For motif detection, one usually directly counts the occurrences of a subgraph pattern, and then compares the number to the expected number of occurrences in a random graph. A random graph is treated as a homogeneous network, i.e., subgraph patterns distribute evenly in a random graph. However, it is possible that a real network have more occurrences of a pattern than that in a random graph, but most of these occurrences depends on a single vertex. In this case, our support measure seems more convincing than the number of occurrences. Mining a large uncertain graph (a random graph, but presence probabilities of different edges may be different) needs a definition of support measure. The uncertainty of this type of graph data increases the difficulty of the task. Our support measure can be the starting point, i.e., if the presence probabilities only take two values 0 (absence) and 1 (presence), then a good support measure in this network should be related to our support measure. A popular way to do privacy preserving data mining is based on the differential privacy technique, and this technique can be used to graph data as well. An important precondition of this method is that every data point should have a very limited effect in statistics, and our graph support measure restricts the contribution of every vertex. If a vertex participates in many examples and we equally weight all examples for some statistics, then the contribution of this vertex will be larger than expected.

The part of networked statistics showed a variance bound and Chernoff-Hoeffding style concentration inequalities which are applied to learning from networked examples. The presented theory forms a first step towards a statistically sound theory for learning in networks. Our ongoing work mainly focuses on this direction.

Our variance bound is related to the standard ANOVA techniques for analyzing k-partite networked examples, and is leading us to stronger concentration results for the k-partite case. The proved Chernoff-Hoeffding style concentration inequalities for the general hypergraph case are powerful, and can be exploited in many tasks that were not discussed in this thesis yet. First, as Janson pointed out in his paper [36], we can apply our concentration results to analyze random structures such as random graphs and random hypergraphs. Second, we can use these concentration inequalities to provide generalization bounds under some mixing conditions [50]. Third, these inequalities are useful for other fields in computer science, e.g., randomized algorithm analysis. Fourth, one of our ongoing studies deals with phylogenetic trees in which data points are

structured as a tree, and we plan to generalize our concentration inequalities to these cases.

However, the assumption we made for this networked example model and corresponding random variables is still not realistic in many cases. As we pointed out with examples, a person usually does not choose movies randomly because (s)he would have some preferences. Besides, as John Donne said, "No man is an Island, entire of itself", data points could be influenced by other connected data points. Therefore, in the future, we have to make a more realistic assumption than the one made in this thesis and study some related questions.

7.3 Future work

We have obtained some results for mining and learning from networked examples, but as we mentioned in the previous section, there are several remaining questions to investigate.

First, in order to analyze more real-world networked cases, one needs to build new models with more flexible assumptions than the classic i.i.d. assumption and the weaker assumption we studied in this thesis which is already more general but still is not sufficiently powerful to properly model a range of real-world scenarios.

This thesis has provided a learning bound based on our weaker assumption. It is also interesting to investigate the conditions of learnability from networked non-i.i.d. data for new models, i.e., study generalization guarantees, e.g., using PAC-style bounds or expected risk bounds, and to investigate to what extent the provided bounds are the best possible ones.

Besides the theory, accurate and efficient algorithms to learn from non-i.i.d. data in networks are also necessary. Part of this task should concern general "upgrades" which are applicable to the majority of existing algorithms.

To design active learning methods for structured data is another useful topic, i.e., to study query strategies to choose objects or examples in a network to perform experiments in order to learn a good predictor at minimal cost.

It would be a meaningful work if one studies the implications of the above theory and algorithms for other learning settings and tasks, e.g., cross-validation, bootstrapping and ranking (see, e.g., [64]), thereby improving on work as [69] on cross-validation and [43] on bootstrapping.

The theory and algorithms described above should be validated on real-world

datasets, especially in the fields of social networks, biological regulatory networks and chemical interaction networks.

For the model and assumption described in this thesis, there are also several problems to solve, e.g., how to get better concentration bounds for k-partite hypergraphs and how to obtain lower bounds for learning from networked examples.

Appendix A

A.1 Decomposition of the variance

In this part, we provide the proofs of the properties of the decomposition in Chapter 4.

Lemma 4.15 *Every μ_S is zero-mean for every dimension, i.e., For any $S \subseteq [k]$ and $i \in S$, $\mathbb{E}_{x^{(i)} \sim \rho^{(i)}} \left[\mu_S \left(x^{(S)} \right) \right] = 0$.*

Proof. We first show that,

$$\mu_S \left(x^{(S)} \right) = \sum_{T \subseteq S} (-1)^{|S \setminus T|} \mathbb{E}_{x^{([k] \setminus T)} \sim \rho^{([k] \setminus T)}} \left[f(x) | x^{(T)} \right] \tag{A.1}$$

by induction on $|S|$. For $S = \emptyset$, from the definition of $S_\emptyset$,

$$\mu_S \left(x^{(S)} \right) = \mu_\emptyset \left(x^{(\emptyset)} \right) = \mathbb{E}_{x^{([k])} \sim \rho^{([k])}} [f(x)]$$

and (A.1) follows because the only subset of the empty set is the empty set itself, Assume that Eq. (A.1) holds for $|S| = 0, \ldots, l$, we now prove Eq. (A.1) holds for $|S'| = l + 1$. By definition,

$$\begin{aligned} \mu_{S'} \left(x^{(S')} \right) &:= \mathbb{E}_{x^{([k] \setminus S')} \sim \rho^{([k] \setminus S')}} \left[f(x) | x^{(S')} \right] - \sum_{T \subset S'} \mu_T \left(x^{(T)} \right) \\ &= (-1)^{|S' \setminus S'|} \mathbb{E}_{x^{([k] \setminus S')} \sim \rho^{([k] \setminus S')}} \left[f(x) | x^{(S')} \right] - \sum_{T \subset S'} \mu_T \left(x^{(T)} \right). \end{aligned}$$

Using the induction hypothesis on $\mu_T \left(x^{(T)} \right)$ for every $T \subset S'$, we see that $\mu_{S'} \left(x^{(S')} \right)$ can be written as a linear combination of

$\left(\mathbb{E}_{x^{([k]\setminus T')}\sim\rho^{([k]\setminus T')}}\left[f(x)|x^{(T')}\right] : T' \subseteq S'\right)$. For any $T' \subset S$, the induction hypothesis implies that there is a term $(-1)^{|T\setminus T'|}\mathbb{E}_{x^{([k]\setminus T')}\sim\rho^{([k]\setminus T')}}\left[f(x)|x^{(T')}\right]$ in the expansion of $\mu_T\left(x^{(T)}\right)$ if $T' \subseteq T \subset S'$. The coefficient of $\mathbb{E}_{x^{([k]\setminus T')}\sim\rho^{([k]\setminus T')}}\left[f(x)|x^{(T')}\right]$ in the expansion of $\mu_{S'}\left(x^{(S')}\right)$ is

$$\sum_{T:T'\subseteq T\subset S'} (-1)^{|T\setminus T'|} = \sum_{i=1}^{|S'\setminus T'|} \binom{|S'\setminus T'|}{i}(-1)^i = (-1)^{|S'\setminus T'|}$$

where the second equality comes from the well known identity $\sum_{i=0}^{a}\binom{a}{i}(-1)^i = 0$, so Eq. (A.1) holds.

