LEARNING WITH STRUCTURES

By

Yang Zhou

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2010

ABSTRACT

LEARNING WITH STRUCTURES

By

Yang Zhou

In this dissertation we discuss learning with structures, which appears frequently in both machine learning theories and applications. First we review existing structure learning algorithms, then we study several specifically interesting problems. The first problem we study is the structure learning of dynamic systems. We investigate using dynamic Bayesian networks to reconstruct functional cortical networks from the spike trains of neurons. Next we study structure learning from matrix factorization, which has been a popular research area in recent years. We propose an efficient non-negative matrix factorization algorithm which derives not only the membership assignments to the clusters but also the interaction strengths among the clusters. Following that we study the hierarchical and grouped structure in regularization. We propose a novel regularizer called group lasso which introduces competitions among variables in groups, and thus results in sparse solutions. Finally we study the sparse structure in a novel problem of online feature selection, and propose an online learning algorithm that only needs to sense a small number of attributes before the reliable decision can be made.

ACKNOWLEDGMENTS

First and foremost I want to thank my advisor Dr. Rong Jin. It has been an honor to be his Ph.D. student. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. The joy and enthusiasm he has for his research was contagious and motivational for me, even during tough times in the Ph.D. pursuit. I am also thankful for the excellent example he has provided as a successful researcher.

I would like to thank Dr. Christina Chan who served as my co-advisor. She has advised me in my biostatistics related research projects. Together with Dr. Jin, she helped me a lot during the most difficult time during my Ph.D. study. I am deeply impressed by her energy and passion for academic research.

I also thank the other two committee members, Dr. Pang-Ning Tan and Dr. Sarat Dass, for their critical comments, which enabled me to notice the weaknesses of my dissertation and make the necessary improvements according to their comments.

The research related with reconstructing gene regulatory networks in this dissertation was a collaborative result with Chan Group. I am thankful to Zheng Li, Xuerui Yang, Linxia Zhang, Shireesh Srivastava and Xuewei Wang for their help and cooperation.

The research related with reconstructing cortical networks was the result of productive collaboration with Dr. Karim G. Oweiss and his student Dr. Seif Eldawlatly.

I am thankful to my previous advisor Dr. Juyang Weng for his support and valuable suggestions.

Special thanks to my labmates, Zhengping Ji, Wei Tong, Tianbao Yang, Yi Liu, Jinfeng Yi, Fengjie Li, Hamed Valizadegan, Mehrdad Mahdavi, Ming Wu and Ming Wu, Chen Zhang,

Feilong Chen and Nan Zhang. It has been a pleasure discussing with them for various problems, which gave me enlightenment and inspiration for my research. I also used to hang out with them in spare time, which provides a lot of fun for the long and sometimes boring life in East Lansing area.

Many people on the faculty and staff of the Computer Science and Engineering department at MSU assisted and encouraged me in various ways during my studies. I am especially grateful to Prof. George Stockman, Prof. Anil Jain, Dr. Eric Torng, Linda Moore and Norma Teague.

My most sincere and most heartfelt thanks go to my wife for her unlimited and unconditional encouragement, support and love. Learning to love you and to receive your love make me a better person. You have my everlasting love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

In this dissertation we discuss learning with structures, which appears frequently in both machine learning theories and applications. Structures can be found in many study areas. In biology, the gene regulatory network is a collection of DNAs in a cell. These DNAs interact with each other indirectly through RNA and protein expression, and thereby governing the rates at which the genes are transcribed into mRNA. In system neuroscience, numerous functional neuroimaging studies suggest that cortical regions selectively couple to each other. These interconnected regions orchestrate together and mediate perception, learning, sensory and motor processing. It is essential to reconstruct this neural network in order to understand the internal mechanism.

A large amount of research work has been devoted to structure learning and many models and algorithms have been developed, including, for example, Bayesian networks, Gaussian graphical models, Markov Random Field, group lasso regularization, etc. Despite significant efforts made in learning with structures, there are a number of challenges remain to be addressed:

- In many applications we are interested in the temporal effects in the *dynamic structures*. Note that the term "dynamic" means that we are modeling a dynamic system, not that the structure changes over time. For example, in functional neural networks, the presynaptic and postsynaptic neurons are connected by channels that are capable of passing electrical current, causing electric spiking (firing) in the presynaptic neuron to influence, either excitatory or inhibitory, the spiking of the postsynaptic neuron. Although this kind of dynamic structures are found in many applications, learning

with dynamic structures is an area that is less studied.

- Many structures in our study are large-scale, leading to computational challenges with the existing algorithms. For example, the gene regulatory networks may be consisted of hundreds, thousands or even more DNAs. In order to study in the mechanism of cortical networks we may have millions of neurons at proposal. However, most of the existing algorithms are designed to handle either small or medium scale datasets. For example, The Bayesian network learning algorithm can usually learn structures with a few hundred nodes at most. The computational complexity grows exponentially with the number of nodes, which hinders us from exploring the structures of large sets of variables.

- Although the structure information often appears explicitly in many situations like the conditional dependency in a Bayesian network which can be intuitively visualized using a directed graph, it may be implicit sometimes and thus difficult to explore. Many algorithms including group lasso and Composite Absolute Penalties (CAP) have been proposed to exploit the information embedded in the group structure. However, these algorithms are restricted to discovering positive correlations among the variables within groups only. Specifically, they assume that if a few variables in a group are important, then most of the variables in the same group should also be important. However, in many real-world applications, we may come to the opposite observation, i.e., variables in the same group exhibit negative correlations by competing with each other. For instance, in visual object recognition, the signature visual patterns of different objects tend to be negatively correlated, i.e., visual patterns valuable for recognizing one object tent to be less useful for other objects. It remains a question of how to infer this kind of negative correlations in implicit group structures.

- The sparse structure is tempting in many applications, one of such is online learning. Most of the existing online learning algorithms have at least one weight for every

feature. Although some algorithms are proposed recently to introduce sparsity to the model, they lack a hard constraint on the number of non-zero features, and thus need access to the full information of the sample in each iteration. On the other hand, sparse representation of the online learning model has many advantages. For example, the time complexity and space complexity is significantly reduced. The constraints on sparsity may also avoid the overfitting problem. In some applications the acquisition of samples is costly, and thus the number of features available is limited by our budget.

This dissertation is devoted to tackling these challenges in learning with structures. Specifically, this dissertation will present research work in the following topics: (i) Application of dynamic structure learning in reconstructing cortical networks. (ii) Developing algorithms for structure learning that can efficiently handle large scale data sets. (iii) Developing a new kind of regularization mechanism for explicitly exploring the negative correlations in implicit group structures. (iv) Developing new online feature selection algorithms with hard sparse constraints on number of features.

The rest of this dissertation is organized as following. Chapter 2 gives a comprehensive review of existing structure learning algorithms. In particular, structure learning of probabilistic graphical models is given special attention since it is representative for many structure learning problems, and this area has been well studies with many models and algorithms developed. In Chapter 3, we discuss dynamic structure learning from time-series or sequential data. We give a detailed introduction to Hidden Markov Models. As a typical time-state model, it has been widely used in various engineering fields for several decades. The extension of HMMs naturally lead to dynamic Bayesian networks. We study the application of dynamic Bayesian networks to the reconstruction of functional cortical networks. In Chapter 4, we study structure learning from matrix factorization, which has been a popular research area in recent years. An efficient non-negative matrix factorization algorithm is proposed. Unlike the traditional matrix factorization methods for clustering, the proposed algorithm not only derives the membership assignment to the clusters, but also computes the

strength of interactions among the clusters. The proposed algorithm is applied to the reconstruction of gene regulatory networks, and it shows superior performance in our experiments compared to state-of-the-art algorithms. In Chapter 5 we study the implicit and indirect structures resulted from regularization. We are particularly interested in the grouped and hierarchical regularization, and we propose a novel exclusive lasso regularizer. Unlike the group lasso which assumes positive correlations in the groups, namely, if one variable in a group is believed to be important, then all variables in the group should be important, our proposed exclusive lasso regularizer introduces competitions among variables in groups, resulting in a sparse solution where the most prominent variable in a group stands out and the others diminish. We apply this regularization to multi-task learning in our experiments, and it show superior performance compared with other state-of-the-art algorithms. In Chapter 6, we further investigate the sparse structure in online learning, specifically in a novel problem of online feature selection. Most online learning studies assume that the learning has full access to all input features. However, in many real world applications, it is expensive, either computationally or money wise, to acquire and use all the input attributes. In this case it is desirable to develop online learning algorithms that only need to sense a small number of attributes before the reliable decision can be made. We make a first step towards solving this problem, and develop theories and algorithms for sparse online feature selection. Specifically, we develop the general algorithms for sparse online feature selection, and examine their theoretic properties such as the upper and lower bounds for the regret bound. We evaluate the proposed algorithms by extensive experiments on benchmark datasets.

CHAPTER 2

Learning with Structures: a Comprehensive Review

In this chapter we review the existing models and algorithms for learning with structures.

An important family of structures are the probabilistic graphical models which combine the graph theory and probability theory to give a multivariate statistical modeling. Many graphical models have been developed so far, e.g., Bayesian networks, Gaussian graphical models, Markov random fields, among others. The structure learning of probabilistic graphical models has been well studied and many algorithms have been proposed. We will review the two most popular approaches: constraint-based algorithms and score-based algorithms.

We will also review other structure learning algorithms for different models. These algorithms can roughly be categorized into regression based approaches, matrix factorization based approaches, group regularization based approaches, and hybrid methods.

2.1 Structure Learning of Probabilistic Graphical Models

Probabilistic graphical models combine the graph theory and probability theory to give a multivariate statistical modeling. They provide a unified description of uncertainty using probability and complexity using the graphical model. Especially, graphical models provide the following several useful properties:

- Graphical models provide a simple and intuitive interpretation of the structures of

probabilistic models. On the other hand, they can be used to design and motivate new models.

- Graphical models provide additional insights into the properties of the model, including the conditional independence properties.
- Complex computations which are required to perform inference and learning in sophisticated models can be expressed in terms of graphical manipulations, in which the underlying mathematical expressions are carried along implicitly.

The graphical models have been applied to a large number of fields, including bioinformatics, social science, control theory, image processing, marketing analysis, among others. However, structure learning for graphical models remains an open challenge, since one must cope with a combinatorial search over the space of all possible structures.

2.1.1 Preliminaries

We will first define a set of notations which will be used in this chapter. We represent a graph as $G = \langle V, E \rangle$ where $V = \{v_i\}$ is the set of nodes in the graph and each node corresponds to a random variable $x_i \in \mathcal{X}$. $E = \{(v_i, v_j) : i \neq j\}$ is the set of edges. In a directed graph, if there is an edge $E_{i,j}$ from v_i to v_j, then v_i is a parent of node v_j and v_j is a child of node v_i. If there is no cycle in a directed graph, we call it a *directed acyclic graph* (DAG). The number of nodes and number of edges in a graph are denoted by $|V|$ and $|E|$ respectively. $\pi(i)$ represent all the parents of node v_i in a graph. $U = \{x_1, \cdots, x_n\}$ denotes the finite set of discrete random variables where each variable x_i may take on values from a finite domain. $Val(x_i)$ denotes the set of values that variable x_i may attain, and $|x_i| = |Val(x_i)|$ denotes the cardinality of this set. In probabilistic graphical network, the Markov blanket ∂v_i [Pearl, 1988] of a node v_i is defined to be the set of nodes in which each has an edge to v_i, i.e., all v_j such that $(v_i, v_j) \in E$. The Markov assumption states that in a probabilistic graphical network, every set of nodes in the network is conditionally independent of v_i when

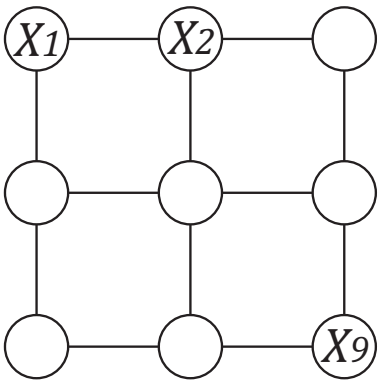


Figure 2.1: An Ising model with 9 nodes.

conditioned on its Markov blanket ∂v_i. Formally, for distinct nodes v_i and v_k,

$$P(v_i|\partial v_i \cap v_k) = P(v_i|\partial v_i)$$

The Markov blanket of a node gives a localized probabilistic interpretation of the node since it identifies all the variables that shield off the node from the rest of the network, which means that the Markov blanket of a node is the only information necessary to predict the behavior of that node. A DAG G is an I-Map of a distribution P if all the Markov assumptions implied by G are satisfied by P.

Theorem 2.1.1. (Factorization Theorem) *If G is an I-Map of P , then*

$$P(x_1, \cdots, x_n) = \prod_i P(x_i|x_{\pi(i)})$$

According to this theorem, we can represent P in a compact way when G is sparse such that the number of parameter needed is linear in the number of variables. This theorem is true in the reverse direction.

The set X is d-separated from set Y given set Z if all paths from a node in X to a node in Y are blocked given Z.

The graphical models can essentially be divided into two groups: directed graphical models and undirected graphical models.

2.1.2 Graphical Models

In this section we review some of the most commonly used graphical models.

Markov Random Field

A *Markov Random Field* (MRF) is defined as a pair $M = \langle G, \Phi \rangle$. Here $G = \langle V, E \rangle$ represents an undirected graph, where $V = \{V_i\}$ is the set of nodes, each of which corresponds to a random variable in $\mathcal{X}$; $E = \{(V_i, V_j) : i \neq j\}$ represents the set of undirected edges. The existence of an edge $\{u, v\}$ indicates the dependency of the random variable u and v. Φ is a set of potential functions (also called factors or clique potentials) associated with the maximal cliques in the graph G. Each potential function $\phi_c(\cdot)$ has the domain of some clique c in G, and is a mapping from possible joint assignments (to the elements of c) to non-negative real values. A maximal clique of a graph is a fully connected sub-graph that can not be further extended. We use C to represent the set of maximal cliques in the graph. ϕ_c is the potential function for a maximal clique $c \in \mathcal{C}$. The joint probability of a configuration x of the variables V can be calculated as the normalized product of the potential function over all the maximal cliques in G:

$$P(\mathbf{x}) = \frac{\prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c)}{\sum_{\mathbf{x}'_c} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c)}$$

where $\mathbf{x}_c$ represents the current configuration of variables in the maximal clique c, $\mathbf{x}'_c$ represents any possible configuration of variable in the maximal clique c. In practice, a Markov network is often conveniently expressed as a log-linear model, given by

$$P(\mathbf{x}) = \frac{\exp\left(\sum_{c \in C} w_c \phi_c(\mathbf{x}_c)\right)}{\sum_{\mathbf{x} \in \mathcal{X}} \exp\left(\sum_{c \in \mathcal{C}} w_c \phi_c(\mathbf{x}_c)\right)}$$

In the above equation, ϕ_c are feature functions from some subset of X to real values, w_c are weights which are to be determined from training samples. A log-linear model can provide more compact representations for any distributions, especially when the variables have large domains. This representation is also convenient in analysis because its negative log likelihood

is convex. However, evaluating the likelihood or gradient of the likelihood of a model requires inference in the model, which is generally computationally intractable due to the difficulty in calculating the partitioning function.

The Ising model is a special case of Markov Random Field. It comes from statistical physics, where each node represents the spin of a particle. In an Ising model, the graph is a grid, so each edge is a clique. Each node in the Ising model takes binary values $\{0, 1\}$. The parameters are θ_i representing the external field on particle i, and θ_{ij} representing the attraction between particles i and j. $\theta_{ij} = 0$ if i and j are not adjacent. The probability distribution is:

$$\begin{aligned} p(x|\theta) &= \exp\left(\sum_{i<j} \theta_{ij} x_i x_j + \sum_i \theta_i x_i = -A(\theta)\right) \\ &= \frac{1}{Z(\theta)} \exp\left(\sum_{i<j} \theta_{ij} x_i x_j + \sum_i \theta_i x_i\right) \end{aligned}$$

where $Z(\theta)$ is the partition function.

Gaussian Graphical Model

A *Gaussian Graphical Model* (GGM) models the Gaussian property of multivariate in an undirected graphical topology. Assuming that there are n variables and all variables are normalized so that each of them follows a standard Gaussian distribution. We use $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ to represent the $n \times 1$ column matrix. In a GGM, the variables $\mathbf{X}$ are assumed to follow a multivariate Gaussian distribution with covariance matrix Σ,

$$P(X) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \mathbf{X}^\top \Sigma^{-1} \mathbf{X}\right)$$

In a Gaussian Graphical Model, the existence of an edge between two nodes indicates that these two nodes are not conditionally independent given other nodes. Matrix $\Omega = \Sigma^{-1}$ is the *precision matrix*. The non-zero elements in the precision matrix correspond to the edges in the Gaussian Graphical Model.

Bayesian Networks

The most commonly used directed probabilistic graphical model is Bayesian Network [Pearl, 1988], which is a compact graphical representation of joint distributions. A Bayesian Network exploits the underlying conditional independencies in the domain, and compactly represent a joint distribution over variables by taking advantages of the local conditional independence structures. A Bayesian network $\mathcal{B} = \langle G, P \rangle$ is made of two components: a directed acyclic graph (DAG) G whose nodes correspond to the random variables, and a set of *conditional probabilistic distributions* (CPD), $P(x_i|\mathbf{x}_{\pi(i)})$, which describe the statistical relationship between each node i and its parents $\pi(i)$. In a CPD, for any specific configuration of $\mathbf{x}_{\pi(i)}$, the sum over all possible values of x_i is 1:

$$\sum_{x_i \in Val(x_i)} P(x_i|\mathbf{x}_{\pi(i)}) = 1.$$

In the continuous case,

$$\int_{x_i \in Val(x_i)} P(x_i|\mathbf{x}_{\pi(i)}) \mathrm{d}x_i = 1$$

where $P(x_i|\mathbf{x}_{\pi(i)})$ is the conditional density function. The conditional independence assumptions together with the CPDs uniquely determine a joint probability distribution via the *chain rule*:

$$P(x_1, \cdots, x_n) = \prod_{i=1}^{n} P(x_i|\mathbf{x}_{\pi(i)})$$

Other Graphical Models

Some other graphical models are briefly listed here.

- **Dependency Networks:** In [Heckerman et al., 2000], the authors proposed a probabilistic graphical model named dependency networks, which can be considered as a

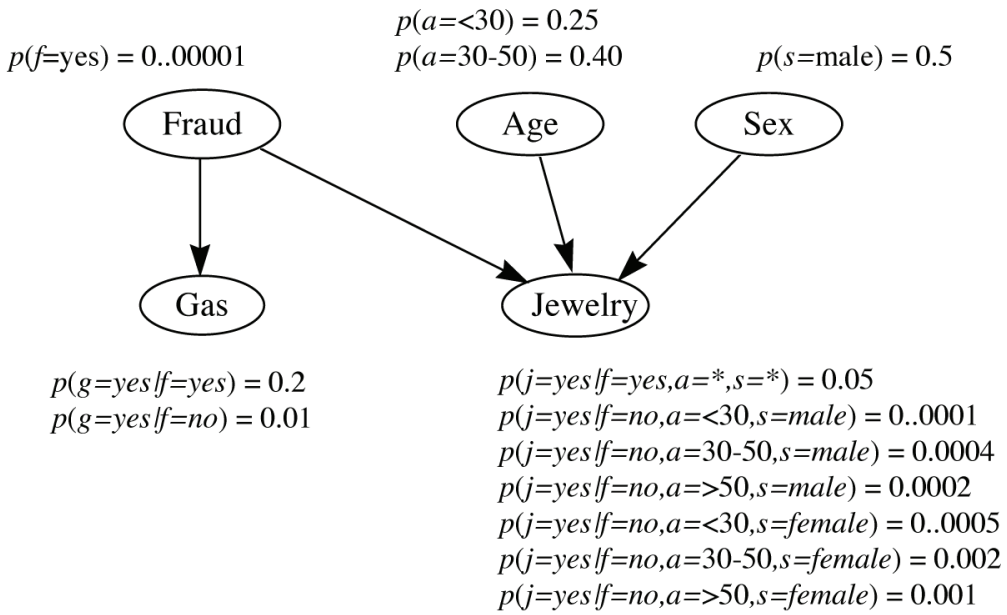


Figure 2.2: A Bayesian network for detecting credit-card fraud. Arcs indicate the causal relationship. The local conditional probability distributions associated with a node are shown next to the node. The asterisk indicates any value for that variable.

combination of Bayesian network and Markov network. The graph of a dependency network, unlike a Bayesian network, can be cyclic. The probability component of a dependency network, like a Bayesian network, is a set of conditional distributions, one for each node given its parents.