Now, starting from Eq. (A.1),

$$\begin{aligned}
& \mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\mu_S\left(x^{(S)}\right)\right] \\
= \; & \mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\sum_{T:T\subseteq S} (-1)^{|S\setminus T|}\mathbb{E}_{x^{([k]\setminus T)}\sim\rho^{([k]\setminus T)}}\left[f(x)|x^{(T)}\right]\right] \\
= \; & \mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\sum_{T:T\subseteq S\wedge i\notin T} (-1)^{|S\setminus T|}\mathbb{E}_{x^{([k]\setminus T)}\sim\rho^{([k]\setminus T)}}\left[f(x)|x^{(T)}\right]\right] \\
& + \mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\sum_{T:T\subseteq S\wedge i\in T} (-1)^{|S\setminus T|}\mathbb{E}_{x^{([k]\setminus T)}\sim\rho^{([k]\setminus T)}}\left[f(x)|x^{(T)}\right]\right].
\end{aligned}$$

The first summation does not depend on $x^{(i)}$, so we can drop the expectation, while in the second term we can merge both expectations:

$$\begin{aligned}
& \mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\mu_S\left(x^{(S)}\right)\right] \\
= & \sum_{T:T\subseteq S\wedge i\notin T}(-1)^{|S\setminus T|}\mathbb{E}_{x^{([k]\setminus T)}\sim\rho^{([k]\setminus T)}}\left[f(x)|x^{(T)}\right] \\
& +\mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\sum_{T:T\subseteq S\wedge i\in T}(-1)^{|S\setminus T|}\mathbb{E}_{x^{([k]\setminus T)}\sim\rho^{([k]\setminus T)}}\left[f(x)|x^{(T)}\right]\right]. \\
= & \sum_{T:T\subseteq S\wedge i\notin T}(-1)^{|S\setminus T|}\mathbb{E}_{x^{([k]\setminus T)}\sim\rho^{([k]\setminus T)}}\left[f(x)|x^{(T)}\right] \\
& +\sum_{T:T\subseteq S\wedge i\in T}(-1)^{|S\setminus T|}\mathbb{E}_{x^{(([k]\setminus T)\cup\{i\})}\sim\rho^{(([k]\setminus T)\cup\{i\})}}\left[f(x)|x^{(T\setminus\{i\})}\right]
\end{aligned}$$

Substituting in the second term T with $T\setminus\{i\}$ we obtain:

$$\begin{aligned}
& \mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\mu_S\left(x^{(S)}\right)\right] \\
= & \sum_{T:T\subseteq S\wedge i\notin T}(-1)^{|S\setminus T|}\mathbb{E}_{x^{([k]\setminus T)}\sim\rho^{([k]\setminus T)}}\left[f(x)|x^{(T)}\right] \\
& +\sum_{T:T\subseteq S\wedge i\notin T}(-1)^{|S\setminus T|+1}\mathbb{E}_{x^{([k]\setminus T)}\sim\rho^{([k]\setminus T)}}\left[f(x)|x^{(T)}\right]=0.
\end{aligned}$$

□

Lemma 4.16 *For any $S\neq T$, the functions μ_S and μ_T are uncorrelated (orthogonal), i.e., they have zero covariance or $cov(\mu_S,\mu_T)=\mathbb{E}\left[\mu_S\left(x^{(S)}\right)\mu_T\left(x^{(T)}\right)\right]=0$.*

Proof. Because $S \neq T$, either $S \setminus T$ or $T \setminus S$ is non-empty. We assume without loss of generality that $T \setminus S \neq \emptyset$. Let $i \in T \setminus S$. Then,

$$\begin{aligned} & \mathbb{E}\left[\mu_S\left(x^{(S)}\right)\mu_T\left(x^{(T)}\right)\right] \\ = \; & \mathbb{E}_{x^{([k]\setminus\{i\})}\sim\rho^{([k]\setminus\{i\})}}\left[\mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\mu_S\left(x^{(S)}\right)\mu_T\left(x^{(T)}\right)\right]\right] \\ = \; & \mathbb{E}_{x^{([k]\setminus\{i\})}\sim\rho^{([k]\setminus\{i\})}}\left[\mu_S\left(x^{(S)}\right)\mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\mu_T\left(x^{(T)}\right)\right]\right] \end{aligned}$$

The second equality holds because μ_S is independent of the value $x^{(i)}$. Now, we use Lemma 4.15 that $\mathbb{E}_{x^{(i)}\sim\rho^{(i)}}\left[\mu_T\left(x^{(T)}\right)\right] = 0$, so $\mathbb{E}\left[\mu_S\left(x^{(S)}\right)\mu_T\left(x^{(T)}\right)\right] = 0$. □

Lemma 4.18 *The variance of the function f is the sum of the variances of μ_S of all S, i.e., $\sigma^2 = \sum_{S\subseteq[k]}\sigma_S^2$ where $\sigma^2 = \mathbb{E}\left[f^2\right]$ and $\sigma_S^2 = \mathbb{E}_{x^{(S)}\sim\rho^{(S)}}\left[\mu_S^2\left(x^{(S)}\right)\right]$.*

Proof. By Eq. (4.5),

$$f(x) = \sum_{S\subseteq[k]}\mu_S\left(x^{(S)}\right).$$

It follows that $\sigma^2 = \sum_{S,T\subseteq[k]}\operatorname{cov}\left(\mu_S,\mu_T\right)$. From Lemma 4.16 we know that if $S \neq T$ then $\operatorname{cov}\left(\mu_S,\mu_T\right) = 0$. Therefore, $\sigma^2 = \sum_{S\subseteq[k]}\sigma_S^2$. □

A.2 Proofs of concentration inequalities

In this part, we prove Lemma 5.11.

Lemma 5.11 *Let $\beta = (\beta_i)_{i=1}^k \in \mathbb{R}_+^k$ such that $\sum_{i=1}^k \beta_i \leq 1$. Then, the function $g(t)$ with $t = (t_i)_{i=1}^k \in \mathbb{R}_+^k$ defined by $g(t) = \prod_{i=1}^k t_i^{\beta_i}$ is concave.*

Proof. We prove by showing that its Hessian matrix $\nabla^2 g(t)$ is negative semidefinite. $\nabla^2 g(t)$ is given by

$$\frac{\partial^2 g(t)}{\partial t_i^2} = \frac{\beta_i(\beta_i-1)g(t)}{t_i^2}, \qquad \frac{\partial^2 g(t)}{\partial t_i \partial t_j} = \frac{\beta_i\beta_j g(t)}{t_i t_j},$$

and can be expressed as

$$\nabla^2 g(t) = \left(qq^{\mathrm{T}} - \mathbf{diag}(\beta_1/t_1^2,\ldots,\beta_n/t_n^2)\right)g(t)$$

where $q = [q_1, \ldots, q_k]$ and $q_i = \beta_i / t_i$. We must show that $\nabla^2 g(t) \preceq 0$, i.e., that

$$u^{\mathrm{T}} \nabla^2 g(t) u = \left(\left(\sum_{i=1}^{k} \beta_i u_i / t_i \right)^2 - \sum_{i=1}^{k} \beta_i u_i^2 / t_i^2 \right) g(t) \leq 0$$

for all $u \in \mathbb{R}^k$. Because $g(t) \geq 0$ for all t, we only need to prove

$$\left(\sum_{i=1}^{k} \beta_i u_i / t_i \right)^2 - \sum_{i=1}^{k} \beta_i u_i^2 / t_i^2 \leq 0.$$

Since β_i is positive for every i and $\sum_{i=1}^{k} \beta_i \leq 1$, we define a random variable ξ with probability $P(\xi = u_i / t_i) = \beta_i$ and $P(\xi = 0) = 1 - \sum_{i=1}^{k} \beta_i$. From basic probability theory, we have

$$\left(\sum_{i=1}^{k} \beta_i u_i / t_i \right)^2 = \left(\mathbb{E}[\xi] \right)^2 \leq \mathbb{E}\left[\xi^2 \right] = \sum_{i=1}^{k} \beta_i u_i^2 / t_i^2.$$

□

A.3 Estimating sample errors

In this part we prove Theorem 6.5. We first give some lemmas which are extended versions of lemmas that were used before to establish the sample error bounds for i.i.d. samples. In particular, the main ideas were borrowed from [17]. For any function $f \in \mathcal{H}$, we define the defect function $\mathcal{D}_{Z_{\nu^*}}(f) = \mathcal{E}(f) - \mathcal{E}_{Z_{\nu^*}}(f)$, the difference between the expected risk of f and the empirical risk of f on the FMN weighted sample Z_{ν^*}.

Lemma A.1. Let $M > 0$ and let $f : \mathbb{X} \mapsto \mathcal{Y}$ be M-bounded. Then for all $\epsilon > 0$,

$$\Pr\left(\mathcal{D}_{Z_{\nu^*}}(f) \geq -\epsilon \right) \geq 1 - \exp\left(\frac{\nu^* \epsilon^2}{2M^4} \right).$$

Proof. Note that $\Pr\left(\mathcal{D}_{Z_{\nu^*}}(f) \geq -\epsilon \right) = \Pr\left(\mathcal{E}_{Z_{\nu^*}}(f) - \mathcal{E}(f) \leq \epsilon \right)$. This lemma then follows directly from Inequality (5.3) in Theorem 5.10 by taking $\xi_i = (f(x_i) - y_i)^2$ satisfying $|\xi_i| \leq M^2$ when f is M-bounded. □

To present Lemma A.3 and A.4, we first define full measure sets.

Definition A.2 (full measure set)**.** A set $U \subseteq \mathcal{Z}$ is full measure for distribution ρ over $\mathcal{Z}$ if $\Pr_{z\sim\rho}(z \in U) = 1$.

Lemma A.3. If for $j = 1, 2$, $|f_j(x) - y| \leq M$ on a full measure set $U \subseteq \mathcal{Z}$ then, for all $Z \in U^n$

$$|\mathcal{D}_{Z_{\nu^*}}(f_1) - \mathcal{D}_{Z_{\nu^*}}(f_2)| \leq 4M\|f_1 - f_2\|_\infty.$$

Proof. Because

$$(f_1(x) - y)^2 - (f_2(x) - y)^2 = (f_1(x) + f_2(x) - 2y)(f_1(x) - f_2(x)),$$

we have

$$\begin{aligned} |\mathcal{E}(f_1) - \mathcal{E}(f_2)| &= \left| \int_{\mathcal{Z}} \rho(z)(f_1(x) + f_2(x) - 2y)(f_1(x) - f_2(x))\mathrm{d}z \right| \\ &\leq \int_{\mathcal{Z}} \rho(z)|(f_1(x) - y) + (f_2(x) - y)|\|f_1 - f_2\|_\infty \mathrm{d}z \\ &\leq 2M\|f_1 - f_2\|_\infty. \end{aligned}$$

For $Z \in U^n$, we have

$$\begin{aligned} & |\mathcal{E}_{Z_{\nu^*}}(f_1) - \mathcal{E}_{Z_{\nu^*}}(f_2)| \\ = \; & \frac{1}{\nu^*} \sum_{i=1}^{n} w_i (f_1(x_i) + f_2(x_i) - 2y_i)(f_1(x_i) - f_2(x_i) \\ \leq \; & \frac{1}{\nu^*} \sum_{i=1}^{n} w_i |(f_1(x_i) - y_i) + (f_2(x_i) - y_i)|\|(f_1 - f_2\|_\infty \\ \leq \; & 2M\|f_1 - f_2\|_\infty. \end{aligned}$$

Thus,

$$|\mathcal{D}_{Z_{\nu^*}}(f_1) - \mathcal{D}_{Z_{\nu^*}}(f_2)| = |\mathcal{E}(f_1) - \mathcal{E}_{Z_{\nu^*}}(f_1) - \mathcal{E}(f_2) + \mathcal{E}_{Z_{\nu^*}}(f_2)| \leq 4M\|f_1 - f_2\|_\infty.$$