A dependency network is a pair $\langle G, P \rangle$ where G is a cyclic directed graph and P is a set of conditional probability distributions. The parents of nodes $\pi(i)$ of node i correspond to those variables that satisfy

$$p(x_i|\mathbf{x}_{\pi(i)}) = p(x_i|\mathbf{x}_{V\setminus i})$$

In other words, a dependency network is simply a collection of conditional distributions that are defined and built separately. In a specific context of sparse normal models, these would define a set of separate conditional linear regressions in which x_i is regressed to a small selected subset of other variables, each being determined separately.

The independencies in a dependency network are the same as those of a Markov network with the same adjacencies. The authors proved that the Gibbs sampler applied to the dependency network will yield a joint distribution for the domain. The applications of dependency network include probabilistic inference, collaborative filtering and the visualization of causal predictive relationships.

- **Module Networks:** In [Segal et al., 2003], the authors proposed a module networks model for gene regulatory network construction. The basic structure is a Bayesian network. Each regulatory module is a set of genes that are regulated in concert by a shared regulation program that governs their behavior. A regulation program specifies the behavior of the genes in the module as a function of the expression level of a small set of regulators. By employing the Bayesian structure learning to the modules instead of genes, this algorithm is able to reduce the computational complexity significantly.

  In [Toh and Horimoto, 2002] the authors proposed a model with the similar idea, yet they build a Gaussian Graphical Model instead of Bayesian networks, of module networks. In their study of the yeast (Saccharomyces cerevisiae) genes measured under 79 different conditions, the 2467 genes are first classified into 34 clusters by a hierarchical clustering analysis [Horimoto and Toh, 2001]. Then the expression levels of the genes in each cluster are averaged for each condition. The averaged expression profile data of 34 clusters were subjected to GGM, and a partial correlation coefficient matrix was obtained as a model of the genetic network.

- **Probabilistic Relational Models:** A probabilistic relational model [Friedman et al., 1999a] is a probabilistic description of the relational models, like the models in relational databases. A relational model consists of a set of classes and a set of relations. Each entity type is associated with a set of attributes. Each attribute takes on values in some fixed domain of values. Each relation is typed. The probabilistic relational model describes the relationships between entities and the properties of entities. The

model consists of two components: the qualitative dependency structure which is a DAG, and the parameters associated with it. The dependency structure is defined by associating with each attribute and its parents, which is modeled as conditional probabilities.

2.1.3 Network Topology

Two classes of network architectures are of special interest [Kitano, 2002]: the *small world networks* [Watts and Strogatz, 1998] and *scall-free power law networks* [Barabasi and Albert, 1999]. Small world networks are characterized by high clustering coefficients and small diameters. The clustering coefficient $C(p)$ is defined as follows. Suppose that a vertex v has k_v neighbors; then at most $k_v(k_v - 1)/2$ edges can exist between them (this occurs when every neighbor of v is connected to every other neighbor of v). Let C_v denote the fraction of these allowable edges that actually exist, then the clustering coefficient C is defined as the average of C_v over all v.

These properties reflect the existence of local building blocks together with long-range connectivity. Most nodes in small world networks have approximately the same number of links, and the degree distribution $P(k)$ decays exponentially for large k. Compared to small world networks, the scale-free power law networks have smaller clustering coefficients and large diameters. Most nodes in the scale-free networks are connected to a few neighbors, and only a small number of nodes, which is often called "hubs", are connected to a large number of nodes. This property is reflected by the power law for the degree distribution $P(k) \sim k^{-v}$.

Previous studies have found that a number of network structures appear to have structures between the small-world network and the scale-free network. In fact, these networks behave more like *hierarchical scale-free* [Han et al., 2004, Jeong et al., 2000, Lukashin et al., 2003, Basso et al., 2005, Bhan et al., 2002, Ravasz et al., 2002]. Nodes within the networks are first grouped into modules, whose connectivity is more like the small worlds network. The grouped

modules are then connected into a large network, which follows the degree distribution that is similar to that of the scale-free network.

2.1.4 Structure Learning of Graphical Models

There are three major approaches of existing structure learning methods: *constraint-based* approaches, *score-based* approaches and *regression-based* approaches.

Constraint-based approaches first attempt to identify a set of conditional independence properties, and then attempt to identify the network structure that best satisfies these constraints. The drawback with the constraints based approaches is that it is difficult to reliably identify the conditional independence properties and to optimize the network structure [Margaritis, 2003]. Plus, the constraints-based approaches lack an explicit objective function and they do not try to directly find the globally optimal structure. So they do not fit in the probabilistic framework.

Score-based approaches first define a score function indicating how well the network fits the data, then search through the space of all possible structures to find the one that has the optimal value for the score function. Problem with this approach is that it is intractable to evaluate the score for all structures, so usually heuristics, like greedy search, are used to find the sub-optimal structures.

Regression-based approaches are gaining popularity in recent years. Algorithms in this category are essentially optimization problems which guarantees global optimum for the objective function, and have better scalability.

2.2 Constraint-based Algorithms

The constraint-based approaches [Tsamardinos et al., 2006, Juliane and Korbinian, 2005, Spirtes et al., 2001, Wille et al., 2004, Margaritis, 2003, Margaritis and Thrun, 1999] employ the conditional independence tests to first identify a set of conditional independence

properties, and then attempts to identify the network structure that best satisfies these constraints. The two most popular constraint-based algorithm are the SGS algorithm and PC algorithm [Tsamardinos et al., 2006], both of which tries to d-separate all the variable pairs with all the possible conditional sets whose sizes are lower than a given threshold.

One problem with constraint-based approaches is that they are difficult to reliably identify the conditional independence properties and to optimize the network structure [Margaritis, 2003]. The constraint-based approaches lack an explicit objective function and thus they do not try to directly find the global structure with maximum likelihood. So they do not fit in the probabilistic framework.

The SGS Algorithm

The SGS algorithm (named after Spirtes, Glymour and Scheines) is the most straightforward constraint-based approach for Bayesian network structure learning. It determines the existence of an edge between every two node variables by conducting a number of independence tests between them conditioned on all the possible subsets of other node variables. The pseudo code of the SGS algorithm is listed in Algorithm 1. After slight modification, SGS algorithm can be used to learn the structure of undirected graphical models (Markov random fields).

The SGS algorithm requires that for each pair of variables adjacent in G, all possible subsets of the remaining variables should be conditioned. Thus this algorithm is super-exponential in the graph size (number of vertices) and thus unscalable. The SGS algorithm rapidly becomes infeasible with the increase of the vertices even for sparse graphs. Besides the computational issue, the SGS algorithm has problems of reliability when applied to sample data, because determination of higher order conditional independence relations from sample distribution is generally less reliable than is the determination of lower order independence relations.

Algorithm 1 SGS Algorithm

1: Build a complete undirected graph H on the vertex set V .
2: For each pair of vertices i and j, if there exists a subset S of $V \setminus \{i, j\}$ such that i and j are d-separated given S, remove the edge between i and j from G.
3: Let G' be the undirected graph resulting from step 2. For each triple of vertices i, j and k such that the pair i and j and the pair j and k are each adjacent in G' (written as $i - j - k$) but the pair i and k are not adjacent in G', orient $i - j - k$ as $i \to j \leftarrow k$ if and only if there is no subset S of $\{j\} \cup V \setminus \{i, j\}$ that d-separate i and k.
4: **repeat**
5: If $i \to j$, j and k are adjacent, i and k are not adjacent, and there is no arrowhead at j, then orient $j - k$ as $j \to k$.
6: If there is a directed path from i to j, and an edge between i and j, then orient $i - j$ as $i \to j$.
7: **until** no more edges can be oriented.

The PC Algorithm

The PC algorithm (named after Peter Spirtes and Clark Glymour) is a more efficient constraint-based algorithm. It conducts independence tests between all the variable pairs conditioned on the subsets of other node variables that are sorted by their sizes, from small to large. The subsets whose sizes are larger than a given threshold are not considered. The pseudo-code of the PC algorithm is given in Algorithm 2. We use $\mathcal{N}(i)$ to denote the adjacent vertices to vertex i in a directed acyclic graph G.

The complexity of the PC algorithm for a graph G is bounded by the largest degree in G. Suppose d is the maximal degree of any vertex and n is the number of vertices. In the worst case the number of conditional independence tests required by the PC algorithm is bounded by

$$2\binom{n}{2}\sum_{i=1}^{d}\binom{n-1}{i}$$

The PC algorithm can be applied on graphs with hundreds of nodes. However, it is not scalable if the number of nodes gets even larger.

Algorithm 2 PC Algorithm

1: Build a complete undirected graph G on the vertex set V .
2: n = 0.
3: **repeat**
4: **repeat**
5: Select an ordered pair of vertices i and j that are adjacent in G such that $\mathcal{N}(i) \setminus \{j\}$ has cardinality greater than or equal to n, and a subset S of $\mathcal{N}(i) \setminus \{j\}$ of cardinality n, and if i and j are d-separated given S delete edge $i - j$ from G and record S in $Sepset(i, j)$ and $Sepset(j, i)$.
6: **until** all ordered pairs of adjacent variables i and j such that $\mathcal{N}(i) \setminus \{j\}$has cardinality greater than or equal to n and all subsets S of $\mathcal{N}(i) \setminus \{j\}$of cardinality n have been tested for d-separation.
7: n = n + 1.
8: **until** for each ordered pair of adjacent vertices i and j, $\mathcal{N}(i) \setminus \{j\}$ is of cardinality less than n.
9: For each triple of vertices i, j and k such that the pair i, j and the pair j, k are each adjacent in G but the pair i, k are not adjacent in G, orient $i - j - k$ as $i \rightarrow j \leftarrow k$ if and only if j is not in $Sepset(i, k)$.
10: **repeat**
11: If $i \rightarrow j$, j and k are adjacent, i and k are not adjacent, and there is no arrowhead at j, then orient $j - k$ as $j \rightarrow k$.
12: If there is a directed path from i to j, and an edge between i and j, then orient $i - j$ as $i \rightarrow j$.
13: **until** no more edges can be oriented.

The GS Algorithm

Both the SGS and PC algorithm start from a complete graph. When the number of nodes in the graph becomes very large, even PC algorithm will be intractable due to the large combinatorial space.

In [Margaritis and Thrun, 1999], the authors proposed a Grow-Shrinkage (GS) algorithm to address the large scale network structure learning problem by exploring the sparseness of the graph. The GS algorithm uses two phases to estimate a superset of the Markov blanket $\hat{\partial}(j)$ for node j as in Algorithm 3. In the pseudo code, $i \leftrightarrow_S j$ denotes that node i and j are dependent conditioned on set S.

Algorithm 3 includes two phases to estimate the Markov blanket. In the "grow" phase,

Algorithm 3 GS: Estimating the Markov Blanket

1: $S \leftarrow \Phi$.
2: **while** $\exists j \in V \setminus \{i\}$ such that $j \leftrightarrow_S i$ **do**
3: $\quad S \leftarrow S \cup \{j\}$.
4: **end while**
5: **while** $\exists j \in S$ such that $j \nleftrightarrow_{S \setminus \{i\}} i$ **do**
6: $\quad S \leftarrow S \setminus \{j\}$.
7: **end while**
8: $\hat{\partial}(i) \leftarrow S$

Algorithm 4 GS Algorithm

1: *Compute Markov Blankets*: for each vertex $i \in V$ compute the Markov blanket $\partial(i)$.
2: *Compute Graph Structure*: for all $i \in V$ and $j \in \partial(i)$, determine j to be a direct neighbor of i if i and j are dependent given S for all $S \subseteq T$ where T is the smaller of $\partial(i) \setminus \{j\}$ and $\partial(j) \setminus \{i\}$.
3: *Orient Edges*: for all $i \in V$ and $j \in \partial(i)$, orient $j \rightarrow i$ if there exists a variable $k \in \partial(i) \setminus \{\partial(j) \cup \{j\}\}$ such that j and k are dependent given $S \cup \{i\}$ for all $S \subseteq U$ where U is the smaller of $\partial(j) \setminus \{k\}$ and $\partial(k) \setminus \{j\}$.
4: **repeat**
5: $\quad$ Compute the set of edges $C = \{i \rightarrow j$ such that $i \rightarrow j$ is part of a cycle$\}$.
6: $\quad$ Remove the edge in C that is part of the greatest number of cycles, and put it in R.
7: **until** there is no cycle exists in the graph.
8: *Reverse Edges*: Insert each edge from R in the graph, reversed.
9: *Propagate Directions*: for all $i \in V$ and $j \in \partial(i)$ such that neither $j \rightarrow i$ nor $i \rightarrow j$, execute the following rule until it no longer applies: if there exists a directed path from i to j, orient $i \rightarrow j$.

variables are added to the Markov blanket $\hat{\partial}(j)$ sequentially using a forward feature selection procedure, which often results in a superset of the real Markov blanket. In the "shrinkage" phase, variables are deleted from the $\hat{\partial}(j)$ if they are independent from the target variable conditioned on the subset of other variables in $\hat{\partial}(j)$. Given the estimated Markov blanket, the algorithm then tries to identify both the parents and children for each variable as in Algorithm 4.

In [Margaritis and Thrun, 1999], the authors further developed a randomized version of the GS algorithm to handle the situation when (i) the Markov blanket is relatively large, (ii) the number of training samples is small compared to the number of variables, or there are

noises in the data.

In a sparse network in which the Markov blankets are small, the complexity of GS algorithm is $O(n^2)$ where n is the number of nodes in the graph. Note that GS algorithm can be used to learn undirected graphical structures (Markov Random Fields) after some minor modifications.

2.3 Score-based Algorithms

Score-based approaches [Heckerman et al., 1995, Friedman et al., 1999b, Hartemink et al., 2001] first posit a criterion by which a given Bayesian network structure can be evaluated on a given dataset, then search through the space of all possible structures and tries to identify the graph with the highest score. Most of the score-based approaches enforce sparsity on the learned structure by penalizing the number of edges in the graph, which leads to a non-convex optimization problem. Score-based approaches are typically based on well established statistical principles such as Minimum Description Length (MDL) [Lam and Bacchus, 1994, Friedman and Goldszmidt, 1996, Allen and Greiner, 2000] or the Bayesian score. The Bayesian scoring approaches was first developed in [Cooper and Herskovits, 1992], and then refined by the BDe score [Heckerman et al., 1995], which is now one the of best known standards. These scores offer sound and well motivated model selection criteria for Bayesian network structure.

The main problem with score based approaches is that their associated optimization problems are intractable, i.e., it is NP-hard to compute the optimal Bayesian network structure using Bayesian scores [Chickering, 1996]. Recent researches have shown that for large samples, optimizing Bayesian network structure is NP-hard for all consistent scoring criteria including MDL, BIC and the Bayesian scores [Chickering et al., 2004]. Since the score-based approaches are not scalable for large graphs, they perform searches for the locally optimal solutions in the combinatorial space of structures, and the local optimal solutions they find

could be far away from the global optimal solutions, especially in the case when the number of sample configurations is small compared to the number of nodes.

The space of candidate structures in scoring based approaches is typically restricted to directed models (Bayesian networks) since the computation of typical score metrics involves computing the normalization constant of the graphical model distribution, which is intractable for general undirected models [Pollard, 1984]. Estimation of graph structures in undirected models has thus largely been restricted to simple graph classes such as trees [Chow et al., 1968], poly-trees [Chow et al., 1968] and hypertrees [Srebro, 2001].

2.3.1 Score Metrics

Score metrics are used to evaluate the goodness of a structure given a dataset. The most commonly used scores include the Minimum Description Length (MDL), the BDe score, and the Bayesian Information Criterion (BIC).

The MDL Score

The Minimum Description Length (MDL) principle [Rissanen, 1989] aims to minimize the space used to store a model and the data to be encoded in the model. In the case of learning a Bayesian network $\mathcal{B}$ which is composed of a graph G and the associated conditional probabilities $P_{\mathcal{B}}$, the MDL criterion requires choosing a network that minimizes the total description length of the network structure and the encoded data, which implies that the learning procedure balances the complexity of the induced network with the degree of accuracy with which the network represents the data.

Since the MDL score of a network is defined as the total description length, it needs to describe the data U, the graph structure G and the conditional probability P for a Bayesian network $\mathcal{B} = \langle G, P \rangle$.

To describe U, we need to store the number of variables n and the cardinality of each variable x_i. We can ignore the description length of U in the total description length since

U is the same for all candidate networks.

To describe the DAG G, we need to store the parents $\pi(i)$ of each variable x_i. This description includes the number of parents $|\pi(i)|$ and the index of the set $\pi(i)$ in some enumeration of all $\binom{n}{|\pi(i)|}$ sets of this cardinality. Since the number of parents $|\pi(i)|$ can be encoded in $\log n$ bits, and the indices of all parents of node i can be encoded in $\log \binom{n}{\pi(i)}$ bits, the description length of the graph structure G is

$$DL_{graph}(G) = \sum_i \left(\log n + \log \binom{n}{|\pi(i)|} \right) \tag{2.1}$$

To describe the conditional probability P in the form of CPD, we need to store the parameters in each conditional probability table. The number of parameters used for the table associated with x_i is $|\pi(i)|(|x_i| - 1)$. The description length of these parameters depends on the number of bits used for each numeric parameter. A usual choice is $1/2 \log N$ [Friedman and Goldszmidt, 1996]. So the description length for x_i's CPD is

$$DL_{tab}(x_i) = \frac{1}{2}|\pi(i)| \left(|x_i| - 1\right) \log N$$

To describe the encoding of the training data, we use the probability measure defined by the network $\mathcal{B}$ to construct a Huffman code for the instances in D. In this code, the length of each codeword depends on the probability of that instance. According to [Cover and Thomas, 1991], the optimal encoding length for instance x_i can be approximated as $-\log P_{x_i}$. So the description length of the data is

$$\begin{aligned} DL_{data}(D|\mathcal{B}) &= -\sum_{i=1}^{N} \log P(x_i) \\ &= -\sum_i \sum_{x_i, \mathbf{x}_{\pi(i)}} \#(x_i, \mathbf{x}_{\pi(i)}) \log P(x_i|\mathbf{x}_{\pi(i)}). \end{aligned}$$

In the above equation, $(x_i, \mathbf{x}_{\pi(i)})$ is a local configuration of variable x_i and its parents, $\#(x_i, \mathbf{x}_{\pi(i)})$ is the number of the occurrence of this configuration in the training data. Thus the encoding of the data can be decomposed as the sum of terms that are "local" to each CPD, and each term only depends on the counts $\#(x_i, \mathbf{x}_{\pi(i)})$.

If $P(x_i|\mathbf{x}_{\pi(i)})$ is represented as a table, then the parameter values that minimize $DL_{data}(D|\mathcal{B})$ are $\theta_{x_i|\mathbf{x}_{\pi(i)}} = \hat{P}(x_i|\mathbf{x}_{\pi(i)})$ [Friedman and Goldszmidt, 1998]. If we assign parameters accordingly, then $DL_{data}(D|\mathcal{B})$ can be rewritten in terms of conditional entropy as $N \sum_i H(x_i|\mathbf{x}_{\pi(i)})$, where

$$H(X|Y) = -\sum_{x,y} \hat{P}(x,y) \log \hat{P}(x|y)$$

is the conditional entropy of X given Y . The new formula provides an information-theoretic interpretation to the representation of the data: it measures how many bits are necessary to encode the values of x_i once we know $\mathbf{x}_{\pi(i)}$.

Finally, by combining the description lengths above, we get the total description length of a Bayesian network as

$$DL(G,D) = DL_{graph}(G) + \sum_i DL_{tab}(x_i) + N \sum_i H(x_i|\mathbf{x}_{\pi(i)}) \tag{2.2}$$

The BDe Score

The Bayesian score for learning Bayesian networks can be derived from methods of Bayesian statistics, one important example of which is BDe score [Cooper and Herskovits, 1992, Heckerman et al., 1995]. The BDe score is proportional to the posterior probability of the network structure given the data. Let G^h denote the hypothesis that the underlying distribution satisfies the independence relations encoded in G. Let Θ_G represent the parameters for the CPDs qualifying G. By Bayes rule, the posterior probability $P(G^h|D)$ is

$$P(G^h|D) = \frac{P(D|G^h)P(G^h)}{P(D)}$$

In the above equation, $1/P(D)$ is the same for all hypothesis, and thus we denote this constant as α. The term $P(D|G^h)$ is the probability given the network structure, and $P(G^h)$ is the prior probability of the network structure. They are computed as follows.

The prior over the network structures is addressed in several literatures. In [Heckerman et al., 1995], this prior is chosen as $P(G^h) \propto \alpha^{\Delta(G,G')}$, where $\Delta(G,G')$ is the difference in

edges between G and a prior network structure G', and $0 < a < 1$ is the penalty for each edge. In [Friedman and Goldszmidt, 1998], this prior is set as $P(G^h) \propto 2^{-DL_{graph}(G)}$, where $DL_{graph}(G)$ is the description length of the network structure G, defined in Equation 2.1.