□

Lemma A.4. Let $\mathcal{H}$ be a compact M-bounded subset of $\mathcal{C}(\mathbb{X})$. Then, for all $\epsilon > 0$,

$$\Pr\left(\sup_{f\in\mathcal{H}} \mathcal{D}_{Z_{\nu^*}}(f) \leq \epsilon \right) \geq 1 - N\left(\mathcal{H}, \frac{\epsilon}{8M}\right) \exp\left(-\frac{\nu^*\epsilon^2}{8M^4}\right).$$

Proof. Let $\{f_j\}_{j=1}^{\ell} \subset \mathcal{H}$ with $\ell = N\left(\mathcal{H}, \frac{\epsilon}{4M}\right)$ such that $\mathcal{H}$ is covered by disks D_j centered at f_j with radius $\frac{\epsilon}{4M}$. Let U be a full measure set on which $\sup_{f\in\mathcal{H}} |f(x) - y| \leq M$. Then for all $Z \in U^n$ and for all $f \in D_j$, according to Lemma A.3, we have

$$|\mathcal{D}_{Z_{\nu^*}}(f) - \mathcal{D}_{Z_{\nu^*}}(f_j)| \leq 4M\|f - f_j\|_\infty \leq 4M\frac{\epsilon}{4M} = \epsilon.$$

Consequently,

$$\sup_{f\in D_j} \mathcal{D}_{Z_{\nu^*}}(f) \geq 2\epsilon \Rightarrow \mathcal{D}_{Z_{\nu^*}}(f_j) \geq \epsilon.$$

Then we conclude that, for $j = 1, \cdots, \ell$,

$$\Pr\left(\sup_{f\in D_j} \mathcal{D}_{Z_{\nu^*}}(f) \geq 2\epsilon\right) \leq \Pr\left(\mathcal{D}_{Z_{\nu^*}}(f_j) \geq \epsilon\right) \leq \exp\left(-\frac{\nu^*\epsilon^2}{2M^4}\right).$$

The last inequality follows from Inequality (5.3) in Theorem 5.10 by taking $\xi_i = -(f_j(x_i) - y_i)^2$. In addition, one can easily see that

$$\sup_{f\in\mathcal{H}} \mathcal{D}_{Z_{\nu^*}}(f) \geq \epsilon \Leftrightarrow \exists j \leq \ell : \sup_{f\in D_j} \mathcal{D}_{Z_{\nu^*}}(f) \geq \epsilon$$

and, from the fact that the probability of a union of events is bounded by the sum of the probabilities of these events, it follows that

$$\Pr\left(\sup_{f\in\mathcal{H}} \mathcal{D}_{Z_{\nu^*}}(f) \geq \epsilon\right) \leq \sum_{j=1}^{\ell} \Pr\left(\sup_{f\in D_j} \mathcal{D}_{Z_{\nu^*}}(f) \geq \epsilon\right) \leq \ell \exp\left(-\frac{\nu^*\epsilon^2}{8M^4}\right).$$

This completes the proof. □

Lemma A.5. Suppose networked random variables $(\xi_i)_{i=1}^n$ satisfy that for all i, $\mathbb{E}[\xi_i] = \mu \geq 0$, and $|\xi_i - \mu| \leq B$ almost everywhere. Let $(w_i)_{i=1}^n$ be any FMN weight vector. If $\mathbb{E}\left[\xi_i^2\right] \leq c\mu$, then for every $\epsilon > 0$ and $0 < \alpha \leq 1$, there holds

$$\Pr\left(\frac{\mu - \frac{1}{\nu^*}\sum_{i=1}^n w_i\xi_i}{\sqrt{\mu+\epsilon}} > \alpha\sqrt{\epsilon}\right) \leq \exp\left(-\frac{\alpha^2\nu^*\epsilon}{2c + \frac{2}{3}B}\right).$$

Proof. We apply Inequality (5.2) in Theorem 5.10 by substituting the ξ_i in Inequality (5.2) with $\xi_i/\sqrt{\mu+\epsilon}$, the ϵ in Inequality (5.2) with $\alpha\sqrt{\epsilon}$, the M in Inequality (5.2) with $B/\sqrt{\mu+\epsilon}$ and the $|w|$ in Inequality (5.2) with ν^*. We get

$$\Pr\left(\frac{\mu - \frac{1}{\nu^*}\sum_{i=1}^n w_i\xi_i}{\sqrt{\mu+\epsilon}} > \alpha\sqrt{\epsilon}\right) \leq \exp\left(-\frac{\alpha^2\nu^*\epsilon}{2(\sigma^2 + B\alpha\sqrt{\epsilon}/3\sqrt{\mu+\epsilon})}\right),$$

where $\sigma^2 = \mathbb{E}\left[(\xi_i/\sqrt{\mu+\epsilon})^2\right] \leq c\mu/(\mu+\epsilon)$. The lemma then follows from observing that $c\mu/(\mu+\epsilon) \leq c$ (as $\mu \geq 0$ and $\epsilon > 0$) and $B\alpha\sqrt{\epsilon}/3\sqrt{\mu+\epsilon} \leq B/3$ (as $\mu \geq 0$, $\epsilon \geq 0$ and $0 < \alpha \leq 1$). □

Lemma A.5 can also be extended to families of functions as follows.

Lemma A.6. Let $\mathcal{G}$ be a set of functions on $\mathcal{Z}$ and $c > 0$ such that, for each $g \in \mathcal{G}$, $\mathbb{E}[g] \geq 0$, $\mathbb{E}\left[g^2\right] \leq c\mathbb{E}[g]$ and $|g - \mathbb{E}[g]| \leq B$ almost everywhere. Let $(w_i)_{i=1}^n$ be any FMN weight vector. Then for every $\epsilon > 0$ and $0 < \alpha \leq 1$, we have

$$\Pr\left(\sup_{g\in\mathcal{G}} \frac{\mathbb{E}[g] - \frac{1}{\nu^*}\sum_{i=1}^n w_i g(z_i)}{\sqrt{\mathbb{E}[g] + \epsilon}} \geq 4\alpha\sqrt{\epsilon}\right) \leq N(\mathcal{G}, \alpha\epsilon)\exp\left(-\frac{\alpha^2\nu^*\epsilon}{2c + \frac{2}{3}B}\right).$$

Proof. Let $\{g_j\}_{j=1}^J \subset \mathcal{G}$ with $J = N(\mathcal{G}, \alpha\epsilon)$ be such that $\mathcal{G}$ is covered by balls in $\mathcal{C}(\mathcal{Z})$ centered at g_j with radius $\alpha\epsilon$.

Applying Lemma A.5 to $\xi_i = g_j(z_i)$ for each j, we have

$$\Pr\left(\frac{\mathbb{E}[g_j] - \frac{1}{\nu^*}\sum_{i=1}^n w_i g_j(z_i)}{\sqrt{\mathbb{E}[g_j] + \epsilon}} \geq \alpha\sqrt{\epsilon}\right) \leq \exp\left(-\frac{\alpha^2\nu^*\epsilon}{2c + \frac{2}{3}B}\right).$$

For each $g \in \mathcal{G}$, there is some j such that $||g - g_j||_{\mathcal{C}(\mathcal{Z})} \leq \alpha\epsilon$. Then $|\frac{1}{\nu^*}\sum_{i=1}^n w_i g(z_i) - \frac{1}{\nu^*}\sum_{i=1}^n w_i g_j(z_i)|$ and $|\mathbb{E}[g] - \mathbb{E}[g_j]|$ are both bounded by $\alpha\epsilon$. Hence, as $\frac{\sqrt{\epsilon}}{\sqrt{\epsilon+\mathbb{E}[g]}} \leq 1$,

$$\frac{|\frac{1}{\nu^*}\sum_{i=1}^n w_i g(z_i) - \frac{1}{\nu^*}\sum_{i=1}^n g_j(z_i)|}{\sqrt{\mathbb{E}[g] + \epsilon}} \leq \alpha\sqrt{\epsilon}$$

and

$$\frac{|\mathbb{E}[g] - \mathbb{E}[g_j]|}{\sqrt{\mathbb{E}[g] + \epsilon}} \leq \alpha\sqrt{\epsilon}.$$

The latter implies that

$$\begin{aligned}\mathbb{E}[g_j] + \epsilon &= \mathbb{E}[g_j] - \mathbb{E}[g] + \mathbb{E}[g] + \epsilon \leq \alpha\sqrt{\epsilon}\sqrt{\mathbb{E}[g] + \epsilon} + (\mathbb{E}[g] + \epsilon) \\ &\leq \sqrt{\epsilon}\sqrt{\mathbb{E}[g] + \epsilon} + (\mathbb{E}[g] + \epsilon) \leq 2(\mathbb{E}[g] + \epsilon).\end{aligned}$$

It follows that $\sqrt{\mathbb{E}[g_j]+\epsilon} \leq 2\sqrt{\mathbb{E}[g]+\epsilon}$. We have thus seen that $\frac{\mathbb{E}[g]-\frac{1}{\nu^*}\sum_{i=1}^n w_i g(z_i)}{\sqrt{\mathbb{E}[g]+\epsilon}} \geq 4\alpha\sqrt{\epsilon}$ implies

$$\begin{aligned}
&\frac{\mathbb{E}[g_j]-\frac{1}{\nu^*}\sum_{i=1}^n w_i g_j(z_i)}{\sqrt{\mathbb{E}[g]+\epsilon}} \\
= \;& \frac{\mathbb{E}[g_j]-\mathbb{E}[g]}{\sqrt{\mathbb{E}[g]+\epsilon}} - \frac{\frac{1}{\nu^*}\sum_{i=1}^n w_i g_j(z_i) - \frac{1}{\nu^*}\sum_{i=1}^n w_i g(z_i)}{\sqrt{\mathbb{E}[g]+\epsilon}} \\
& + \frac{\mathbb{E}[g]-\frac{1}{\nu^*}\sum_{i=1}^n w_i g(z_i)}{\sqrt{\mathbb{E}[g]+\epsilon}} \\
\geq \;& \frac{-2a\epsilon}{\sqrt{\mathbb{E}[g]+\epsilon}} + 4a\sqrt{\epsilon} \geq 2\alpha\sqrt{\epsilon}
\end{aligned}$$

and hence $\frac{\mathbb{E}[g_j]-\frac{1}{\nu^*}\sum_{i=1}^n w_i g_j(z_i)}{\sqrt{\mathbb{E}[g_j]+\epsilon}} \geq \alpha\sqrt{\epsilon}$. Therefore,

$$\begin{aligned}
&\Pr\left(\sup_{g\in\mathcal{G}} \frac{\mathbb{E}[g]-\frac{1}{\nu^*}\sum_{i=1}^n w_i g(z_i)}{\sqrt{\mathbb{E}[g]+\epsilon}} \geq 4\alpha\sqrt{\epsilon}\right) \\
&\leq \sum_{j=1}^J \Pr\left(\frac{\mathbb{E}[g_j]-\frac{1}{\nu^*}\sum_{i=1}^n w_i g_j(z_i)}{\sqrt{\mathbb{E}[g_j]+\epsilon}} \geq \alpha\sqrt{\epsilon}\right)
\end{aligned}$$

which is bouned by $J\cdot\exp\left(-\frac{\alpha^2\nu^*\epsilon}{2c+\frac{2}{3}B}\right)$. □

Let $\mathcal{L}^2_\rho(\mathbb{X})$ be a Banach space with the norm $\|f\|_{\mathcal{L}^2_\rho(\mathbb{X})} = \left(\int_{\mathbb{X}} |f(x)^2|\rho_{\mathbb{X}}(x)\mathrm{d}x\right)^{\frac{1}{2}}$. where $\rho_{\mathbb{X}}(x) = \prod_{i=1}^k x^{(i)}$. We define the error in $\mathcal{H}$ of a function $f \in \mathcal{H}$,

$$\mathcal{E}_{\mathcal{H}}(f) = \mathcal{E}(f) - \mathcal{E}(f_{\rho,\mathcal{H}})$$

which is always nonnegative.

Lemma A.7. Let $\mathcal{H}$ be a convex subset of $\mathcal{C}(\mathbb{X})$ such that $f_{\rho,\mathcal{H}}$ exists. Then $f_{\rho,\mathcal{H}}$ is unique as an element in $\mathcal{L}^2_\rho(\mathbb{X})$ and for all $f \in \mathcal{H}$,

$$\int_{\mathbb{X}} (f_{\rho,\mathcal{H}}(x) - f(x))^2 \rho_{\mathbb{X}}(x)dx \leq \mathcal{E}_{\mathcal{H}}(f).$$

In particular, if $\rho_{\mathbb{X}}(x)$ is not degenerate then $f_{\rho,\mathcal{H}}$ is unique in $\mathcal{H}$.