The evaluation of $P(D|G^h)$ needs to consider all possible parameter assignments to G, namely

$$P(D|G^h) = \int P(D|\Theta_G, G^h) P(\Theta_G|G^h) \mathrm{d}\Theta_G, \tag{2.3}$$

where $P(D|\Theta_G, G^h)$ is the probability of the data given the network structure and parameters. $P(\Theta_G|G^h)$ is the prior probability of the parameters. Under the assumption that each distribution $P(x_i|\mathbf{x}_{\pi(i)})$ can be learned independently of all other distributions [Heckerman et al., 1995], Equation 2.3 can be written as

$$P(D|G^h) = \prod_i \prod_{\pi(i)} \int \prod_{x_i} \theta_{i,\pi(i)}^{N(x_i, \mathbf{x}_{\pi(i)})} P(\Theta_{i,\pi(i)}|G^h) \mathrm{d}\Theta_{i,\pi(i)}.$$

Note that this decomposition is analogous to the decomposition in Equation 2.2. In [Heckerman et al., 1995], the author suggested that each multinomial distribution $\Theta_{i,\pi(i)}$ takes a *Dirichlet prior*, such that

$$P(\Theta_{\mathbf{X}}) = \beta \prod_x \theta_x^{N'_x}$$

where $N'_x : x \in Val(X)$ is a set of *hyper parameters*, β is a normalization constant. Thus, the probability of observing a sequence of values of X with counts $N(x)$ is

$$\int \prod_x \theta_x^{N(x)} P(\Theta_X|G^h) \mathrm{d}\Theta_X = \frac{\Gamma(\sum_x N'(x))}{\Gamma(\sum_x (N'_x + N(x)))} \prod_x \frac{\Gamma(N'_x + N(x))}{\Gamma(N'_x)}$$

where $\Gamma(x)$ is the *Gamma* function defined as

$$\Gamma(x) = \int_0^\infty t^{x-t} e^{-t} \mathrm{d}t$$

The Gamma function has the following properties:

$$\begin{aligned} \Gamma(1) &= 1 \\ \Gamma(x+1) &= x\Gamma(x) \end{aligned}$$

If we assign each $\Theta_{i,\pi(i)}$ a Dirichlet prior with hyperparameters N then

$$P(D|G^h) = \prod_i \prod_{\pi(i)} \frac{\Gamma(\sum_i N'_{i,\pi(i)})}{\Gamma(\sum_i N'_{i,\pi(i)} + N(\pi(i)))} \prod_{x_i} \frac{\Gamma(N'_{i,\pi(i)+N(i,\pi(i))})}{\Gamma(N'_{i,\pi(i)})}$$

Bayesian Information Criterion (BIC)

A natural criterion that can be used for model selection is the logarithm of the relative posterior probability:

$$\log P(D, G) = \log P(G) + \log P(D|G) \tag{2.4}$$

Here the logarithm is used for mathematical convenience. An equivalent criterion that is often used is:

$$\log\left(\frac{P(G|D)}{P(G_0|D)}\right) = \log\left(\frac{P(G)}{P(G_0)}\right) + \log\left(\frac{P(D|G)}{P(D|G_0)}\right)$$

The ratio $P(D|G)/P(D|G_0)$ in the above equation is called *Bayes factor* [Kass and Raftery, 1995]. Equation 2.4 consists of two components: the log prior of the structure and the log posterior probability of the structure given the data. In the large-sample approximation we drop the first term.

Let us examine the second term. It can be expressed by marginalizing all the assignments of the parameters Θ of the network:

$$\log P(D|G) = \log \int_\Theta P(D|G, \Theta) P(\Theta|G) \mathrm{d}\Theta \tag{2.5}$$

In [Kass and Raftery, 1995], the authors proposed a *Gaussian approximation* for $P(\Theta|D, G) \propto P(D|\Theta, G)P(\Theta|G)$ for large amounts of data. Let

$$g(\Theta) \equiv \log(P(D|\Theta, G)P(\Theta|G))$$

We assume that $\tilde{\Theta}$ is the *maximum a posteriori* (MAP) configuration of Θ for $P(\Theta|D, G)$, which also maximizes $g(\Theta)$. Using the second degree Taylor series approximation of $g(\Theta)$ at $\tilde{\Theta}$:

$$g(\Theta) \approx g(\tilde{\Theta}) - \frac{1}{2}(\Theta - \tilde{\Theta})A(\Theta - \tilde{\Theta})^\top$$

Where A is the negative Hessian of $g(\Theta)$ at $\tilde{\Theta}$. Thus we get

$$\begin{aligned} P(\Theta|D,G) &\propto P(D|\Theta,G)P(\Theta,G) \\ &\approx P(D|\tilde{\Theta},S)P(\tilde{\Theta}|S)\exp\left(\frac{1}{2}(\Theta-\tilde{\Theta})A(\Theta-\tilde{\Theta})^{\top}\right) \end{aligned} \tag{2.6}$$

So we approximate $P(\Theta|D,G)$ as a multivariate Gaussian distribution. Plugging Equation 2.6 into Equation 2.5 and we get:

$$\log P(D|G) \approx \log P(D|\tilde{\Theta},G) + \log P(\tilde{\Theta}|G) + \frac{d}{2}\log(2\pi) - \frac{1}{2}\log|A| \tag{2.7}$$

where d is the dimension of $g(\Theta)$. In our case it is the number of free parameters.

Equation 2.7 is called a *Laplace approximation*, which is a very accurate approximation with relative error $O(1/N)$ where N is the number of samples in D [Kass and Raftery, 1995].

However, the computation of $|A|$ is a problem for large-dimension models. We can approximate it using only the diagonal elements of the Hessian A, in which case we assume independencies among the parameters.

In asymptotic analysis, we get a simpler approximation of the Laplace approximation in Equation 2.7 by retaining only the terms that increase with the number of samples N: $\log P(D|\tilde{\Theta},G)$ increases linearly with N; $\log|A|$ increases as $d\log N$. And $\tilde{\Theta}$ can be approximated by the maximum likelihood configuration $\tilde{\Theta}$. Thus we get

$$\log P(D|G) \approx P(D|\tilde{\Theta},S) - \frac{d}{2}\log N \tag{2.8}$$

This approximation is called the *Bayesian Information Criterion* (BIC) [Schwarz, 1978]. Note that the BIC does not depend on the prior, which means we can use the approximation without assessing a prior. The BIC approximation can be intuitively explained: in Equation 2.8 , $\log P(D|\tilde{\Theta},G)$ measures how well the parameterized structure predicts the data, and $(d/2\log N)$ penalizes the complexity of the structure. Compared to the Minimum Description Length score defined in Equation 2.2, the BIC score is equivalent to the MDL except term of the description length of the structure.

2.3.2 Search for the Optimal Structure

Once the score is defined, the next task is to search in the structure space and find the structure with the highest score. In general, this is an NP-hard problem [Chickering, 1996].

Note that one important property of the MDL score or the Bayesian score (when used with a certain class of *factorized* priors such as the BDe priors) is the *decomposability* in presence of complete data, i.e., the scoring functions we discussed earlier can be decomposed in the following way:

$$\text{Score}(G : D) = \sum_i \text{Score}(x_i | \mathbf{x}_{\pi(i)} : N_{x_i, \mathbf{x}_{\pi(i)}})$$

where $N_{x_i, \mathbf{x}_{\pi(i)}}$ is the number of occurrences of the configuration $(x_i, \mathbf{x}_{\pi(i)})$.

The decomposability of the scores is crucial for score-based learning of structures. When searching the possible structures, whenever we make a modification in a local structure, we can readily get the score of the new structure by re-evaluating the score at the modified local structure, while the scores of the rest part of the structure remain unchanged.

Due to the large space of candidate structures, simple search would inevitably leads to local maxima. To deal with this problem, many algorithms were proposed to constrain the candidate structure space. Here they are listed as follows.

Search over Structure Space

The simplest search algorithm over the structure is the greedy hill-climbing search [Heckerman et al., 1995]. During the hill-climbing search, a series of modifications of the local structures by adding, removing or reversing an edge are made, and the score of the new structure is reevaluated after each modification. The modifications that increase the score in each step is accepted. The pseudo-code of the hill-climbing search for Bayesian network structure learning is listed in Algorithm 5.

Besides the hill-climbing search, many other heuristic searching methods have also been used to learn the structures of Bayesian networks, including the simulated annealing [Chick-

Algorithm 5 Hill-climbing search for structure learning

1: Initialize a structure G'.
2: **repeat**
3: Set $G = G'$.
4: Generate the acyclic graph set $Neighbor(G)$ by adding, removing or reversing an edge in graph G.
5: Choose from $Neighbor(G)$ the one with the highest score and assign to G'.
6: **until** Convergence.

ering and Boutilier, 1996], best-first search [Russel and Norvig, 1995] and genetic search [Larranaga et al., 1996].

A problem with the generic search procedures is that they do not exploit the knowledge about the expected structure to be learned. As a result, they need to search through a large space of candidate structures. For example, in the hill-climbing structure search in Algorithm 5, the size of $Neighbor(G)$ is $O(n^2)$ where n is the number of nodes in the structure. So the algorithm needs to compute the scores of $O(n^2)$ candidate structures in each update (the algorithm also need to check acyclicity of each candidate structure), which renders the algorithm unscalable for large structures.

In [Friedman et al., 1999b], the authors proposed a Sparse Candidate Hill Climbing (SCHC) algorithm to solve this problem. The SCHC algorithm first estimates the possible candidate parent set for each variable and then use hill-climbing to search in the constrained space. The structure returned by the search can be used in turn to estimate the possible candidate parent set for each variable in the next step.

The key in SCHC is to estimate the possible parents for each node. Early works [Chow et al., 1968, Sahami, 1996] use *mutual information* (see Appendix E) to determine if there is an edge between two nodes:

$$I(X;Y) = \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)}$$

where $\hat{P}(\cdot)$ is the observed frequencies in the dataset. A higher mutual information indicates a stronger dependence between X and Y . Yet this measure is not suitable for determin-

ing the existence of an edge between two nodes and essentially has problems because, for example, it does not consider the information that we already learnt about the structure. Instead, Friedman et al. [1999b] proposed two other metrics to evaluate the dependency of two variables:

- The first metric is based on an alternative definition of mutual information. The mutual information between X and Y is defined as the distance between the joint distribution of $\hat{P}(X,Y)$ and the distribution $\hat{P}(X)\hat{P}(Y)$, which assumes the independency of the two variables:

$$I(X;Y) = D_{KL}\left(\hat{P}(X,Y)||\hat{P}(X)\hat{P}(Y)\right)$$

  where $D_{KL}(P||Q)$ is the *Kullback-Leibler divergence* (see Appendix D). Under this definition, the mutual information measures the error we introduce if we assume the independence of X and Y . During each step of the search process, we already have an estimation of the network $\mathcal{B}$. To utilize this information, similarly, we measure the discrepancy between the estimation $P_{\mathcal{B}}(X,Y)$ and the empirical estimation $\hat{P}(X,Y)$ as:

$$D_{KL}(P(X)||Q(X)) = \sum_{X} P(X)\log\frac{P(X)}{Q(X)}$$

  One issue with this measure is that it requires to compute $P_{\mathcal{B}}(X_i,Y_i)$ for pairs of variables. When learning networks over large number of variables this can be computationally expensive. However, one can easily approximate these probabilities by using a simple sampling approach.

- The second measure utilizes the Markov property that each node is independent of other nodes given its Markov blanket. First the *conditional mutual information* is defined as:

$$I(X;Y|Z) = \sum_{Z} \hat{P}(Z)D_{KL}(\hat{P}(X,Y|Z)||\hat{P}(X|Z)\hat{P}(Y|Z)).$$

This metric measures the error that is introduced when assuming the conditional independence of X and Y given Z. Based upon this, another metric is defined as:

$$M_{shield}(X_i, X_j|\mathcal{B}) = I(X_i; X_j|X_{\pi(i)})$$

Note that using either of these two metrics for searching, at the beginning of the search, i.e., $\mathcal{B}_0$ is an empty network, the measure is equivalent to $I(X;Y)$. Later iterations will incorporate the already estimated network structure in choosing the candidate parents.

Another problem with the hill-climbing algorithm is the stopping criteria for the search. There are usually two types of stopping criteria:

- *Score-based criterion*: the search process terminates when $Score(\mathcal{B}_t) = Score(\mathcal{B}_{t-1})$. In other words, the score of the network can no longer be increased by updating the network from candidate network space.

- *Candidate-based criterion*: the search process terminates when $C_i^t = C_i^{t-1}$ for all i, that is, the candidate space of the network remains unchanged.

Since the score is a monotonically increasing bounded function, the score-based criterion is guaranteed to stop. The candidate-based criterion might enter a loop with no ending, in which case certain heuristics are needed to stop the search.

There are four problems with the SCHC algorithm. First, the estimation of the candidate sets is not sound (i.e., may not identify the true set of parents), and it may take a number of iterations to converge to an acceptable approximation of the true set of parents. Second, the algorithm needs a pre-defined parameter k, the maximum number of parents allowed for any node in the network. If k is underestimated, there is a risk of discovering a suboptimal network. On the other hand, if k is overestimated, the algorithm will include unnecessary parents in the search space, thus jeopardizing the efficiency of the algorithm. Third, as already implied above, the parameter k imposes a uniform sparseness constraint on the network, thus may sacrifice either efficiency or quality of the algorithm. A more efficient way

to constrain the search space is the Max-Min Hill-Climbing (MMHC) algorithm [Tsamardinos et al., 2006], a hybrid algorithm which will be explained in Section 2.8. The last problem is that the constraint of the maximum number of parents k will conflict with the scale-free networks due to the existence of hubs (this problem exists for any algorithm that imposes this constraint).

Using the SCHC search, the number of candidate structures in each update is reduced from $O(n^2)$ to $O(n)$ where n is the number of nodes in the structure. Thus, the algorithm is capable to learn large-scale structures with hundreds of nodes.

The hill-climbing search is usually applied with multiple restarts and tabu list [Cvijovica-cute and Klinowski, 1995]. Multiple restarts are used to avoid local optima, and the tabu list is used to record the path of the search so as to avoid loops and local minima.

To solve the problem of large candidate structure space and local optima, some other algorithms are proposed which are briefly listed as follows.

- In [Moore and Wong, 2003], the authors proposed a search strategy based on a more complex search operator called optimal reinsertion. In each optimal reinsertion, a target node in the graph is picked and all arcs entering or exiting the target are deleted. Then a globally optimal combination of in-arcs and out-arcs are found and reinserted into the graph subject to some constraints. With the optimal reinsertion operation defined, the search algorithm generates a random ordering of the nodes and applies the operation to each node in the ordering in turn. This procedure is iterated, each with a newly randomized ordering, until no change is made in a full pass. Finally, a conventional hill-climbing is performed to relax the constraint of max number of parents in the optimal reinsertion operator.

- In [Xiang et al., 1997], the authors state that with a class of domain models of probabilistic dependency network, the optimal structure can not be learned through the search procedures that modify a network structure one link at a time. For example, given the XOR nodes there is no benefit in adding any *one* parent individually without

the others and so single-link hill-climbing can make no meaningful progress. They propose a multi-link lookahead search for finding decomposable Markov Networks (DMN). This algorithm iterates over a number of levels where at level i, the current network is continually modified by the best set of i links until the entropy decrement fails to be significant.

- Some algorithms identify the Markov blanket or parent sets by either using conditional independency test, mutual information or regression, then use hill-climbing search over this constrained candidate structure space [Tsamardinos et al., 2006, Schmidt and Murphy, 2007]. These algorithms belong to the hybrid methods. Some of them are listed in Section 2.8.

Search over Ordering Space

The acyclicity of the Bayesian network implies an *ordering* property of the structure such that if we order the variables as $\langle x_1, \cdots, x_n \rangle$, each node x_i would have parents only from the set $\{x_1, \cdots, x_{i-1}\}$. Fundamental observations [Buntine, 1991, Cooper and Herskovits, 1992] have shown that given an ordering on the variables in the network, finding the highest-scoring network consistent with the ordering is not NP-hard. Indeed, if the in-degree of each node is bounded to k and all structures are assumed to have equal probability, then this task can be accomplished in time $O(n^k)$ where n is the number of nodes in the structure.

Search over the ordering space has some useful properties. First, the ordering space is significantly smaller than the structure space: $2^{O(n \log n)}$ orderings versus $2^{\Omega(n^2)}$ structures where n is the number of nodes in the structure [Robinson, 1973]. Second, each update in the ordering search makes a more global modification to the current hypothesis and thus has more chance to avoid local minima. Third, since the acyclicity is already implied in the ordering, there is no need to perform acyclicity checks, which is potentially a costly operation for large networks.

The main disadvantage of ordering-based search is the need to compute a large set of

sufficient statistics ahead of time for each variable and each possible parent set. In the discrete case, these statistics are simply the frequency counts of instantiations: $\#(x_i, \mathbf{x}_{\pi(i)})$ for each $x_i \in Val(x_i)$ and $\mathbf{x}_{\pi(i)} \in Val(\mathbf{x}_{\pi(i)})$. This cost would be very high if the number of samples in the dataset is large. However, the cost can be reduced by using AD-tree data structure [Moore and Lee, 1998], or by pruning out possible parents for each node using SCHC [Friedman et al., 1999b], or by sampling a subset of the dataset randomly.

Here some algorithms that search through the ordering space are listed:

- The ordering-based search was first proposed in [Larranaga et al., 1996] which uses a genetic algorithm search over the structures, and thus is very complex and not applicable in practice.

- In [Friedman and Koller, 2003], the authors proposed to estimate the probability of a structural feature (i.e., an edge) over the set of all orderings by using a Markov Chain Monte Carlo (MCMC) algorithm to sample over the possible orderings. The authors asserts that in the empirical study, different runs of MCMC over network structure typically lead to very different estimates in the posterior probabilities over network structure features, illustrating poor convergence to the stationary distribution. By contrast, different runs of MCMC over orderings converge reliably to the same estimates.

- In [Teyssier and Koller, 2005], the authors proposed a simple greedy local hill-climbing with random restarts and a tabu list. First the score of an ordering is defined as the score of the best network consistent with it. The algorithm starts with a random ordering of the variables. In each iteration, a swap operation is performed on any two adjacent variables in the ordering. Thus the branching factor for this swap operation is $O(n)$. The search stops at a local maximum when the ordering with the highest score is found. The tabu list is used to prevent the algorithm from reversing a swap that was executed recently in the search. Given an ordering, the algorithm then tries to find

the best set of parents for each node using the Sparse Candidate algorithm followed by exhaustive search.

- In [Koivisto, 2004, Singh and Moore, 2005], the authors proposed to use dynamic programming to search for the optimal structure. The key in the dynamic programming approach is the ordering $\prec$, and the marginal posterior probability of the feature f:

$$p(f|\prec) = \sum_{\prec} p(\prec|x)p(f|x,\prec)$$

Unlike [Friedman and Koller, 2003] which uses MCMC to approximate the above value, the dynamic programming approach does exact summation using the permutation tree. Although this approach may find the exactly best structure, the complexity is $O(n2^n + n^{k+1}C(m))$ where n is the number of variables, k is a constant in-degree, and $C(m)$ is the cost of computing a single local marginal conditional likelihood for m data instances. The authors acknowledge that the algorithm is feasible only for $n \leq 26$.

2.4 Regression-based Structure Learning

A large number of structure learning algorithms use regression based approaches. Approaches in this category model the interactions among genes by a collection of linear equations, and tried to fit the equations using the training data.

2.4.1 Regression Model

Given N data samples as (x_i, y_i) and pre-defined basis functions $\phi(\cdot)$, the task of regression is to find a set of weights $\mathbf{w}$ such that the basis functions give the best prediction of the label y_i from the input x_i. The performance of the prediction is given by an loss function $E_D(\mathbf{w})$. For example, in a linear regression,

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \mathbf{w}^\top\phi(\mathbf{x}_i)\right)^2 \tag{2.9}$$

To avoid over-fitting, a regularizer is usually added to penalize the weights $\mathbf{w}$. So the regularized loss function is:

$$E(\mathbf{w}) = E_D(\mathbf{w}) + \lambda E_W(\mathbf{w}) \tag{2.10}$$

The regularizer penalizes each element of $\mathbf{w}$:

$$E_W(\mathbf{w}) = \sum_{j=1}^{M} \alpha_i \|w_j\|_q$$

When all α_i's are the same,

$$E_W(\mathbf{w}) = \|\mathbf{w}_j\|_q$$

where $\|\cdot\|_q$ is the L_q norm, λ is the regularization coefficient that controls the relative importance of the data-dependent error and the regularization term. With different values of q, the regularization term may give different results:

1. When $q = 2$, the regularizer is in the form of sum-of-squares

$$E_W(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{w}$$

This particular choice of regularizer is known in machine learning literature as *weight decay* [Bishop, 2006] because in sequential learning algorithm, it encourages weight values to decay towards zeros, unless supported by the data. In statistics, it provides an example of a parameter shrinkage method because it shrinks parameter values towards zero.