Proof. The proof can be found in [17] (Lemma 3.16). □

Proof of Theorem 6.5 For every function $f \in \mathcal{H}$, we define a function

$$g_f(x,y) = (f(x)-y)^2 - (f_{\rho,\mathcal{H}}(x)-y)^2.$$

We define $\mathcal{G}$ as the set of all functions g_f with $f \in \mathcal{H}$. For any function $g_f \in \mathcal{G}$, we have

$$\mathbb{E}_{z\sim\rho}[g_f] = \mathcal{E}_{\mathcal{H}}(f) \geq 0. \tag{A.2}$$

We first show that the two preconditions of Lemma A.6 are true (for $B = 2M^2$ and $c = 4M^2$):

1. $|g_f - \mathbb{E}_{z\sim\rho}[g_f]| \leq 2M^2$
2. $\mathbb{E}_{z\sim\rho}\left[g_f^2\right] \leq 4M^2\mathbb{E}_{z\sim\rho}[g_f]$.

First, since $\mathcal{H}$ is M-bounded, we have that $-M^2 \leq g_f(z) \leq M^2$ holds almost everywhere. It follows that $|g_f - \mathbb{E}_{z\sim\rho}[g_f]| \leq 2M^2$ holds almost everywhere. This is the first precondition above. Second, one can easily see that

$$g_f(z) = (f(x) - f_{\rho,\mathcal{H}}(x))[(f(x)-y) + (f_{\rho,\mathcal{H}}(x)-y)].$$

It follows that $|g_f(z)| \leq 2M|f(x) - f_{\rho,\mathcal{H}}(x)|$ holds almost everywhere. Then, $\mathbb{E}_{z\sim\rho}\left[g_f^2\right] \leq 4M^2\mathbb{E}_{x\sim\rho_{\mathbb{X}}}\left[(f(x)-f_{\rho,\mathcal{H}}(x))^2\right] = 4M^2\int_{\mathbb{X}}(f(x)-f_{\rho,\mathcal{H}}(x))^2\,\rho_{\mathbb{X}}(x)\mathrm{d}x$. Together with Lemma A.7 this implies that $\mathbb{E}_{z\sim\rho}\left[g_f^2\right] \leq 4M^2\mathcal{E}_{\mathcal{H}}(f) = c\mathbb{E}_{z\sim\rho}[g_f]$ with $c = 4M^2$. Hence, all the conditions of Lemma A.6 hold and we get that for every $\epsilon > 0$ and $0 < \alpha \leq 1$,

$$\Pr\left(\sup_{g\in\mathcal{G}} \frac{\mathbb{E}[g] - \frac{1}{\nu^*}\sum_{i=1}^n w_i g(z_i)}{\sqrt{\mathbb{E}[g]+\epsilon}} \geq 4\alpha\sqrt{\epsilon}\right) \leq N(\mathcal{G},\alpha\epsilon)\exp\left(-\frac{\alpha^2\nu^*\epsilon}{2.4M^2 + \frac{2}{3}2M^2}\right). \tag{A.3}$$

Remind from Equation (A.2) that $\mathbb{E}[g_f] = \mathcal{E}_{\mathcal{H}}(f)$. We also define

$$\mathcal{E}_{\mathcal{H},Z_{\nu^*}}(f) = \frac{1}{\nu^*}\sum_{i=1}^n w_i g_f(z_i) = \frac{1}{\nu^*}\sum_{i=1}^n w_i(f(x)-y)^2 - \frac{1}{\nu^*}\sum_{i=1}^n w_i(f_{\rho,\mathcal{H}}(x)-y)^2$$

Furthermore, we take $\alpha = \sqrt{2}/8$. Substituting all these into Inequality (A.3) we get that for all $\epsilon > 0$,

$$\Pr\left(\sup_{f\in\mathcal{H}} \frac{\mathcal{E}_{\mathcal{H}}(f) - \mathcal{E}_{\mathcal{H},Z_{\nu^*}}(f)}{\sqrt{\mathcal{E}_{\mathcal{H}}(f)+\epsilon}} \geq 4\frac{\sqrt{2}}{8}\sqrt{\epsilon}\right) \leq N\left(\mathcal{G},\frac{\sqrt{2}}{8}\epsilon\right)\exp\left(-\frac{\left(\frac{\sqrt{2}}{8}\right)^2\nu^*\epsilon}{28M^2/3}\right).$$

As this holds for the supremum over f, it also holds for $f = f_{Z_{\nu^*},\mathcal{H}}$: for all $\epsilon > 0$,

$$\Pr\left(\frac{\mathcal{E}_{\mathcal{H}}(f_{Z_{\nu^*},\mathcal{H}}) - \mathcal{E}_{\mathcal{H},Z_{\nu^*}}(f_{Z_{\nu^*},\mathcal{H}})}{\sqrt{\mathcal{E}_{\mathcal{H}}(f_{Z_{\nu^*},\mathcal{H}}) + \epsilon}} \geq \sqrt{\frac{\epsilon}{2}}\right) \leq N\left(\mathcal{G}, \frac{\sqrt{2}}{8}\epsilon\right) \exp\left(-\frac{\nu^*\epsilon}{896M^2/3}\right).$$

The definition of $f_{Z_{\nu^*},\mathcal{H}}$ (Eq. (6.2)) tells us that $\mathcal{E}_S(Z_{\nu^*}) = \mathcal{E}_{\mathcal{H}}(f_{Z_{\nu^*},\mathcal{H}})$ and $\mathcal{E}_{\mathcal{H},Z_{\nu^*}}(f_{Z_{\nu^*},\mathcal{H}}) \leq 0$. It follows that (we also upper-bound 896/3 by 300)

$$\forall\epsilon > 0, \Pr\left(\frac{\mathcal{E}_S(Z_{\nu^*})}{\sqrt{\mathcal{E}_S(Z_{\nu^*}) + \epsilon}} \geq \sqrt{\frac{\epsilon}{2}}\right) \leq N\left(\mathcal{G}, \frac{\sqrt{2}}{8}\epsilon\right) \exp\left(-\frac{\nu^*\epsilon}{300M^2}\right).$$

It is easy to see that $\mathcal{E}_S(Z_{\nu^*}) \geq \epsilon$ implies $\frac{\mathcal{E}_S(Z_{\nu^*})}{\sqrt{\mathcal{E}_S(Z_{\nu^*})+\epsilon}} \geq \sqrt{\frac{\epsilon}{2}}$, so

$$\forall\epsilon > 0, \Pr\left(\mathcal{E}_S(Z_{\nu^*}) \geq \epsilon\right) \leq N\left(\mathcal{G}, \frac{\sqrt{2}}{8}\epsilon\right) \exp\left(-\frac{\nu^*\epsilon}{300M^2}\right).$$

Finally, the inequality $\|g_{f_1} - g_{f_2}\|_{\mathcal{C}(\mathcal{Z})} = \|f_1(x) - f_2(x)[(f_1(x) - y) + (f_2(x) - y)]\|_{\mathcal{C}(\mathcal{Z})} \leq 2M\|f_1 - f_2\|_{\mathcal{C}(\mathbb{X})}$, tells us that

$$N(\mathcal{G}, \frac{\sqrt{2}\epsilon}{8}) \leq N(\mathcal{H}, \frac{\sqrt{2}\epsilon}{16M}) \leq N(\mathcal{H}, \frac{\epsilon}{12M}).$$

This completes our proof. □

Bibliography

[1] Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of SIGMOD'93* (1993), pp. 207–216.

[2] Arcones, M. A. A Bernstein-type inequality for U-statistics and U-processes. *Statistics & Probability Letters 22*, 3 (1995), 239–247.

[3] Barabási, A.-L., and Reka, A. Emergence of scaling in random networks. *Science 286* (1999), 509–512.

[4] Berlingerio, M., Bonchi, F., Bringmann, B., and Gionis, A. Mining graph evolution rules. In *Proceedings of ECML/PKDD'09* (2009), pp. 115–130.

[5] Borgelt, C. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2012), 437–456.

[6] Boyd, S., and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

[7] Bradley, R. C. Basic properties of strong mixing conditions, a survey and some open questions. *Probability Surveys 2* (2005), 107–144.

[8] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[9] Bringmann, B., and Nijssen, S. What is frequent in a single graph? In *Proceedings of PAKDD'08,* (2008), pp. 858–863.

[10] Calders, T., Ramon, J., and Dyck, D. All normalized anti-monotonic overlap graph measures are bounded. *Data Mining and Knowledge Discovery 23(3)* (2011), 503–548.

[11] Chakrabarti, D., and Faloutsos, C. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys 38(1)* (2006), 1–69.

[12] Chan, T., Chang, K., and Raman, R. An SDP primal-dual algorithm for approximating the Lovász-theta function. In *Proceedings of the IEEE ISIT'09* (2009), pp. 2808–2812.

[13] Chan, Y., and Lau, L. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical Programming 135*, 1-2 (2012), 123–148.

[14] Chernoff, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics 23.4* (1952), 493–507.

[15] Chung, F. R., and Lu, L. *Complex Graphs and Networks*, vol. 107. American mathematical society, 2006.

[16] Colbourn, C. J., and Dimitz, J. H., Eds. *Handbook of Combinatorial Designs*. CRC press, 2010.

[17] Cucker, F., and Zhou, D.-X. *Learning Theory: An Approximation Theory Viewpoint*. Cambridge University Press, 2007.

[18] Dreweke, A., W'orlein, M., Fischer, I., Schell, D., Meinl, T., and Philippsen, M. Graph-based procedural abstraction. In *Proceedings of the International Symposium on Code Generation and Optimization'07* (2007), pp. 259–270.

[19] Engebretsen, L., and Holmerin, J. Clique is hard to approximate within n 1-o(1). In *Automata, Languages and Programming*, U. Montanari, J. Rolim, and E. Welzl, Eds., vol. 1853 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 2–12.

[20] Fagin, R. Probabilities on finite models. *Journal of Symbolic Logic 41(1)* (1976), 50–58.

[21] Feige, U., Goldwasser, S., Lovász, L., Safra, S., and Szegedy, M. Approximating clique is almost NP-Complete. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on* (1991), pp. 2–12.

[22] Fiedler, M., and Borgelt, C. Support computation for mining frequent subgraphs in a single graph. In *Proceedings of the workshop on Mining and Learning with Graphs (MLG'07)* (2007).

[23] Fierens, D., Blockeel, H., Bruynooghe, M., and Ramon, J. Logical Bayesian networks and their relation to other probabilistic logical models. In *Inductive Logic Programming* (2005), Springer Berlin Heidelberg, pp. 121–135.

[24] Fortunato, S. Community detection in graphs. *Physics Reports 486*, 3 (2010), 75–174.

[25] Friedman, N., Getoor, L., Koller, D., and Pfeffer., A. Learning probabilistic relational models. In *IJCAI* (1999), vol. 99, pp. 1300–1309.

[26] Garey, M., and Johnson, D. *Computers and Intractibility, A Guide to the Theory of NP-Completeness*. W. H. Freeman Company, 1979.

[27] Getoor, L., and Taskar, B., Eds. *Introduction to Statistical Relational Learning*. MIT press, 2007.

[28] Gjoka, M., Kurant, M., Butts, C. T., and Markopoulou, A. Walking in Facebook: A case study of unbiased sampling of OSNs. In *Proceedings of IEEE INFOCOM '10* (San Diego, CA, March 2010).

[29] Guo, Z.-C., and Shi, L. Classification with non-iid sampling. *Mathematical and Computer Modelling 54.5* (2011), 1347–1364.

[30] Hastad, J. Clique is hard to approximate within n 1-&epsiv. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on* (1996), IEEE, pp. 627–636.

[31] He, J., Liu, Y., and Richard, L. Graph-based transfer learning. In *Proceedings of the 18th ACM conference on Information and knowledge management. ACM* (2009), pp. 937–946.

[32] Hoeffding, W. A class of statistics with asymptotically normal distributions. *Annals of Statistics 19* (1948), 293–325.

[33] Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58.301* (1963), 13–30.

[34] Iyengar, G., Phillips, D., and Stein, C. Approximating semidefinite packing programs, 2011.

[35] Jaeger, M. Relational Bayesian networks. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence* (1997), Morgan Kaufmann Publishers, pp. 266–273.

[36] Janson, S. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms 24.3* (2004), 234–248.

[37] Jensen, J. L. W. V. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica 30*, 1 (1906), 175–193.

[38] Kendall, M. G. A New Measure of Rank Correlation. *Biometrika 30*, 1/2 (1938), 81–93.

[39] Kibriya, A., and Ramon, J. Nearly exact mining of frequent trees in large networks. In *Proceedings of ECML-PKDD 2012* (2012), pp. 426–441.

[40] Knuth, D. The sandwich theorem. *The Electronic Journal of Combinatorics 1* (1994), 1–48.