One advantage of the L_2 regularizer is that it is rotationally invariant in the feature space. To be specific, given a deterministic learning algorithm L, it is rotationally invariant if, for any training set S, rotational matrix M and test example x, there is $L[S](x) = L[MS](Mx)$. More generally, if L is a stochastic learning algorithm so that its predictions are random, it is rotationally invariant if, for any S, M and x, the

prediction $L[S](x)$ and $L[MS](Mx)$ have the same distribution. A complete proof in the case of logistic regression is given in [Ng, 2004].

This quadratic (L_2) regularizer is convex, so if the loss function being optimized is also a convex function of the weights, then the regularized loss has a single global optimum. Moreover, if the loss function $E_D(\mathbf{w})$ is in quadratic form, then the minimizer of the total error function has a closed form solution. Specifically, if the data-dependent error $E_D(\mathbf{w})$ is the sum-of-squares error as in Equation 2.10, then setting the gradient with respect to w to zero, then the solution is

$$\mathbf{w} = (\lambda \mathbf{I} + \Phi^\top \Phi)^{-1} \Phi^\top \mathbf{t}$$

This regularizer is seen in *ridge regression* [Hoerl and Kennard, 2000], the support vector machine [Hoerl and Kennard, 2000, Schölkopf and Smola, 2002] and regularization networks [Girosi et al., 1995].

2. $q = 1$ is called *lasso* regression in statistics [Tibshirani, 1996]. It has the property that if λ is sufficiently large, then some of the coefficients w_i are driven to zero, which leads to a sparse model in which the corresponding basis functions play no role. To see this, note that the minimization of Equation 2.10 is equivalent to minimizing the unregularized sum-of-squares error subject to the constraint over the parameters:

$$\arg\min_{\mathbf{w}} \quad \frac{1}{2} \sum_{i=1}^{N} \Big(y_i - \mathbf{w}^\top \phi(\mathbf{x}_i) \Big)^2 \tag{2.11}$$

$$\text{s. t.} \quad \sum_{j=1}^{M} \|w_j\|_q \leq \eta \tag{2.12}$$

The Lagrangian of Equation 2.11 gives Equation 2.10. The sparsity of the solution can be seen from Figure 2.3. Theoretical study has also shown that lasso L_1 regularization may effectively avoid over-fitting. In [Dudk et al., 2004], it is shown that the density

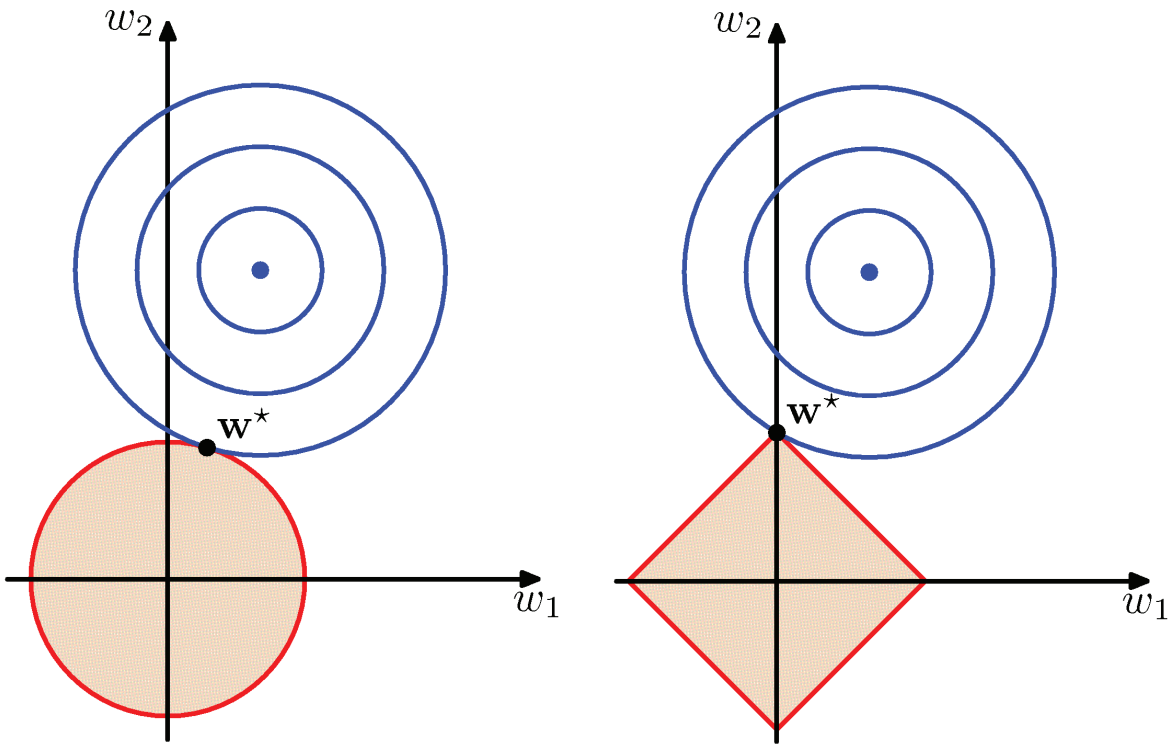


Figure 2.3: Contours of the unregularized objective function (blue) along with the constraint region (yellow) with L_2 regularizer (left) and L_1 regularizer. The lasso regression gives a sparse solution. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

estimation in log-linear models using L_1 regularized likelihood has sample complexity that grows only logarithmically in the number of features of the log-linear model; Ng [2004] and Wainwright et al. [2006] show a similar result for L_1 regularized logistic regression respectively.

The asymptotic properties of Lasso-type estimates in regression have been studied in detail in [Knight and Fu, 2000] for a fixed number of variables. Their results say that the regularization parameter λ should decay for an increasing number of observations at least as fast as $N^{-1/2}$ to obtain $N^{1/2}$-consistent estimate where N is the number of observations.

3. If $0^0 \equiv 0$ is defined, then the L_0 regularization contributes a fixed penalty α_i for each weight $w_i \neq 0$. If all α_i are identical, then this is equivalent to setting a limit on the

maximum number of non-zero weights. However, the L_0 norm is not a convex function, and this tends to make exact optimization of the objective function expensive.

In general, the L_q norm has parsimonious property (with some components being exactly zero) for $q \leq 1$, while the optimization problem is only convex for $q \geq 1$. So L_1 regularizer occupies a unique position, as $q = 1$ is the only value of q such that the optimization problem leads to a sparse solution, while the optimization problem is still convex.

2.4.2 Structure Learning through Regression

Learning a graphical structure by regression is gaining popularity in recent years. The algorithms proposed mainly differ in the choice of the objective loss functions. They are listed in the following according to the different objectives they use.

Likelihood Objective

Methods in this category use the negative likelihood or log-likelihood of the data given the parameters of the model as the objective loss function $E_D(\cdot)$.

- In [In Lee et al., 2006], the authors proposed an L_1 regularized structure learning algorithm for Markov Random Field, specifically, in the framework of log-linear models. Given a variable set $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, a log-linear model is defined in terms of a set of feature functions $f_k(\mathbf{x}_k)$, each of which is a function that defines a numerical value for each assignment $\mathbf{x}_k$ to some subset $\mathbf{x}_k \subset \mathcal{X}$. Given a set of feature functions $F = \{f_k\}$, the parameters of the log-linear model are weights $\theta = \{\theta_k : f_k \in F\}$. The overall distribution is defined as:

$$P_\theta(\mathbf{x}) = \frac{1}{Z(\theta)} \exp\left(\sum_{f_k \in F} \theta_k f_k(\mathbf{x})\right)$$

where $Z(\theta)$ is a normalizer called partition function. Given an iid training dataset $\mathcal{D}$, the log-likelihood function is:

$$\mathcal{L}(\mathcal{M}, \mathcal{D}) = \sum_{f_k \in F} \theta_k f_k(\mathcal{D} - M \log Z(\theta) = \theta^\top \mathbf{f}(\mathcal{D}) - M \log Z(\theta)$$

where $f_k(\mathcal{D}) = \sum_{m=1}^{M} f_k(x_k[m])$ is the sum of the feature values over the entire data set, $\mathbf{f}(\mathcal{D})$ is the vector where all of these aggregate features have been arranged in the same order as the parameter vector, and $\theta^\top \mathbf{f}(\mathcal{D})$ is a vector dot-product operation. To get a sparse MAP approximation of the parameters, a Laplacian parameter prior for each feature f_k is introduced such that

$$P(\theta_k) = \frac{\beta_k}{2} \exp(-\beta_k |\theta_k|)$$

And finally the objective function is:

$$\max_{\theta} \quad \theta^\top \mathbf{f}(\mathcal{D}) - M \log Z(\theta) - \sum_k \beta_k |\theta_k|$$

Before solving this optimization problem to get the parameters, features should be included into the model. Instead of including all features in advance, the authors use *grafting* procedure [Perkins et al., 2003] and *gain-based* method [Pietra et al., 1997] to introduce features into the model incrementally.

- In [Wainwright et al., 2006], the authors restricted to the Ising model, a special family of MRF, defined as

$$p(x, \theta) = \exp\left(\sum_{s \in V} \theta_s x_s + \sum_{(s,t) \in E} \theta_{st} x_s x_t - \Psi(\theta) \right)$$

The logistic regression with L_1-regularization that minimizing the negative log likelihood is achieved by optimizing:

$$\hat{\theta}^{s,\lambda} = \underset{\theta \in \mathbb{R}^p}{\arg\min} \left(\frac{1}{n} \sum_{i=1}^{n} \Big(\log(1 + \exp(\theta^\top z^{(i,s)})) - x_s^{(i)} \theta^\top z^{(i,s)} \Big) + \lambda_n \|\theta_{\setminus s}\|_1 \right)$$

Dependency Objective

Algorithms in this category use linear regression to estimate the Markov blanket of each node in a graph. Each node is considered dependent on nodes with nonzero weights in the regression.

- In [Meinshausen and Bühlmann, 2006], the authors used linear regression with L_1 regularization to estimate the neighbors of each node in a Gaussian graphical model:

$$\hat{\theta}_{i,\lambda} = \underset{\theta:\theta_i=0}{\arg\min} \quad \frac{1}{n}\|x_i - \theta^\top \mathbf{x}\|_2^2 + \lambda\|\theta\|_1$$

  The authors discussed in detail the choice of regularizer weight λ, for which the cross-validation choice is not the best under certain circumstances. For the solution, the authors proposed an optimal choice of λ under certain assumptions with full proof.

- In [Fan, 2006], the authors proposed to learn GGMs from directed graphical models using modified Lasso regression, which seems a promising method. The algorithm is listed here in detail.

  Given a GGM with variables $\mathbf{x} = [x_1, \cdots, x_p]^\top$ and the multivariate Gaussian distribution with covariance matrix Σ:

$$P(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1}\mathbf{x}\right)$$

  This joint probability can always be decomposed into the product of multiple conditional probabilities:

$$P(\mathbf{x}) = \prod_{i=1}^{p} P(x_i | x_{i+1,\cdots,p})$$

  Since the joint probability in the GGM is a multivariate Gaussian distribution, each conditional probability also follows Gaussian distribution. This implies that for any GGM there is at least one DAG with the same joint distribution.

Suppose that for a DAG there is a specific ordering of variables as $1, 2, \cdots, p$. Each variable x_i only has parents with indices larger than i. Let β denote the regression coefficients and D denote the data. The posterior probability given the DAG parameter β is

$$P(D|\beta) = \prod_{i=1}^{p} P(x_i|\mathbf{x}_{(i+1):p}, \beta)$$

Suppose linear regression $x_i = \sum_{j=i+1}^{p} \beta_{ji} x_j + \epsilon_i$ where the error ϵ_i follows normal distribution $\epsilon_i \sim N(0, \psi_i)$, then

$$\begin{aligned} \mathbf{x} &= \Gamma\mathbf{x} + \epsilon \\ \epsilon &\sim N_p(0, \Psi) \end{aligned}$$

where Γ is an upper triangular matrix, $\Gamma_{ij} = \beta_{ji}, i < j$, $\epsilon = (\epsilon_1, \cdots, \epsilon_p)^\top$ and $\Psi = diag(\psi_1, \cdots, \psi_p)$. Thus

$$\mathbf{x} = (I - \Gamma)^{-1}\epsilon$$

So $\mathbf{x}$ follows a joint multivariate Gaussian distribution with covariance matrix and precision matrix as:

$$\begin{aligned} \Sigma &= (I - \Gamma)^{-1}\Psi((I - \Gamma)^{-1})^\top \\ \Omega &= (I - \Gamma)^\top \Psi^{-1}(I - \Gamma) \end{aligned}$$

Wishart prior is assigned to the precision matrix Ω such that $\Omega \sim W_p(\delta, T)$ with δ degrees of freedom and diagonal scale matrix $T = diag(\theta_1, \cdots, \theta_p)$. Each θ_i is a positive hyper prior and satisfies

$$P(\theta_i) = \frac{\lambda}{2} \exp(\frac{-\lambda\theta_i}{2})$$

Let $\beta_i = (\beta_{(i+1)i}, \cdots, \beta_{pi})^\top$, and T_i represents the sub-matrix of T corresponding to variables $\mathbf{x}_{(i+1):p}$. Then the associated prior for β_i is $P(\beta_i|\psi_i, \theta_{(i+1):p}) =$

$N_{p-1}(0, T_i\psi_i)$ [Geiger and Heckerman, 2002], thus:

$$P(\beta_{ji}|\psi_i, \theta_j) = N(0, \theta_j\psi_i)$$

And the associated prior for ψ_i is

$$P(\psi_i^{-1}|\theta_i) = \Gamma\left(\frac{\delta + p - 1}{2}, \frac{\theta_i^{-1}}{2}\right)$$

where $\Gamma(\cdot)$ is the Gamma distribution. Like in [Figueiredo and Jain, 2001], the hyper prior θ can be integrated out from prior distribution of β_{ji} and thus

$$\begin{aligned} P(\beta_{ji}|\psi_i) &= \int_0^\infty P(\beta_{ji}|\psi_i, \theta_j)P(\theta_j) \\ &= \frac{1}{2}\left(\frac{\lambda}{\psi_i}\right)\exp\left(-\left(\frac{\lambda}{\psi_i}\right)^{\frac{1}{2}}|\beta_{ji}|\right) \end{aligned}$$

Suppose there are K samples in the data D and x_{ki} is the i-th variable in the k-th sample, then

$$\begin{aligned} P(\beta_i|\psi_i, D) &\propto P(x_i x_{(i+1):p}, \beta_i, \psi_i)P(\beta_i|\psi_i) \\ &\propto \exp\left(\frac{\sum_k (x_{ki} - \sum_{j=i+1}^p \beta_{ji}x_{kj})^2 + \sqrt{\lambda\psi_i}\sum_{j=i+1}^p |\beta_{ji}|}{-\psi_i}\right) \end{aligned}$$

and

$$P(\psi_i^{-1}|\theta_i, \beta_i, D) = \Gamma\left(\frac{\delta + p - i + K}{2}, \frac{\theta_i^{-1} + \sum_k (x_{ki} - \sum_{j=i+1}^p \beta_{ji}x_{kj})^2}{2}\right)$$

The MAP estimation of β_i is:

$$\hat{\beta}_i = \arg\min \sum_k \left(x_{ki} - \sum_{j=i+1}^p \beta_{ji}x_{kj}\right)^2 + \sqrt{\lambda\psi_i}\sum_{j=i+1}^p |\beta_{ji}|$$

$\hat{\beta}_i$ is the solution of a Lasso regression.

The authors further proposed a *Feature Vector Machine* (FVM) which is an advance to the the generalized Lasso regression (GLR) [Roth, 2004] which incorporates kernels,

to learn the structure of undirected graphical models. The optimization problem is:

$$\begin{aligned} \arg\min_{\beta} \quad & \frac{1}{2}\sum_{p,q}\beta_p\beta_q K(f_p, f_q) \\ \text{s.t.} \quad & \left|\sum_{p}\beta_p K(f_q, f_p) - K(f_q, y)\right| \leq \frac{\lambda}{2}, \forall q \end{aligned}$$

where $K(f_i, f_j) = \phi(f_i)^\top \phi(f_j)$ is the kernel function, $\phi(\cdot)$ is the mapping, either linear or non-linear, from original space to a higher dimensional space; f_k is the k-th feature vector, and y is the response vector from the training dataset.

System-identification Objective

Algorithms in this category [Arkin et al., 1998, Gardner et al., 2003, Glass and Kauffman, 1973, Gustafsson et al., 2003, McAdams and Arkin, 1998] get ideas from network identification by multiple regression (NIR) It is derived from a branch of engineering called system identification [Ljung, 1999], in which a model of the connections and functional relations between elements in a network is inferred from measurements of system dynamics. The entire system is modeled using a differential equation, then regression algorithms are used to fit the parameters in this equation. We illustrate the key idea of this type of approaches by using the algorithm in [Gustafsson et al., 2003] as an example.

Near a steady-state point (e.g., when gene expression does not change substantially over time), the nonlinear system of the genes may be approximated to the first order by a linear differential equation as:

$$\frac{\mathrm{d}x_i^t}{\mathrm{d}t} = \sum_{j=1}^{n} w_{ij} x_j^t + \epsilon_i$$

where x_i^t is the expression of gene i at time t. The network of the interaction can be inferred

by minimizing the residual sum of squares with constraints on the coefficients:

$$\begin{aligned} \arg\min_{w_{ij}} \quad & \sum_t \left(\sum_{j=1}^{n} w_{ij} x_j^t - \frac{\mathrm{d}x_i^t}{\mathrm{d}t} \right)^2 \\ \text{s.t.} \quad & \sum_{j=1}^{n} |w_{ij}| \leq \mu_i \end{aligned}$$

Note that this is essentially a Lasso regression problem since the constraints added to the Lagrangian is equivalent to L_1 regularizers. Finally the adjancency matrix A of the network is identified from the coefficients by

$$A_{ij} = \begin{cases} 0 & \text{if } w_{ji} = 0 \\ 1 & \text{otherwise} \end{cases}$$

One problem with this approach is when the number of samples is less than the number of variables, the linear equation is undertermined. To solve this problem, D'haeseleer et al. [1999] use non-linear interpolation to generate more data points to make the equation determined; Yeung et al. [2002] use singular value decomposition (SVD) to first decompose the training data, and then constrain the interaction matrix by exploring the sparseness of the interactions.

Precision Matrix Objective

In [Banerjee et al., 2006], the authors proposed a convex optimization algorithm for fitting sparse Gaussian graphical model from precision matrix. Given a large-scale empirical dense covariance matrix S of multivariate Gaussian data, the objective is to find a sparse approximation of the precision matrix. Assuming X is the estimate of the precision matrix (the inverse of the variance matrix). The optimization of the penalized maximum likelihood (ML) is:

$$\max_{X \succ 0} \quad \log\det(X) - \mathrm{Tr}(SX) - \rho \|X\|_1$$

The problem can be efficiently solved by Nesterovs method [Nesterov, 2005].

MDL Objective

Methods in this category encode the parameters into the Minimum Description Length (MDL) criterion, and tries to minimize the MDL with respect to the regularization or constraints.

- In [Schmidt and Murphy, 2007], the authors proposed a structure learning approach which uses the L_1 penalized regression with MDL as loss function to find the parents/neighbors for each node, and then apply the score-based search to find the appropriate structure for the given data set. The first step is the L_1 variable selection to find the parents/neighbors set of a node by solving:

$$\hat{\theta}_j^{L_1}(U) = \arg\min_{\theta} NLL(j, U, \theta) + \lambda \|\theta\|_1 \tag{2.13}$$

  where λ is the regularization parameter for the L_1 norm of the parameter vector. $NLL(j, U, \theta)$ is the negative log-likelihood of node j with parents $\pi(j)$ and parameters θ:

$$MDL(G) = \sum_{j=1}^{d} NLL(j, \pi_j, \hat{\theta}_j^{mle}) + \frac{|\hat{\theta}_j^{mle}|}{2} \log n \tag{2.14}$$

$$NLL(j, \pi(j), \theta) = -\sum_{i=1}^{N} \log P(X_{ij} | X_{i,\pi(j)}, \theta) \tag{2.15}$$

  where N is the number of samples in the dataset.

  The L_1 regularizer will generate a sparse solution with many parameters being zero. The set of variables with non-zero values are set as the parents of each node. This hybrid structure learning algorithm is further discussed in Section 2.8.

- In [Guo and Schuurmans, 2006], the authors proposed an interesting structure learning algorithm for Bayesian Networks, which incorporates parameter estimation, feature selection and variable ordering into one single convex optimization problem, leading to a constrained regression problem. The parameters of the Bayesian network and the

variables selected are encoded in the MDL objective function which is to be minimized. The topological properties of the Bayesian network (antisymmetricity, transitivity and reflexivity) are encoded as the constraints of the optimization problem.

2.5 Clustering Based Structure Learning

The simplest structure learning method is through clustering. First the similarities of any two variables are estimated, then any two sets of variables are linked if they meet certain standards [Lukashin et al., 2003]. Here the similarity may take different measures, including

- Correlation [Eisen et al., 1998, Spellman et al., 1998, Iyer et al., 1999, Alizadeh et al., 2000]
- Euclidean distance [Wen et al., 1998, Tamayo et al., 1999, Tavazoie et al., 1999]
- Mutual information (see Appendix E) [Butte and Kohane, 2000]
- Pearson correlation coefficient (see Appendix B) [Eisen et al., 1998]
- Self organizing map (SOM) [Törönen et al., 1999]
- Inner product [Alizadeh et al., 2000]

Using hierarchical clustering [Manning et al., 2008], the hierarchy structure can be presented at different scales. Figure 2.4 shows an example of hierarchical clustering of genes.

2.6 Matrix Factorization Based Structure Learning

Methods in this category use matrix factorization techniques to identify the interactions between variables. The matrix factorization algorithms used include singular value decomposition [Alter et al., 2000, D'haeseleer et al., 1999, Raychaudhuri et al., 2000], max-margin

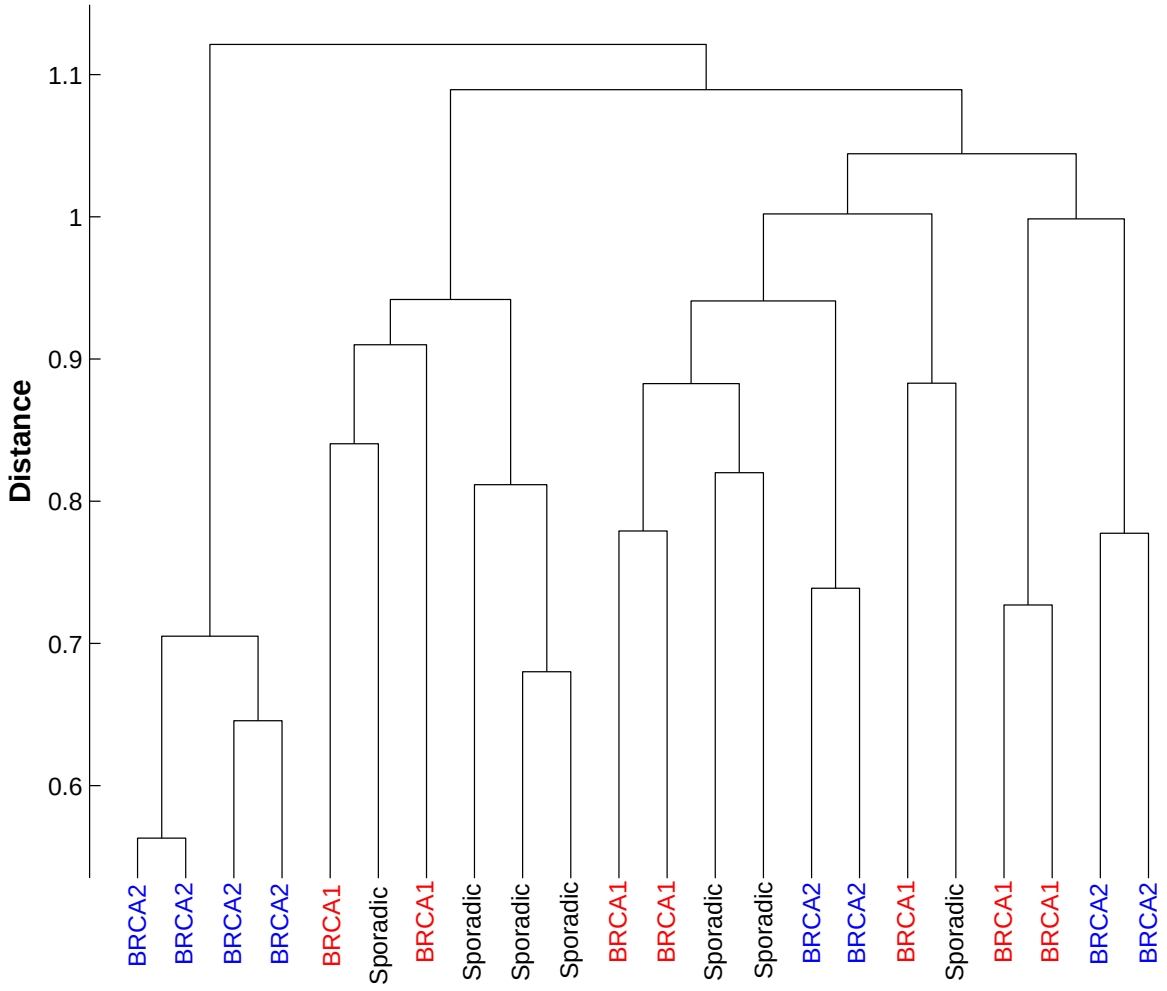


Figure 2.4: A structure of genes from hierarchical clustering. Figure excerpted from [Varma and Simon, 2004].

matrix factorization [DeCoste, 2006, Rennie and Srebro, 2005, Srebro et al., 2005] and non-negative matrix factorization [Badea and Tilivea, 2005, Paatero and Tapper, 1994, Hoyer and Dayan, 2004, Lee and Seung, 2001, Shahnaz et al., 2006, Weinberger et al., 2005], network component analysis (NCA) [Liao et al., 2003]. In Chapter 4 we will review the different matrix factorization algorithms in detail, where we will also develop a knowledge-driven matrix factorization (KMF) which is able to combine side information in the matrix factorization. It not only derives the membership assignment to the clusters, but also computes the strength of interactions among the clusters.

2.7 Regularization Based Structure Learning

Although the structure information often appears explicitly in many situations like the conditional dependency in a Bayesian network which can be manifested using a directed graph, it may be implicit sometimes and thus difficult to explore. For example, a group of variables show co-varying property in group lasso regularization, i.e., they either co-exist or diminish together. Several algorithms including group lasso [Yuan and Lin, 2005] and Composite Absolute Penalties (CAP) [Zhao et al., 2009] have been proposed to exploit the information embedded in the group structure.

These grouped and hierarchical regularizations are mostly restricted to discovering positive correlations among the variables within groups only. Specifically, they assume that if a few variables in a group are important, then most of the variables in the same group should also be important. However, in many real-world applications, we may come to the opposite observation, i.e., variables in the same group exhibit negative correlations by competing with each other. For instance, in visual object recognition, the signature visual patterns of different objects tend to be negatively correlated, i.e., visual patterns useful for recognizing one object tent to be useless for other objects.

In Chapter 5, we will review the existing regularization methods to derive implicit group and hierarchical structures. We will also propose a new regularization which we call *exclusive lasso*. Different from the group lasso regularizer, if one feature in a group is given a large weight, the exclusive lasso regularizer tends to assign small or even zero weights to the other features in the same group. We will present a theoretical analysis to verify the exclusive nature of the proposed regularizer in detail.

2.8 Hybrid Methods

Some algorithms perform the structure learning in a hybrid manner to combine multiple structure learning methods. Here we list some examples.

- *Max-min Hill-climbing* (MMHC): In [Tsamardinos et al., 2006], the authors proposed a Max-min Hill-climbing (MMHC) algorithm for structure learning of Bayesian networks. The MMHC algorithm shares the similar idea as the Sparse Candidate Hill Climbing (SCHC) algorithm. The MMHC algorithm works in two steps. In the first step, the skeleton of the network is learned using a local discovery algorithm called Max-Min Parents and Children (MMPC) to identify the parents and children of each node through the conditional independency test, where the conditional sets are grown in a greedy fashion. In this process, the Max-Min heuristic is used, which selects the variables that maximize the minimum association with the target variable relative to the candidate parents and children. In the second step, the greedy hill-climbing search is performed within the constraint of the skeleton learned in the first step. Unlike the SCHC algorithm, MMHC does not impose a maximum in-degree for each node.

- In [Schmidt and Murphy, 2007], the authors proposed a structure learning approach which uses the $L1$ penalized regression to find the parents/neighbors for each node, and then apply the score-based search. The first step is the L_1 variable selection to find the parents/neighbors set of a node. The regression algorithm is discussed in Section 2.4.2.

  After the parent sets of all node are identified, a skeleton of the structure is created using the 'OR' strategy [Meinshausen and Bühlmann, 2006]. This procedure is called L1MB (L_1-regularized Markov blanket). The L1MB is plugged into structure search (MMHC) or ordering search [Teyssier and Koller, 2005]. In the application to MMHC, L1MB replaces the Sparse Candidate procedure to identify potential parents. In the application to ordering search in [Teyssier and Koller, 2005], given the ordering, L1MB

replaces the SC and exhaustive search.

CHAPTER 3

Inferring Functional Cortical Networks Using Dynamic Bayesian Networks

In this chapter, we discuss using dynamic Bayesian networks to identify functional cortical connectivity from simultaneously recorded spike trains. We first introduce the background of the research in this area, and then present our work on applying the dynamic Bayesian network to reconstructing functional cortical networks. Experiments with various configurations are conducted to verify the efficacy of DBN in recovering both linear and non-linear neural activities.

3.1 Introduction

Brain networks are of fundamental interest in system neuroscience. Numerous functional neuroimaging studies suggest that the neural circuits are temporarily configured to mediate perception, learning, sensory and motor processing [Winder et al., 2007, Koshino et al., 2005]. Understanding the highly-distributed nature of cortical information processing mechanisms remains challenging despite these significant findings. An essential step towards understanding how the brain orchestrates information processing is to uncover the topology of the neural circuits. The temporal and spatial resolution of current neuroimaging techniques do not enable the investigation of the brain's functional dynamics at the single cell level, and they

do not enable the identification of causal relationships between multi-units across multiple cortical regions.

Recent advances in the microfabrication of multi-electrode arrays (MAE) [Normann et al., 1999, Wise et al., 2004] have enabled scrutinizing activities from cortical neurons at an unprecedented scale, and greatly facilitated our ability to observe functional interactions of cortical networks in awake and behaving animals. The abundant data generated calls for new development of analytical tools that can infer the functional connectivity in these networks. Such information can yield more insight into how these networks respond collectively to dynamic complex stimuli, or to signify movement intention and execution through causal interactions between the observed neurons referred to as the effective connectivity [Aertsen et al., 1989]. Such tools can reveal more evidence in support of modern views suggesting that multisensory integration occur early in the neocortex, as opposed to the more traditional cognitive models of the sensory brain that contends higher level integration of unisensory processing. They can also help identify plastic changes in cortical circuitry during learning and memory [Martin and Morris, 2002] or post traumatic brain injury [Girgis et al., 2007].

3.2 Dynamic System Reconstruction

Sequential or time series data arises in many areas of science and engineering. Classical approaches to analyze sequential data include linear models, such as ARIMA, ARMAX, etc [Hamilton, 1994], or non-linear models, such as neural networks [Connor et al., 1994] or decision trees [Meek et al., 2002]. N-gram models [Jelinek, 1997] and variable-length Markov models [Ron et al., 1996] are frequently used for discrete data.

The classical methods suffer from several drawbacks. First, they make predictions of the future based on only a finite window into the past. Second, it is difficult to incorporate prior knowledge into the model. Third, the classical methods often have difficulties when the system has multi-dimensional input/output.

The state-space models try to solve these problems. In a state-space model, it is assumed that there is a underlying hidden state of the world that generates the observations. The hidden state evolves over time, possibly as a function of input. The state-space models are superior to the classical time-series modeling approaches in many aspects [Durbin and Koopman, 2001]. They overcome the problems mentioned above naturally.

One of the most commonly used state-space models is the Hidden Markov Model (HMM). However, HMMs are limited to its expressive power. Dynamic Bayesian Networks (DBNs) generalize HMMs by allowing the state space to be represented in factored form, instead of as a single discrete random variable. DBNs also generalize HMMs by allowing arbitrary probability distributions, not just (unimodal) linear-Gaussian. Besides the generality, DBNs interpret the interactions of multi-variates using DAGs, leading to a natural expression of the structures of the variables in the time-series background.

3.3 HMM and DBN

In this section, we first give a brief introduction to Hidden Markov Models followed by dynamic Bayesian networks. We then compare these two models and explain why we choose DBN for cortical network reconstruction.

3.3.1 Discrete Markov Process

In a discrete Markov process [Drake, 1967], the system belongs to one of a set of N distinct states at any time. We use $S_1, S_2, \cdots, S_N$ to denote these states. At regularly spaced discrete times, the state of the system changes (possibly back to the same state) according to a set of transition probabilities associated with the state. We use $t = 1, 2, \cdots$ to denote the time instants at each state change, and use q_t to denote the actual state at time t. The state of the system at each time t can be described as a probabilistic model of the past states. Specifically, for a discrete first order Markov chain, the probabilistic description involves only

the current and the preceding state,

$$P(q_t = S_j | Q_{t-1} = S_i, q_{t-2} = S_k, \cdots) = P(q_t = S_j | q_{t-1} = s_i)$$

We assume that the right-hand side of the above equation is independent of time, which leads us to a set of state transition probabilities a_{ij} of changing from state S_i to S_j:

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i), \quad 1 \leq i, j \leq N$$

The state transition coefficients obey standard stochastic constraints:

$$\begin{aligned} a_{ij} &\geq 0 \\ \sum_{i=1}^{N} a_{ij} &= 1 \end{aligned}$$

The above stochastic process can be called an observable Markov model since the output of the process corresponds to the state of the system.

3.3.2 Hidden Markov Model

In a Hidden Markov Model (HMM) [Rabiner, 1989], the internal state of a system is not directly observable. Instead, the observation O_t at time instant t comes from a set of M distinct observation symbols denoted as $V = \{v_1, v_2, \cdots, v_M\}$, and follows the observation symbol probability according to the state q_t. Specifically, a HMM can be described using the following elements:

- The number of distinct states N of the model. These states are denoted as $S = \{S_1, S_2, \cdots, S_N\}$ and they are *hidden* from the outside.
- The number of distinct observable symbols M. We denote the symbols as $V = \{v_1, v_2, \cdots, v_M\}$. They correspond to the output of the system.
- The state transition probability distribution $A = \{a_{ij}\}$:

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i), \quad 1 \leq i, j \leq N.$$

Algorithm 6 The procedure of a HMM

1: Start the system with an initial state $q_1 = S_i$ according to the initial state distribution π.
2: **repeat**
3: Generate an observation O_t according to the symbol probability distribution $b_i(k)$ at state s_i.
4: The system transits to a new state $q_{t+1} = S_j$ according to the state transition probability distribution at state S_i, i.e., a_{ij}.
5: Update time $t \leftarrow t + 1$
6: **until** $t = T$.

In a special case where each state does not reproduce itself, we have $a_{ij} > 0$ for all i, j.

- The observation probability distribution $B = \{b_j(k)\}$ where

$$b_j(k) = P(O_t = v_k | q_t = S_j), \quad 1 \leq j \leq N; \quad 1 \leq k \leq M.$$

- The initial state distribution of the system $\pi = \{\pi_i\}$:

$$\pi_i = P(q_1 = S_i), \quad 1 \leq i \leq N.$$

In all, a complete specification of a HMM includes two model parameters (N and M), specification of observation symbols, and the specification of the three probability distributions A, B, and π. We use the compact notion

$$\lambda = (A, B, \pi)$$

to indicate a complete parameter set of a HMM.

Given the specifications, a HMM generates a sequence of observations according to the procedure described in Algorithm 6.

3.3.3 Basic Usages of HMM

Given the Hidden Markov Model defined in Section 3.3.2, there are three basic problem of interest that should be solved for the model to be useful in real-world applications:

1. Given the observation sequence $O = O_1 \cdots O_T$ and the model $\lambda = (A, B, \pi)$, compute the probability of the sequence given the model $P(O|\lambda)$.

   This is the evaluation problem, namely, given a HMM and a sequence of observations, how to compute the probability that the observation is generated by the model. This can also be seen as how well a model fits the observations.

   Clearly we can enumerate all state transitions and sum up the corresponding observations, but the computation will be intractable using this method since the computational complexity will be $O(2TN^T)$. A much more efficient method is the forward-backward procedure [Baum and Sell, 1968]. The computation complexity of this method is $O(N^2T)$.

2. Given the observation sequence $O = O_1 \cdots O_T$ and the model λ, infer a corresponding state sequence $Q = Q_1 \cdots Q_T$ which best explains the observations. This can be solved using a dynamic programming method called the *Viterbi Algorithm* [Forney, 1973].

3. Given an observation sequence O, adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$. There is no analytical method to optimally estimate the model parameters. However, we can choose λ such that $P(O|\lambda)$ is locally maximized using an iterative procedure such as the Expectation-Maximization method [Dempster et al., 1977] (also called the Baum-Welch method when specifically applied to HMMs [Baum et al., 1970]) or using gradient techniques [Rabiner, 1989].

3.3.4 Limitations of HMM

Despite its versatility in many applications, Hidden Markov Models suffer from several limitations:

- Interpretability: the parameters associated with HMMs are interpretable in so far as they are clearly labeled as *transition* or *emission* probabilities, however, the states of a HMM do not always have a clear interpretation, especially after training.

- Factorization: in the standard definition, HMMs are fundamentally unfactored. If the state of the system consists of a combination of factors, it can not be represented concisely.

- Extensibility: HMMs are limited in their extensibility in that the main way to increase their complexity is simply to increase the number of states. This can be unwieldy when the overall state of the system is actually composed of a combination of separately identifiable factors.

3.3.5 Dynamic Bayesian Network

A dynamic Bayesian network (DBN) [Dean and Kanazawa, 1990, Murphy, 2002] is an extension of Bayesian Network and Hidden Markov Model to model probability distributions of random variables over time. Here we only consider discrete-time stochastic processes. Note that the term *dynamic* means that we are modeling a dynamic system, not that the network changes over time. In other words, the model is time-homogeneous, both the network topology and the model parameters are time-invariant.

The topology of a DBN is defined as a directed acyclic graph (DAG), but we allow a node to have an arc to itself. In this case the node is persistent. This is practically useful in some situations, for example, to model the self-inhibitory or self-excitatory effect of neurons as described later for the application of DBN.

The parameters of a DBN is defined as (B, A), where B defines the prior $P(X_1)$, and A defines temporal conditional probability as:

$$P(X_t|X_{t-1}) = \prod_{i=1}^{N} P(X_t^i|\pi(X_t^i))$$

where X_t^i is the i-th node at time t, and $\pi(X_i^i)$ are the parents of variable X_t^i. Each node is conditionally independent of other nodes given its parents. The conditional probability distribution (CPD) may have various forms similar to those of Bayesian networks, but now over time slices. Generally we assume that the model is a first-order Markov, i.e., the parents

$\pi(X_t^i)$ of a node i can either be in the same time slice or in the previous time slice. Higher order Markov is sometimes used when we want to study the influence of variables over a longer period.

The difference between DBN and HMM is that DBN represents the hidden states using a set of random variables, i.e., it uses a distributed representation of state. In contrast, in HMM, the state space consists of a single random variable. This gives us the advantage for learning a DBN because many algorithms described in Section 2.1.4 are also applicable to DBN. A detail description of learning DBN can be found in [Ghahramani, 1998].

DBN is also advantageous over HMM in the following aspects:

- Interpretability: Each variable in a DBN represents a specific concept.
- Factorization: The joint distribution of the variables in a DBN is factorized, leading to the following benefits:
  - Statistical efficiency: compared to an unfactored HMM with an equal number of possible states, a DBN with factored state representation and sparse connections between variables will require exponentially fewer parameters.
  - Computational efficiency: depending on the exact graph topology, the reduction in model parameters may be reflected in a reduction in running time.
- DBNs have precise and well-understood probabilistic semantics. The combination of theoretical underpinning, expressiveness, and efficiency bode well for the future of DBNs in many applications.

3.4 Experiments

We present the empirical study with the reconstruction of cortical network in this section to verify the efficacy of DBNs. First we present the population model used in our experiments to generate the spike trains. Then we give a brief introduction of a baseline algorithm

which is frequently used in the study of this area. Finally we demonstrate the application of dynamic Bayesian networks to reconstructing the functional neuronal circuits under various configurations.