[41] Kuramochi, M., and Karypis, G. Finding frequent subgraph patterns in a large sparse graph. *Data Mining and Knowledge Discovery 11(3)* (2005), 243–271.

[42] Liben-Nowell, D., and Kleinberg, J. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology 58*, 7 (2007), 1019–1031.

[43] Liu, R. Y. Bootstrap procedures under some non-iid models. *The Annals of Statistics 16*, 4 (1988), 1696–1708.

[44] Lovász, L. On the ratio of optimal integral and fractional covers. *Discrete Mathematics 13.4* (1975), 383–390.

[45] Marcus, R., Sandran, E., and Bernt, S. Transfer learning in a transductive setting. In *In Advances in neural information processing systems* (2013), pp. 46–54.

[46] Matousek, J., and Nesetril, J. *Invitation to Discrete Mathematics*. Oxford University Press, 1998.

[47] Modha, D. S., and Masry, E. Minimum complexity regression estimation with weakly dependent observations. *Information Theory, IEEE Transactions on 42.6* (1996), 2133–2145.

[48] Nijssen, S., and Fromont, E. Optimal constraint-based decision tree induction from itemset lattices. *Data Mining and Knowledge Discovery 21* (2010), 9–51.

[49] Oosterhoff, K., and van Zwet, W. R. Wassily Hoeffding's work in the Sixties. In *The Collected Works of Wassily Hoeffding*, N. Fisher and P. Sen, Eds., Springer Series in Statistics. Springer New York, 1994, pp. 3–15.

[50] Ralaivola, L., Szafranski, M., and Stempfel, G. Chromatic PAC-Bayes bounds for non-iid data. In *AISTATS 2009* (2009), vol. 5, pp. 416–423.

[51] Ramon, J., Comendant, C., Haghir Chehreghani, M., and Wang, Y. Graph and network pattern mining. In *Social Media and Social Computing*, M.-F. Moens, J. Li, and T.-S. Chua, Eds. CRC Press, 2014, ch. Graph and network pattern mining.

[52] Richardson, M., and Domingos, P. Markov logic networks. *Machine learning 62*, 1-2 (2006), 107–136.

[53] Robins, G., Pattison, P., Kalish, Y., and Lusher, D. An introduction to exponential random graph (p*) models for social networks. *Social Networks 29*, 2 (2007), 173–191.

[54] Rossi, R. A., McDowell, L. K., Aha, D. W., and Neville, J. Transforming graph data for statistical relational learning. *Journal of Artificial Intelligence Research 45* (2012), 363–441.

[55] Scheffe, H. *The Analysis of Variance*, vol. 72. John Wiley & Sons, 1999.

[56] Schrijver, A. A comparison of the Delsarte and Lovász bounds. *IEEE Transactions on Information Theory 25(4)* (1979), 425–429.

[57] Shawe-Taylor, J., Bartlett, P. L., Williamson, R. C., and Anthony, M. Structural risk minimization over data dependent hierarchies. *Information Theory, IEEE Transactions on 44*, 5 (Sept. 1998), 1926–1940.

[58] Shen, E., and Yu, T. Mining frequent graph patterns with differential privacy. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), ACM, pp. 545–553.

[59] Shen-Orr, S. S., Milo, R., Mangan, S., and Alon, U. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature Genetics 31*, 1 (2002), 64–68.

[60] Sion, M. On general minimax theorems. *Pacific Journal of Mathematics 8*, 1 (1958), 171–176.

[61] Sun, H., and Wu, Q. Regularized least square regression with dependent samples. *Advances in Computational Mathematics 32.2* (2010), 175–189.

[62] Suykens, J. A., and Vandewalle, J. Least squares support vector machine classifiers. *Neural Processing Letters 9*, 3 (1999), 293–300.

[63] Tsuda, K. Optimal hyperplane classifier based on entropy number bound. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)* (1999), vol. 1.4, pp. 419–424.

[64] Usunier, N., Amini, M.-r., and Gallinari, P. Generalization error bounds for classifiers trained with interdependent data. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)* (2006), MIT Press, pp. 1369–1376.

[65] Vanetik, N., Gudes, E., and Shimony, S. Computing frequent graph subgraph patterns from semistructured data. In *Proceeding of the IEEE International Conference on Data Mining (ICDM'02)* (2002), pp. 458–465.

[66] Vanetik, N., Shimony, S., and Gudes, E. Support measures for graph data. *Data Mining and Knowledge Discovery 13(2)* (2006), 243–260.

[67] Vapnik, V., Levin, E., and Cun, Y. L. Measuring the VC-dimension of a learning machine. *Neural Computation 6*, 5 (1994), 851–876.

[68] Vasishth, S., and Broe, M. Analysis of variance (ANOVA). In *The Foundations of Statistics: A Simulation-based Approach*. Springer Berlin Heidelberg, 2011, pp. 97–126.

[69] Wang, T., Neville, J., Gallagher, B., and Eliassi-Rad, T. Correcting bias in statistical tests for network classifier evaluation. In *Proceedings of ECML/PKDD* (2011), vol. 6913 of *LNCS*, pp. 506–521.

[70] Wang, Y., and Ramon, J. An efficiently computable support measure for frequent subgraph pattern mining. In *Proceedings of ECMLPKDD* (2012), pp. 362–377.

[71] Wang, Y., Ramon, J., and Fannes, T. An efficiently computable subgraph pattern support measure: Counting independent observations. *Data Mining and Knowledge Discovery 27*, 3 (2013), 444–477.

[72] Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin 1*, 6 (1945), 80–83.

[73] Yan, X. *Linear Regression Analysis: Theory and Computing*. World Scientific, 2009.

[74] Yan, X., and Han, J. gSpan: Graph-based substructure pattern. In *IEEE International Conference on Data Mining (ICDM)* (2002).

[75] Zhou, D.-X. The covering number in learning theory. *Journal of Complexity 18*, 3 (2002), 739–767.

[76] Zhou, F., Malher, S., and Toivonen, H. Network simplification with minimal loss of connectivity. In *IEEE 10th International Conference on Data Mining (ICDM)* (2010).

[77] Zou, Z., Li, J., Gao, H., and Zhang, S. Mining frequent subgraph patterns from uncertain graph data. *Knowledge and Data Engineering, IEEE Transactions on 22*, 9 (2010), 1203–1218.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
MACHINE LEARNING GROUP
Celestijnenlaan 200A
B-3001 Heverlee
yuyi.wang@dept.kuleuven.be
http://www.dept.cs.kuleuven.be