3.4.1 Population Model

Given the limited data that is available for evaluating algorithms for neuron network reconstruction, it is essential to develop a model that provides appropriate simulation for the spike train data in which the ground truth is known. In this study, we follow [Truccolo et al., 2005] by using multivariate point process models for data simulation. Specifically, we use a variant of the generalized linear model (GLM) [Truccolo et al., 2005]. The firing probability of a neuron at a certain time point is determined by both the background level of activity and the spiking history of the neuron itself and its pre-synaptic neurons connected to it. Consider a population with N neurons, a linear model of the conditional intensity function $\lambda_i(t|\alpha_i, H_t)$ of neuron i at time t is mathematically expressed as:

$$\lambda_t(t|\alpha_i, H_t)\Delta = \left(\exp\left(\beta_i + \sum_{j\in\pi(i)} \sum_{m=1}^{M_{ij}} \alpha_{ij}(m\Delta) S_j(t - m\Delta)\right)\right) \cdot \Delta \tag{3.1}$$

where β_i is the log of the background firing rate of neuron i, α_{ij} is the synaptic interaction, excitatory or inhibitory, from neuron j to neuron i, $\pi(i)$ is the set of pre-synaptic neurons of neuron i, H_t is the spiking history divided into M_{ij} non-overlapping windows each with the bin width Δ. $S_j(t - m\Delta)$ is the number of spikes fired by neuron j in time window m. The spiking history interval of the interaction between neuron i and j is $M_{ij} \cdot \Delta$.

To describe higher degree of non-linearity, a multiplication-based model is expressed as:

$$\lambda_t(t|\alpha_i, H_t)\Delta = \left(\exp\left(\beta_i + \prod_{j\in\pi(i)} \sum_{m=1}^{M_{ij}} \alpha_{ij}(m\Delta) S_j(t - m\Delta)\right)\right) \cdot \Delta \tag{3.2}$$

The difference between Equation 3.1 and 3.2 is that the linear summation of the pre-synaptic influences is replaced by a product. The latter is used to model the coincidence

detection where the post-synaptic neuron fires only when its pre-synaptic neurons fire synchronously.

To model the influence of excitatory post-synaptic potential and inhibitory post-synaptic potential, we use the following exponential functions for synaptic coupling [Sprekeler et al., 2007, Zhang and Carney, 2005]:

$$\alpha_{ij}^{\pm}(t) = \begin{cases} 0 & \text{if } t < l_{ij}\Delta \\ \pm A_{ij} \exp\left(-3000(t - l_{ij}\Delta)/M_{ij}\right) & \text{if } t \geq l_{ij}\Delta \end{cases} \tag{3.3}$$

where $\pm$ indicates excitatory/inhibitory interactions and A_{ij} is the strength of the connection between i and j, l_{ij} is the synaptic latency from neuron j to i. The decaying exponential function in this model indicates that the influence of a spike from a pre-synaptic neuron declines with time.

3.4.2 Granger Causality

We give a short introduction of Granger causality [Kamiński et al., 2001] here since it is commonly used in analyzing neural data. It will be used in the experiment as a baseline to compare with dynamic Bayesian networks in inferring non-linear interactions.

Granger causality assumes that if two neurons are connected, then the firing history of the pre-synaptic neuron should help to linearly predict the firing occurrences of the post-synaptic neuron [Seth, 2008]. In particular, the spike train of neuron S_i at time t can be predicted given its own history and the history of a pre-synaptic neuron S_j as:

$$S_i(t) = \sum_{k=1}^{K} A_{ii}(k) S_i(t-k) + \sum_{k=1}^{K} A_{ij}(k) S_j(t-k) + \epsilon_{i|j}(t) \tag{3.4}$$

where K is the maximum number of lagged observations and A_{ij} is the least square regression coefficient when S_j is used to predict S_i.

Neuron j is said to *Granger cause* neuron i if including S_j in the Equation 3.4 reduces the variance of the prediction error. The strength of interaction from neuron j to i is measured

by the Granger causality index (GCI) [Ganapathy et al., 2007] as:

$$GCI(i,j) = 1 - \frac{\text{var}(\epsilon_{i|j})}{\text{var}(\epsilon_i)}$$

where ϵ_i is the prediction error when only the firing history of neuron i is considered in the model. A CGI value of 0 indicates no causality. When the CGI is statistically significant, neuron j is considered to influence the firing of neuron i in the model.

3.4.3 Experimental Settings

We simulated neuron spike trains using the point process model in Equation 3.1. For each parameter setting, we generated 100 random networks each containing 10 neurons with randomly generated connections. The pre-synaptic neurons were drawn from a uniform distribution. Each neuron also has a self-inhibitory connection. The duration of the spike trains is 1 minute with bin width of 3 ms.

We used the Bayesian Network Inference with Java Objects (BANJO) toolbox [Smith et al., 2006] in our experiments. It is an open source DBN structure learning software. We used the BDe score function and simulated annealing search.

We used the F-measure to evaluate the experimental performance. It is defined as the harmonic mean of precision and recall:

$$\begin{aligned}
\text{precision} &= \frac{\text{\# of correctly inferred connections}}{\text{\# of inferred connections}} \\
\text{recall} &= \frac{\text{\# of correctly inferred connections}}{\text{\# of connections in the network}} \\
F &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\end{aligned}$$

3.4.4 Experiments with Excitatory Connections

We thoroughly investigated the performance of DBN in inferring connections in the network at different settings for the parameters in Equation 3.1 and 3.3. The synaptic latency l_{ij} is

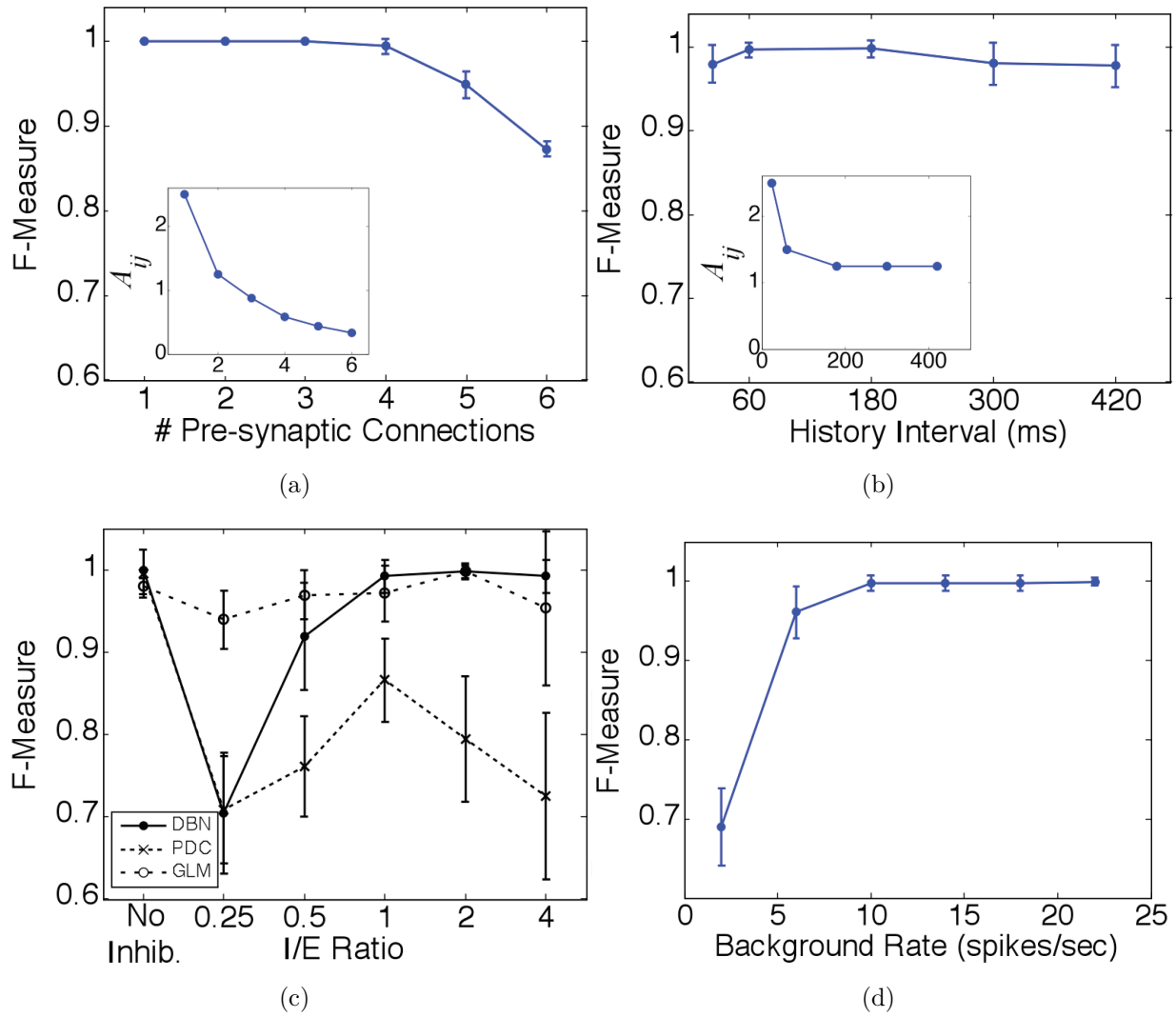


Figure 3.1: The performance of using DBN to infer neuron connections at fixed latencies. (a) performance at different number of excitatory connections. (b) performance at different firing history interval length. (c) performance at different inhibition/excitation ratio. (d) performance at different background rate.

fixed at 3 ms for all neurons. This mimics *direct* connectivity that may exist in local population rather than diffuse connections that may be observed across different parts of cortex. Figure 3.1 shows the performance at different settings of the model parameters when the DBN Markov lag is set to match the synaptic latency. Each point in the figure represents the mean and standard deviation of the inference accuracy for 100 different network structures.

We first examined the performance as the number of pre-synaptic neurons varies from 1

to 6 while fixing all other parameters. All connections here are excitatory with a history interval of 180 ms. The connection strength A_{ij} in Equation 3.3 is decreased as shown in Figure 3.1(a) when the number of pre-synaptic neurons is increased in order to keep a stable average firing rate in the range of 20 to 25 spikes per second and prevent unstable network dynamics while the background rate of each neuron was set to 10 spikes per second. The results in Figure 3.1(a) shows that DBN achieves 100% accuracy with a slight decline above 4 pre-synaptic connections. This decline can be attributed to the decrease in the synaptic strength A_{ij}, thereby reducing the influence of a pre-synaptic spike on the firing probability of the post-synaptic neuron.

We also varied the history interval of the interaction by varying M_{ij} and examined the performance. Each neuron receives two excitatory connections of equal weights from two neurons. The weights were adjusted in order to keep the average firing rate in the same range for different settings of M_{ij} as illustrated in Figure 3.1(b). Since the synaptic latency is fixed among all neurons and is consistent with the DBN Markov lag used, we expected the performance to be invariant to changes in the history interval length. This is confirmed in Figure 3.1(b) as we observed the performance of DBN remain almost unchanged as history interval is increased.

3.4.5 Experiments with Inhibitory Connections

We further examined the performance of using DBN to infer networks containing inhibitory connections. When inhibitory connections exist, the inference of network becomes more complicated, particularly for hyperexcitable neurons, due to the fact that the neuron's spiking pattern becomes very sparse. As a result, observing a spike event may contain a lot more information in the presence of inhibitory connections with variable degrees of strength. In our experiments, each neuron received two pre-synaptic connections: one excitatory and one inhibitory. The excitatory synaptic strengths in Equation 3.3 were fixed at 2.5, while those of the inhibitory connections were varied. We defined *I/E ratio* as the ratio of the

cross-inhibitory to cross-excitatory synaptic strength and tested ratios at 0.25, 0.5, 1, 2 and 4 respectively. All other parameters were set same as above. Taking the self-inhibition mechanism inherent in our model into account ($A_{ii} = -2.5$), an I/E ratio of 4 corresponds to a post-synaptic neuron with equal degrees of inhibition and excitation. Such neurons are expected to exhibit homogeneous (Poisson-like) characteristics reminiscent of independent firing. Figure 3.1(c) illustrates the inference accuracy with respect to the I/E ratio. For ratios below 1, a neuron is effected more by the excitatory connection than by the cross-inhibition connection. Therefore, the drop in performance observed suggests that DBN is unable to detect the presence of *weak* inhibition in the presence of a *strong* excitation. This can be attributed to the relatively insignificant effect of weak inhibitory input on the firing characteristics of post-synaptic neurons conditioned on receiving strong excitation. When the I/E ratio increases above 1, the accuracy rapidly gets closer to unity and does not deteriorate even when the inhibitory connections are 4 times stronger than the excitatory connections.

To interpret these results, we used the coefficient of variation (CV) of the inter-spike interval (ISI) histograms (CV=std(ISI)/mean(ISI)) to quantify the variability in the firing characteristics of post-synaptic neurons for variable I/E ratios. We hypothesized that if weak inhibitory connections do not significantly influence the firing of these neurons, then the CV should be close to those receiving pure excitation. Moreover, one would not expect to see a sharp transition in CV characteristics for I/E values adjacent to purely-excitatory connections. Figure 3.1(d) demonstrates that this is indeed the case. The average CV for I/E ratios of 0.25 is not significantly different from that of purely excitatory connections. This suggests that weak cross-inhibition did not result in strong effects on the post-synaptic neurons' firing characteristics, making it more difficult for DBN to detect this type of connectivity. The CV monotonically increases as the I/E ratio increases, reaching an average of 1 (similar to that of an independent Poisson neuron) when cross-inhibition is on average 4 times stronger than excitation (I/E ratio = 4).

We compared the performance of DBN to a related measure of connectivity, Partial Directed Coherence (PDC), which has been proposed to study causal relationships between signal sources at coarser resolution in EEG and fMRI data [Sameshima and Baccalá, 1999]. PDC is the frequency domain equivalence of Granger causality that is based on vector autoregressive models of certain order. A connection is inferred between a pair of neurons if the PDC at any frequency exceeds a threshold of 0.1. For each network, PDC was applied using models of orders 1 to 30. The accuracy shown by the dashed plot in Figure 3.1(c) represents the maximum accuracy achieved across all model orders. As can be seen, DBN outperformed PDC over the entire range of the I/E ratio, while exhibited similar performance around an I/E ratio of 0.25. Closer examination of the networks inferred using PDC for I/E ratios greater than 0.25 revealed that the deterioration in the PDC performance compared to the DBN was due to its inability to detect most of the inhibitory connections, consistent with previous findings with spectral coherence as well as Granger causality [Cadotte et al., 2008].

We also compared DBN with Generalized Linear Model (GLM) fit [Truccolo et al., 2005, Czanner et al., 2008]. Ideally, GLM fit should yield the best result for our data since this is the generative model we used in Equation 3.1. A maximum likelihood estimate of the coupling function α_{ij} in Equation 3.3 for each neuron i is computed in terms of the spiking history of all other neurons j within a certain window of length WGLM. Since our goal is to identify the connections and not to estimate the coupling function, we post-processed the estimated coupling functions such that a connection was inferred if the estimated coupling function was larger/lower than a given threshold for excitatory/inhibitory connections, respectively, for 3 consecutive time slots while their p-values were significant. The spiking history considered for fitting WGLM was set to 60 ms. The threshold was varied between 0 and ± 1.5 and the p-value between 0.1 and 0.0001. The dotted plot in Figure 3.1(c) shows the maximum accuracy obtained across all thresholds and p-values. Superior performance of the GLM can be seen compared to DBN at I/E ratios below 1, while comparable if not slightly inferior for I/E ratios above 1. We note, however, that the GLM approach has a number of limitations:

First, the performance is highly dependent on the choice of the fitting spiking history interval WGLM. Second, the inference threshold and the p-values have to be carefully set to identify the connections. Finally, the search time of GLM method was approximately 20 times that of DBN to estimate the coupling functions for each 10-neuron population.

Finally, we examined the performance with respect to the background rate in Equation 3.1. This may mimic variations in the afferent input current to the neuron, and increments in this input when there is no coupling to other neurons known to impact the estimates of correlation between their output spike trains [de La Rocha et al., 2007]. This is because the correlation between a neuron pair can not be orthogonally separated from their firing rates, thereby potentially leading to spurious connectivity inference [Amari, 2009]. Here we have 2 pre-synaptic connections per neuron, one excitatory and one inhibitory with the same firing strength. Figure 3.1(d) shows that the inference accuracy was above 96% for background rates higher than 5 spikes per second. Most remarkable is the ability of DBN to infer roughly 70% of the connections at background rates around 2 spikes per second, despite that neurons are silent most of the time at this low rate.

3.4.6 Experiments with Non-linear Connections

We also find that the DBN model is especially effective in inferring non-linear connections. We generated a spike train data set using the multiplicative integration model of the pre-synaptic inputs given in Equation 3.2. Figure 3.2 shows a network in which neuron 3 acts as a coincidence detector of neurons 1 and 2 spikes. The firing rate of neurons 1 and 2 was set to 30 Hz. The resulting firing rate of neuron 3 was observed to be about 4 Hz. Despite the low firing rate of neuron 3, DBN was able to discover the causal relationships between neurons 1 and 2 and neuron 3 as shown in Figure 3.2. When Granger causality was applied to the same population, the average of $GCI(3,1)$ and $GCI(3,2)$ were 0.21, which were statistically insignificant. This further confirms that linear inference algorithms lead to erroneous conclusions when applied to highly non-linear situations.

3.5 Conclusion

In this chapter we study the learning of structures from time-series or sequential data by exploring DBNs. We demonstrated the use of DBNs in identifying the structures of neural circuits from observed spike trains. DBN can identify direct synaptic connections between distinct neuronal elements. It can also identify the direction of those connections. We applied the method to probabilistic neuronal circuit models that mimic the stochastic variability experimentally observed in the discharge pattern of cortical neurons. The experimental results demonstrate the capability of DBN to identify direct connections in multiple networks of various characteristics.

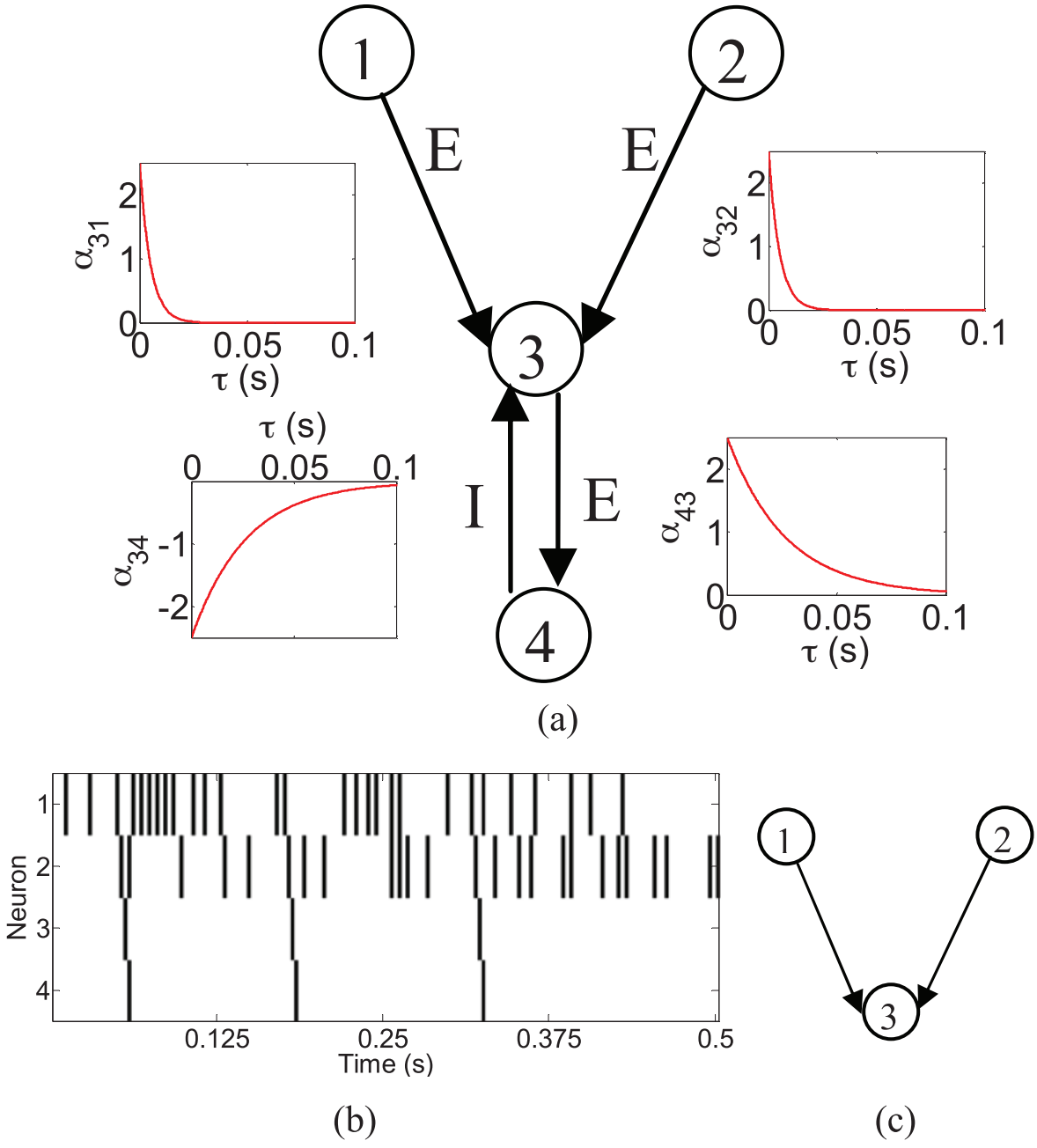


Figure 3.2: (a) Network topology for the inhibitory feedback case. α_{ij} for each synaptic connection is shown in the inserts beside each connection. (b) A 0.5 sec trace of the obtained spike trains. (c) Network inferred using DBN.

CHAPTER 4

Using Knowledge Driven Matrix Factorization to Reconstruct Modular Gene Regulatory Network

4.1 Introduction

Reconstructing gene regulatory network from the micro-array data is important for understanding the underlying mechanism behind cellular processes. A number of computational methods have been developed or applied to automatically reconstruct gene networks from gene expression data. Clustering methods, such as hierarchical clustering, K-means and self-organizing map [Eisen et al., 1998], are commonly used to identify gene modules. The main disadvantage of clustering methods is that they are unable to uncover the interaction among different modules, which is crucial to the understanding of disease mechanisms. To overcome this problem, several studies have proposed to integrate clustering methods with structure learning algorithms. In [Toh and Horimoto, 2002], the authors combined a clustering method with the Graphical Gaussian Model (GGM) for module network reconstruction. In [Segal et al., 2003], a Bayesian framework is presented to integrate a clustering method with Bayesian network learning. A disadvantage with these approaches is that they rely solely on gene expression data, which are noisy, in the analysis. Furthermore, as revealed by several studies [Husmeier, 2003, Yu et al., 2004], structure learning methods tend to perform poorly when the number of experimental conditions is significantly smaller than the number

of genes.

In the past, a large number of studies have been devoted to exploiting prior knowledge for gene network reconstruction to alleviate the problem that expression data are often sparse and noisy [Bar-Joseph et al., 2003, Berman et al., 2002, Hartemink et al., 2002, Ideker et al., 2001, Ihmels et al., 2002, Pilpel et al., 2001, Li and Yang, 2004]. A typical approach is to construct a Bayesian prior for the directed arcs in the Bayesian network using the prior knowledge of regulator-regulatee relationships that are derived from other data such as location analysis data and protein interaction data. A problem with this type of approach is that it is often difficult to extend them to incorporate the co-regulation relationships that can be easily derived from the GO database. This is a shortcoming with Bayesian network analysis especially for mammalian systems, where interaction data are not as readily available, whereas GO information are. Therefore, developing a framework of knowledge driven analysis with high-throughput data that effectively exploits the prior knowledge of co-regulation relationships from GO could enhance the robustness of the network reconstruction from gene expression data.

The key challenge with using GO for network reconstruction is that the co-regulation relationships derived from GO may be noisy and inaccurate. In this paper, we propose a framework for gene modular network reconstruction based on the **Knowledge driven Matrix Factorization** (**KMF**) that is able to effectively exploit the prior knowledge derived from GO. The key features of the proposed framework are

- It derives both the gene modules and their interaction from a combination of expression data and the GO database,
- It incorporates the prior knowledge of co-regulation relationships into network reconstruction via matrix regularization,
- It presents an efficient learning algorithm that combines the techniques of non-negative matrix factorization and semi-definite programming.

It is important to note that although our framework is closely related to other algorithms for matrix factorization (e.g., non-negative matrix factorization), they differ significantly in both their computational methods and goals. First, unlike the existing algorithms for matrix factorization that are designed either for clustering or for dimensionality reduction, our framework aims to learn a module network structure from gene expression data. Second, unlike other matrix factorization algorithms that solely depend on iterative algorithms for optimization, the proposed framework exploits both convex and non-convex optimization strategies for finding the optimal network structure.

4.2 Matrix Factorization

Matrix factorization is a unifying theme in numerical linear algebra. A wide variety of matrix factorization algorithms have been developed, including LU decomposition, QR factorization, spectral decomposition, singular value decomposition (SVD), etc. [Meyer, 2000] These algorithms provide a numerical platform for matrix operations such as solving linear systems, spectral decomposition, subspace identification and statistical data analysis.

Recent work in machine learning has focused on matrix factorization that directly target some of the special features of statistical data analysis. In particular, non-negative matrix factorization (NMF) [Lee and Seung, 1999, 2001] and maximum margin matrix factorization (MMMF) [Srebro et al., 2005] have been popular in recent years. Non-negative matrix factorization focuses on the analysis of data matrices whose elements are non-negative which is common in statistical data sets. It yields non-negative factors which is advantageous for interpretation. Maximum margin matrix factorization uses low-norm factorization and is strongly connected with large-margin linear discrimination. It has been shown to be effective in collaborative filtering [Rennie and Srebro, 2005].

4.2.1 Non-negative Matrix Factorization and Extensions

For a non-negative $n \times m$ matrix V, the non-negative matrix factorization (NMF) [Lee and Seung, 1999, 2001] aims to find non-negative matrix factors W and H such that:

$$V \approx WH \tag{4.1}$$

Where W is an $n \times r$ matrix and H is $r \times m$. Usually r is set to be smaller than n and m, such that the ranks of W and H are smaller than that of the original matrix V, leading to the compression of the original data matrix.

Equation 4.1 can be written column by column as $V_{*k} \approx WH_{*k}$ where V_{*k} and H_{*k} are the corresponding column k of V and H, which means each data vector V_{*k} is approximated by a linear combination of the columns of W. In other words, the columns of W contain a basis for the linear approximation of the data V.

In [Lee and Seung, 2001], the authors proposed two cost functions to measure the distance between two functions A and B. One measure is the square of the Euclidean distance,

$$\|A - B\|^2 = \sum_{ij} (A_{ij} - B_{ij})^2 \tag{4.2}$$

Another measure is

$$D(A||B) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right) \tag{4.3}$$

which is equivalent to Kullback-Leibler divergence, or relative entropy, when $\sum_{ij} A_{ij} = \sum_{ij} B_{ij} = 1$. Both measures are non-negative and vanishes if and only if $A = B$.

Non-negative matrix factorization has been used in dimensionality reduction [Hoyer and Dayan, 2004], classification [Sha et al., 2007].

Divergence D_ϕ	ϕ	α	β	Reference
$\|A - BC\|_F^2$	$\frac{1}{2}x^2$	0	0	Lee and Seung [1999, 2001]
$\|A - BC\|_F^2$	$\frac{1}{2}x^2$	0	$\lambda \mathbf{1}^\top C \mathbf{1}$	Hoyer [2002]
$\|W \bigodot (A - BC)\|_F^2$	$\frac{1}{2}x^2$	0	0	Paatero and Tapper [1994]
$\mathrm{KL}(A, BC)$	$x \log x - x$	0	0	Lee and Seung [2001]
$\mathrm{KL}(A, WBC)$	$x \log x - x$	0	0	Guillamet et al. [2001]
$\mathrm{KL}(A, BC)$	$x \log x - x$	$c_1 \mathbf{1}^\top B^\top B \mathbf{1}$	$-c_2 \|C\|_F^2$	Feng et al. [2002]
$D_\phi(A, W_1 B C W_2)$	$\phi(x)$	$\alpha(B)$	$\beta(C)$	Dhillon and Sra [2005]

Table 4.1: Variations of NMF algorithms with different divergence measures. Revised from Dhillon and Sra [2005].

Extensions of Non-negative Matrix Factorization

- The general form of non-negative matrix factorization, i.e., to approximate a non-negative matrix A using the product of two non-negative matrices B and C, is:

$$\min_{B,C \geq 0} \quad D_\phi(BC, A) + \alpha(B) + \beta(C) \tag{4.4}$$

$$\min_{B,C \geq 0} \quad D_\phi(A, BC) + \alpha(B) + \beta(C) \tag{4.5}$$

  The function $\alpha(\cdot)$ and $\beta(\cdot)$ are penalty functions which enforce regularization or other constraints on B and C, and D_ϕ is the Bregman divergence (see Appendix C).

  Variations from Equation 4.5 and 4.5 lead to many NMF algorithms, some of which are briefly list in Table 4.1. In the table, $\mathrm{KL}(x, y)$ denotes the generalized KL divergence:

$$\mathrm{KL}(x, y) = \sum_i x_i \log \frac{x_i}{y_i} - x_i + y_i$$

- In [Hoyer and Dayan, 2004], the authors extend NMF to explicitly control sparseness. The authors argue that this my discover parts-based representations that are

qualitatively better than those given by basic NMF. Their algorithm is defined as:

$$\begin{aligned} \underset{W\in\mathbb{R}_+^{n\times r},H\in\mathbb{R}_+^{r\times m}}{\arg\min} \quad & \|V-WH\|_F^2 \\ \text{s.t.} \quad & \text{sparseness}(\mathbf{w}_i)=S_w,\forall i \\ & \text{sparseness}(\mathbf{h}_i)=S_h,\forall i \end{aligned}$$

where $\mathbf{w}_i$ is the i-th column of W, $\mathbf{h}_i$ is the i-th row of H, and sparseness$(\cdot)$ is defined as:

$$\text{sparseness}(\mathbf{x})=\frac{\sqrt{n}-(\sum|x_i|)/\sqrt{\sum x_i^2}}{\sqrt{n}-1}$$

- In [Ding et al., 2008], the authors extend NMF to semi-NMF and convex-NMF. The data matrix X is allowed to have mixed signs when factored as

$$X\approx FG^\top$$

The semi-NMF is motivated from the perspective of clustering. Suppose we do K-means clustering on X and obtain cluster centroids $F=(\mathbf{f}_1,\cdots,\mathbf{f}_k)$. G is the cluster indicators such that $g_{ik}=1$ if $\mathbf{x}_i$ belongs to cluster c_k and $g_{ik}=0$ otherwise. Then the K-means clustering objective function is

$$J=\sum_{i=1}^{n}\sum_{k=1}^{K}g_{ik}g_{ik}\|\mathbf{x}_i-\mathbf{f}_k\|^2=\|X-FG\|_F^2$$

The optimization problem can be relaxed by allowing g_{ij} to range over $(0,1)$ or $(0,\infty)$, leading to the form of semi-KMF.

The convex-NMF imposes a constraint on the basis vectors $F=(\mathbf{f}_1,\cdots,\mathbf{f}_k)$ such that they lie within the convex hull of X, namely:

$$\mathbf{f}_\ell=w_{1\ell}\mathbf{x}_1+\cdots w_{n\ell}\mathbf{x}_n=X\mathbf{w}_\ell,\quad \text{or } F=XW.$$

Thus the data matrix X is factored as:

$$X\approx XWG^\top$$

where W and G are both non-negative matrices. As indicated in [Ding et al., 2008], convex-NMF tends to generate very sparse factors W and G.

4.2.2 Maximum-Margin Matrix Factorization

Given a matrix of binary labels, $Y \in \{\pm 1\}^{n \times m}$, the Maximum-Margin Matrix Factorization [Srebro et al., 2005] seeks to approximate it using a matrix X, while minimizing the trade-off between the trace norm of X and its hinge-loss relative to Y:

$$\underset{X \in \mathbb{R}^{n \times m}}{\arg\min} \ \|X\|_\Sigma + c \sum_{ij} \max(0, 1 - Y_{ij} X_{ij}) \tag{4.6}$$

Lemma 4.2.1. $\|X\|_\Sigma = \min_{X=UV'} \|U\|_F \|V\|_F = \min_{X=UV'} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2)$.

Lemma 4.2.2. *For any $X \in \mathbb{R}^{n \times m}$ and $t \in \mathbb{R}$, $\|X\|_\Sigma \leq t$ iff there exists $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{m \times m}$ such that $\left[\begin{smallmatrix} A & X \\ X' & B \end{smallmatrix}\right] \succcurlyeq 0$ and $\operatorname{tr}(A) + \operatorname{tr}(B) \leq 2t$.*

The proof of Lemma 4.2.2 can be found in [Srebro et al., 2005].

Using Lemma 4.2.2, Equation 4.6 can be written as:

$$\begin{aligned} \underset{A,B}{\arg\min} \quad & \frac{1}{2}(\operatorname{tr}(A) + \operatorname{tr}(B)) + c \sum_{ij} \xi_{ij} \\ \text{s.t.} \quad & \begin{bmatrix} A & X \\ X' & B \end{bmatrix} \succcurlyeq 0, \\ & y_{ij} X_{ij} \geq 1 - \xi_{ij}, \\ & \xi_{ij} \geq 0 \end{aligned} \tag{4.7}$$

The dual of the above problem is:

$$\begin{aligned} \underset{Q}{\arg\max} \quad & \sum_{ij} Q_{ij} \\ \text{s.t.} \quad & \begin{bmatrix} I & (-Q \otimes Y) \\ (-Q \otimes Y)' & I \end{bmatrix} \succcurlyeq 0, \\ & 0 \leq Q_{ij} \leq c \end{aligned} \tag{4.8}$$

which is equivalent to

$$\begin{aligned} \arg\max_{Q} \quad & \sum_{ij} Q_{ij} \\ \text{s.t.} \quad & \|Q \otimes Y\|_2 \leq 1, \\ & 0 \leq Q_{ij} \leq c \end{aligned} \tag{4.9}$$

Maximum-Margin Matrix Factorization has been used in collaborative filtering [Srebro et al., 2005].

4.2.3 Limitations with Existing Matrix Factorization Methods

Although many algorithms have been proposed for matrix factorization and these algorithms have been used in many applications, there are a number of challenges remain to be addressed:

- A major problem with matrix factorization is the computational cost of the related optimization problem. Since the matrix factorization problems mentioned above do not have closed form solution, iterative updating methods are used to find the solutions. And usually the algorithms require many iterations to reach convergence. This hinders matrix factorization from been used with large-scale datasets.

- Due to the greedy nature of the iterative updating methods, no global optimum is guaranteed. Instead these algorithms only lead to local optima, which make these algorithms have low numerical stability.

- An important parameter in the matrix factorization is the dimension of the resulting low rank matrix. When matrix factorization is used for clustering, this parameter corresponds to the number of clusters. As previous studies suggested [Tibshirani et al., 2001], this parameter has significant influences on the stability of the solutions. However, it has not been well addressed in existing matrix factorization studies.

- Although matrix factorization algorithms have been used in structure learning problems like gene regulatory network reconstruction [Carmona-Saez et al., 2006, Dueck et al., 2005, Schachtner et al., 2008], they are often limited in the expression power in the following aspects:
  - The existing matrix factorization algorithms are limited to using only one information source. However, in some applications we my wish to combine multiple information sources for more robust solutions. In the study of reconstructing gene regulatory networks, it has been shown that expression profile of a transcription factor and its regulated genes do not often show obvious correlation while the regulated genes often show strong correlation among themselves, which may result in poor network reconstruction result given sparse micro-array data [Husmeier, 2003].
  - In reconstructing gene networks, we are interested not only in the gene modules that are co-regulated, but also in the interactions among the modules. The existing matrix factorization algorithms are unable to infer this information.
  - The existing matrix factorization algorithms are limited in embedding the network topology in inferring the structures. As revealed by previous studies [Bhan et al., 2002, Jeong et al., 2000], the structure of many gene networks appears to be hierarchical and scale-free. In particular, genes are first clustered into modules, and the gene modules are then connected by scale-free networks. The most important feature of a scale-free network is that most nodes in the network are connected to a few neighbors, and only a small number of nodes, which is often called "hubs", are connected to a large number of nodes. Compared to other network structures, a scale-free network has a very skewed degree distribution, and therefore a relatively smaller number of edges [Barabasi and Albert, 1999]. This fact implies that the network structure matrix C should be a *sparse matrix* of which most elements are small or close to zero.

In the following sections, we address the above limitations by proposing a knowledge driven matrix factorization (KMF) framework for reconstructing modular gene networks. In KMF, gene expression data is initially used to estimate the correlation matrix. The gene modules and the interactions among the modules are derived by factorizing the correlation matrix. The prior knowledge in GO is integrated into matrix factorization to help identify the gene modules. An alternating optimization algorithm is presented to efficiently find the solution. Experiments show that our algorithm performs significantly better in identifying gene modules than several state-of-the-art algorithms, and the interactions among the modules uncovered by our algorithm are proved to be biologically meaningful.

4.3 A Framework for Knowledge Driven Matrix Factorization (KMF)

The following terminology and notations will be used throughout the rest of this chapter. Let m be the number of experimental conditions, and $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \in \mathbb{R}^m$ be the expression levels of the ith gene measured under m conditions. Let n be the number of genes, and $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ include the expression levels of all n genes. Given the expression data, we can estimate the pairwise correlation between any two genes. A number of statistical correlation metrics can be used for this purpose, such as Pearson correlation, mutual information, and chi-square statistics [Yang and Pedersen, 1997]. The computation results in a symmetric matrix $W = [w_{ij}]_{n \times n}$ where w_{ij} measures the correlation between gene $\mathbf{x}_i$ and $\mathbf{x}_j$.

The main idea behind the knowledge driven matrix factorization framework is to compute the network structure by factorizing the correlation matrix W into the matrices for gene modules (i.e., the module matrix) and the module network structure (i.e., the network matrix). We denote by M the module matrix, of which element M_{ij} represents the confidence of assigning the ith gene to the jth module. We denote by C the network matrix, of which

element C_{ij} represents the interaction strength between module i and module j. Note that the computational problem addressed here is fundamentally different from the problems addressed by the previous studies of matrix factorization [Lee and Seung, 2001] that mainly focused on dimensionality reduction and data clustering.

To determine the gene modules (i.e., M) and their network structure (i.e., C), we consider the following three criteria when formulating the framework of knowledge driven matrix factorization for hierarchical module network reconstruction:

1. The module matrix M and the network structure matrix C should be combined to accurately reproduce the correlation matrix W. This is based on the assumption that gene correlation information can essentially be explained by the gene modules and their interaction.

2. The module matrix M is expected to be consistent with the prior knowledge collected from Gene Ontology. In particular, two genes that bear a large similarity in gene functions as described in GO are likely to be assigned into the same module.

3. The network structure matrix C is expected to be consistent with the hierarchical scale-free structure of gene networks. As suggested in [Barabasi and Albert, 1999], a network with hierarchical scale-free structure tends to have a small number of linkages in total. We thus expect matrix C to be a sparse matrix with most of its elements being small and close to zero.

In the following subsections, we will first discuss how to capture the above three criteria in formulating the objective function, followed by the description of the full framework.

4.3.1 Matrix Reconstruction Error

Before discussing the reconstruction error, we need to first describe how to approximate the gene correlation matrix W by the gene modules and the module network. We assume that the correlation between two genes $\mathbf{x}_i$ and $\mathbf{x}_j$ arises because of the interaction between the two

modules that are associated with the two genes. Hence, variable W_{ij} can be approximated by $\sum_{a,b} M_{ia}M_{jb}C_{ab}$ where M_{ia} and M_{jb} represent the association of genes to gene modules and C_{ab} represent the interaction between modules a and b. In the form of matrices, the above idea is summarized as to approximate W by the product $M \times C \times M^\top$. We denote by $l_d(W, M, C)$ the matrix reconstruction error between W and $M \times C \times M^\top$.

A number of measurements have been proposed to calculate the matrix reconstruction error. One general approach is to measure the norm of the difference between W and the reproduced matrix $MCM^\top$, i.e., $l_d(W, M, C) = \|W - MCM^\top\|$. Two types of matrix norms used in measuring the reconstruction errors are the entry-wise norms (e.g, Frobenius norm) and the induced norms (e.g., spectral norm). The key difference between these two types of norms is that the entry-wise norms measure the error by the entry-wise mismatch, while the induced norm measures the mismatch between two matrices by the difference in their eigenspetra. Here we adopt the Frobenius norm,

$$\begin{aligned} l_d(W, MCM^T) &= ||W - MCM^\top||_F^2 \\ &= \sum_{i,j=1}^{n} (W_{ij} - [MCM^T]_{ij})^2 \end{aligned}$$

4.3.2 Consistency with the Prior Knowledge from GO

Second, we exploit the prior knowledge from GO to regularize the solution of M. We encode the information within GO by a similarity matrix S, where $S_{ij} \geq 0$ represents the similarity between two genes in their biological functions [Jin et al., 2006]. Furthermore we create a normalized combinatorial graph Laplacian L from the similarity matrix S. Then the disagreement between gene modules and collected gene information from GO is measured by $l_m(M, L) = \mathrm{tr}(MLM^\top)$. To better understand this quantity, we expand $l_m(M, L)$ as follows:

$$\mathrm{tr}(MLM^\top) = \sum_{i,j=1}^{N} S_{ij} \left(\sum_z (M_{iz} - M_{jz})^2 \right)$$

Note that term S_{ij} measures the similarity between gene i and j in gene functions described in GO, and term $\sum_z (M_{iz} - M_{jz})^2$ measures the difference in module memberships between two genes. Hence, the product of the two terms essentially indicates the disagreement between M and the gene information from GO. By minimizing this disagreement, we ensure that the gene modules are consistent with the prior information of genes.

4.3.3 Gene Module Network with Hierarchical Scale-free Structure

As revealed by previous studies [Bhan et al., 2002, Jeong et al., 2000], the structure of many gene networks appears to be hierarchical and scale-free. In particular, genes are first clustered into modules, and the gene modules are then connected by scale-free networks. The most important feature of a scale-free network is that most nodes in the network are connected to a few neighbors, and only a small number of nodes, which is often called "hubs", are connected to a large number of nodes. Compared to other network structures, a scale-free network has a very skewed degree distribution, and therefore a relatively smaller number of edges [Barabasi and Albert, 1999]. This fact implies that the network structure matrix C should be a *sparse matrix* of which most elements are small or close to zero. Thus, to ensure a scale-free network structure, we regularize the sparsity of matrix C by term $l_c(C) = \|C\|_F^2$.

4.3.4 Matrix Factorization Framework for Hierarchical Module Network Reconstruction

Combining the measurements for the above three criteria, we have the final formulation for finding the optimal M and C, i.e.,

$$\begin{aligned} \min_{M,C} \quad & l_d(W, M, C) + \alpha l_m(M, L) + \beta l_c(C) \\ \text{s. t.} \quad & C \succeq 0 \\ & C_{ii} = 1,\ i = 1, 2, \ldots, n \\ & C_{ij} \geq 0,\ i, j = 1, 2, \ldots, r \\ & M_{ij} \geq 0,\ i, j = 1, 2, \ldots, n \end{aligned} \tag{4.10}$$

where parameter α and β weight the contribution of terms l_m and l_c respectively. The constraint $C \succeq 0$ ensures that the interaction among modules complies with the triangular inequality, i.e., if module i has strong interactions with both module j and module k, then module j and k are also expected to have strong interactions. Unlike the Bayesian network based structure learning that require solving a discrete optimization problem, (4.10) is an optimization problem of continuous variables and therefore can usually be solved more efficiently than Bayesian network.

4.4 Solving the Constrained Matrix Factorization

We solve the above optimization problem through alternating optimization. It alters the process of optimizing M with fixed C and the process of optimizing C with fixed M iteratively till the solution converges to a local optimum. We describe these two processes as follows:

Optimize M by fixing C: The related optimization problem is:

$$\begin{aligned} \underset{M\in\mathbb{R}^{n\times r}}{\arg\min} \quad & F_m(M) = \|W - MCM^\top\|_F^2 + \alpha \mathrm{tr}(M^\top LM) \\ \text{s. t.} \quad & M_{ij} \geq 0,\ i,j = 1,2,\ldots,n \end{aligned}$$

To find an optimal solution for M, we propose the following bound optimization algorithm. Let $\tilde{M}$ represent the solution of the previous iteration, and our goal is to find a solution of M for the current iteration. First, we consider bounding the first term in $F_m(M)$ by the following expression:

$$\begin{aligned} & \left(W_{i,j} - \sum_{k,l=1}^{r} M_{i,k}M_{j,l}C_{k,l}\right)^2 \\ = & \left(W_{i,j} - \frac{[\tilde{M}C\tilde{M}^\top]_{i,j}}{[\tilde{M}C\tilde{M}^\top]_{i,j}} \sum_{k,l=1}^{r} \frac{\tilde{M}_{i,k}\tilde{M}_{j,l}C_{k,l}}{\tilde{M}_{i,k}\tilde{M}_{j,l}C_{k,l}} M_{i,k}M_{j,l}C_{k,l}\right)^2 \\ \leq & \sum_{k,l=1}^{r} \frac{\tilde{M}_{i,k}\tilde{M}_{j,l}C_{k,l}}{[\tilde{M}C\tilde{M}^\top]_{i,j}} \left(W_{i,j} - [\tilde{M}C\tilde{M}^\top]_{i,j} \frac{M_{i,k}M_{j,l}C_{k,l}}{\tilde{M}_{i,k}\tilde{M}_{j,l}C_{k,l}}\right)^2 \\ \leq & W_{i,j}^2 + \frac{1}{2}\sum_{k,l=1}^{r} [\tilde{M}C\tilde{M}^\top]_{i,j}\tilde{M}_{i,k}\tilde{M}_{j,l}C_{k,l} \left(\left[\frac{M_{i,k}}{\tilde{M}_{i,k}}\right]^4 + \left[\frac{M_{j,l}}{\tilde{M}_{j,l}}\right]^4\right) \\ & -2\sum_{k,l=1}^{r} W_{i,j}\tilde{M}_{i,k}\tilde{M}_{j,l}C_{k,l}(\log M_{i,k} - \log \tilde{M}_{i,k} + \log M_{j,l} - \log \tilde{M}_{j,l} + 1) \end{aligned}$$

We then upper bound the second term in $F_m(M)$, i.e., $S_{i,j}(M_{i,k} - M_{j,k})^2$, such that:

$$\begin{aligned} & S_{ij}(M_{ik} - M_{jk})^2 \\ = & S_{ij}M_{ik}^2 + S_{ij}M_{jk}^2 - 2S_{ij}M_{ik}M_{jk} \\ \leq & S_{ij}M_{ik}^2 + S_{ij}M_{jk}^2 - 2S_{ij}\tilde{M}_{ik}\tilde{M}_{jk}\left(\log\frac{M_{ik}}{\tilde{M}_{ik}} + \log\frac{M_{jk}}{\tilde{M}_{ik}} + 1\right) \end{aligned}$$

Taking the derivative of $F_m(M)$ with respect to $M_{i,k}$, we have

$$\begin{aligned} \frac{\partial F_m(M)}{\partial M_{i,k}} = & 4\left[\frac{M_{i,k}}{\tilde{M}_{i,k}}\right]^3 [\tilde{M}C\tilde{M}^\top\tilde{M}C]_{i,k} - 4\frac{\tilde{M}_{i,k}}{M_{i,k}}[W\tilde{M}C]_{i,k} \\ & +4\alpha M_{i,k}D_i - 4\alpha\frac{\tilde{M}_{i,k}}{M_{i,k}}[S\tilde{M}]_{i,k} \end{aligned}$$

where D_i is the degree of the ith gene and is defined before.

By setting the derivative to be zero, we have

$$\left[\frac{M_{ik}}{\tilde{M}_{ik}}\right]^4 [\tilde{M}C\tilde{M}^\top \tilde{M}C]_{ik} + \alpha \left[\frac{M_{ik}}{\tilde{M}_{ik}}\right]^2 \tilde{M}_{ik} D_i \\ -\alpha[S\tilde{M}]_{ik} - [W\tilde{M}C]_{ik} = 0$$

where $D_i = \sum_{k=1}^{n} S_{ik}$ is the degree of the ith gene. Thus we have the optimal solution for M as following:

$$M_{ik} = \tilde{M}_{ik} \left(\frac{2c_{ik}}{b_{ik} + \sqrt{b_{ik}^2 + 4a_{ik}c_{ik}}} \right)^{\frac{1}{2}} \tag{4.11}$$

where

$$a_{ik} = [\tilde{M}C\tilde{M}^\top \tilde{M}C]_{ik}$$

$$b_{ik} = \alpha \tilde{M}_{ik} D_i$$

$$c_{ik} = \alpha[S\tilde{M}]_{ik} + [W\tilde{M}C]_{ik}$$

Optimize C by fixing M: This corresponds to the following optimization problem:

$$\begin{aligned} \underset{C \in \mathbb{S}^r}{\arg\min} \quad & F_c(C) = \|W - MCM^\top\|_F^2 + \beta \|C\|_F^2 \\ \text{s. t.} \quad & C_{ii} = 1,\ i = 1, 2, \ldots, r \\ & C_{ij} \geq 0,\ i, j = 1, 2, \ldots, r \\ & C \succeq 0 \end{aligned}$$

We expand the objective function $F_c(C)$ as follows:

$$F_c(C) = \mathrm{tr}(WW^\top) - 2\mathrm{tr}(WMCM^\top) \quad + \mathrm{tr}(MCM^\top MCM^\top) + \beta \mathrm{tr}(CC^\top)$$

We then introduce auxiliary variable B and slack variables η, ξ as following

$$B = M^\top MC, \quad \eta \geq \textstyle\sum_{i,j=1}^{r} B_{ij} B_{ji}, \quad \xi \geq \displaystyle\sum_{i,j=1}^{r} C_{ij}^2$$

Finally, the optimization problem becomes:

$$\begin{aligned}
\underset{C\in\mathbb{S}^r}{\arg\min} \quad & \eta + \beta\xi - 2\mathrm{tr}(M^\top WMC) \\
\text{s. t.} \quad & C_{ii} = 1,\ i = 1, 2, \ldots, r \\
& C_{ij} \geq 0,\ i, j = 1, 2, \ldots, r \\
& C \succeq 0 \\
& \eta \geq \sum_{i,j=1}^{r} B_{ij}B_{ji}, \quad B = M^\top MC \\
& \xi \geq \sum_{i,j=1}^{r} C_{ij}^2
\end{aligned} \tag{4.12}$$

This optimization problem can be solved effectively using semi-definite programming technique.

In summary, the iterative optimization algorithm to solve problem 4.10 can be formulated as following:

Step 1 Randomly initialize M subject to the constraints in (4.11)

Step 2 Compute C by (4.12) using the M initialized in step 1

Step 3 Until convergence, do

1. Fix C, update M using Equation (4.11)
2. Fix M, update C using Equation (4.12)

4.4.1 Determining Parameters α and β

The regularizer parameter α and β significantly affect the outcome of the proposed algorithm: α balances the information from GO against the information from gene expression data, and β controls the sparseness of the interaction matrix C. Here we use the *stability analysis* to determine the value of α and β. The basic assumption of stability analysis is that if the parameters are set properly, then then algorithm runs with different random initialization

should result in more or less similar results [Tibshirani et al., 2001]. We run our algorithm multiple times with a given setting of α and β, then evaluate each result to all other results using the evaluation metric defined in the next section, then we calculate the standard deviation of all these evaluation metrics. α and β are tuned to minimize this standard deviation.

4.5 Experimental Results and Discussion

Our experiments are designed to evaluate our proposed knowledge driven matrix factorization framework in reconstructing modular gene regulatory network, particularly in identifying gene modules and uncovering the interactions among gene modules.

4.5.1 Datasets

Two datasets are used in our experiments:

- *Gene expression data of yeast cell cycle system:* The gene expression data for 104 genes involved in yeast cell cycle were obtained from the Yeast Cell Cycle Analysis Project (`http://genome-www.stanford.edu/cellcycle/data/rawdata/`). These genes were divided into six groups based on their peak expression in the different phases of the cell cycle and the transcription factors that regulate them [Spellman et al., 1998].

- *Gene expression data of liver cell system:* Gene expression data was obtained for HepG2 cells exposed to free fatty acids (FFAs) and tumor necrosis factor (TNF-α) [Srivastava and Chan, 2007]. Gene expression data were obtained for 15 different conditions. The original data consisted of 19458 genes. The analysis of variance (ANOVA) was applied to the entire list of genes with $P < 0.01$ to compare the effect of treatment (e.g. FFA or TNF-α) and to determine whether a treatment had a significant effect. The expression levels of 830 genes were found to be significant due to either TNF-α or FFA [Li et al., 2007] and this subset of genes are further analyzed in our experiment.

4.5.2 Evaluation of KMF on Yeast Cell Cycle Data

In this study, we focus on evaluating the capability of KMF in identifying gene modules because the interaction information among gene modules is not available. We compared the gene modules identified by KMF and three other baseline algorithms to the ground truth which includes six groups defined by [Spellman et al., 1998]. The three algorithms used as baseline algorithms in our study to identify the gene modules in the yeast cell cycle genes include 1) Bayesian Module network in Genomica [Segal et al., 2003], 2) Probabilistic Spectral Clustering (PSC) [Jin et al., 2005], and 3) Sparse Matrix Factorization (SMF) [Badea and Tilivea, 2005]. Bayesian Module network in Genomica was chosen since it had used the yeast cell cycle genes to identify gene modules. PSC was chosen since it had been proved to be one of the state-of-the-art clustering algorithms. We evaluated three settings when applying PSC to identify the gene modules. In the first two settings, either the gene expression data or GO is used to construct the similarity matrix before applying PSC. In the third setting, the two similarity matrices based on gene expression and GO are combined linearly. SMF was chosen since it had been shown to identify gene modules using a non-negative factorization algorithm that combines gene expression data and transcription factor binding data. To obtain the cluster membership of the genes, we set a threshold on the membership matrix M. A natural choice for the threshold is $1/r$ where r is the number of clusters. The same method was applied to PSC and SMF to determine the binary cluster memberships.

As shown in Figure 4.1, we can see that KMF is more effective in capturing the structure of the gene clusters compared to the other algorithms. Comparing 4.1(a) and 4.1(b) in Figure 4.1, we can see that KMF captured well clusters 3, 4, 5, 6 and part of cluster 1. In contrast, the other algorithms captured fewer clusters. For example, PSC using both expression and GO captured clusters 3, 4 and 6. Interestingly, Genomica, which is based only on expression data, captured clusters 1, 5 and 6.

To quantitatively evaluate the performance of the algorithms in our experiment, we use

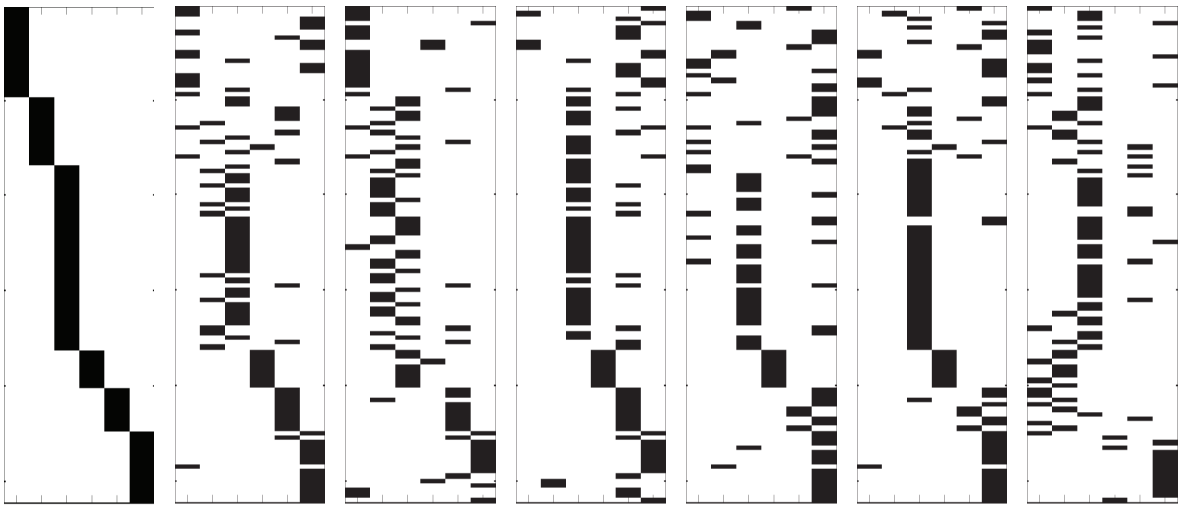


Figure 4.1: Visualization of the clustering results. In each subfigure, the horizontal axis represents the cluster IDs and the vertical axis represents the genes. From (b) to (g), the seven subfigures show the distribution of the genes for each analysis as compared to the manually assigned genes in the original 6 groups in (a) [Spellman et al., 1998]. The clusters are generated by KMF (b), Genomica (c), PSC using the expression data (d), GO (e), and their combination (f), and SMF (g). For better visualization and comparison, each gene was assigned to only one cluster in the figure.

the Pairwise F-measure (PWF1) metric [Liu et al., 2007]. Let U be the set of gene pairs that share at least one cluster in the experiment, and T be the set of gene pairs that actually share at least one cluster, PWF1 is defined as

$$\begin{aligned} \text{precision} &= \frac{|U \cap T|}{|U|} \\ \text{recall} &= \frac{|U \cap T|}{|T|} \\ \text{PWF1} &= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \end{aligned}$$

where $|\cdot|$ is cardinality operator on a set. The precision measures the accuracy in identifying co-regulated genes, and the recall measures the percentage of co-regulated genes that are correctly identified, where we assume that the genes within an original group [Spellman et al., 1998] were co-regulated. PWF1 combines these two factors by their harmonic mean.

Algorithm	PWF1(mean±std)
KMF	**0.625±0.007**
Genomica	0.473
PSC (expression)	0.446±0.057
PSC (GO)	0.386±0.020
PSC (expression+GO)	0.571±0.013
SMF (expression+binding)	0.413±0.118

Table 4.2: PWF1 measure of the experimental results on the Yeast cell cycle dataset. Each algorithm except Genomica (which does not need initialization at the beginning of execution) was executed 10 times with different random initializations, and the mean and standard deviation of PWF1 from 10 runs are calculated.

Table 4.2 shows the PWF1 measure of different algorithms on the Yeast cell cycle data. We can see from the results that KMF outperformed the other algorithms significantly. Note that KMF performed better than PSC using the combination of GO and gene expression data. This suggests that KMF is more effective in exploiting prior knowledge than PSC. In addition, KMF also showed a lower standard deviation on the PWF1 over multiple runs. This suggested that KMF performed robustly by utilizing the GO information to Guinney2007de the modular network reconstruction from gene expression data.

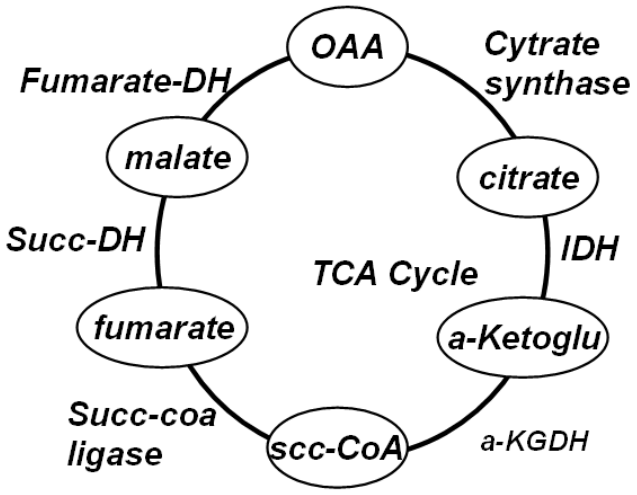
4.5.3 Application to Identifying Gene Modules and Modular network in Liver Cells

We also applied KMF to gene expression data obtained from liver cells where the main objective was to identify the interactions between the modules.

In our experiments we manually set the number of clusters to be 30 according to the suggestions of biologists. We found that for most identified gene modules, genes with similar functions were enriched in their own separate modules/gene groups. For example, 7 out of the 11 genes in a module encode the 5 of the 6 enzymes involved in the TCA cycle, see Figure 4.2. Similarly, one module consisted primarily of NADH dehydrogenases and one

module consisted of the genes involved in the metabolism of ATP. 7 out of the 18 genes in a module encode different sub-complexes of cytochrome-c oxidase (complex IV). In general, most of the gene-groups could be assigned a particular function/process based upon the list of genes enriched in them. We only give a few examples above to illustrate the function enrichment of gene modules. We also note that a common practice in evaluating function enrichment by GO can not be applied here since we already utilize GO for the identification of gene modules and their interaction. The full list of gene clusters is available online at `http://www.chems.msu.edu/groups/chan/genecluster.xls`.

Next, we examined if KMF is able to correctly uncover the interaction among different modules by looking into the C matrix whose coefficients indicate the strengths of interactions, which is analogous to a correlation matrix. After sorting out the significantly higher values in C matrix, we found that module *complex III*, *complex IV*, *complex I*, *complex V* and *TCA and complex II* are closely connected as shown in Figure 4.3. From the aspect of molecular biology, most of the proteins in these modules are located in the mitochondria or on the mitochondria membrane, and these modules are indeed biologically connected. The modules of TCA cycle, electron transport chain (ETC) complex I, complex III, complex IV, and ATP synthase are related to energy production in the mitochondria, which are often referred to as intracellular powerhouse because they produce most of the energy used by the cell. The production of energy in the mitochondria is accomplished by two closely linked metabolic processes, the TCA cycle and oxidative phosphorylation. The TCA cycle converts carbohydrates, lipids, and amino acids into ATP and energy rich molecules, such as NADH. Oxidative phosphorylation generates ATP through the ETC consisting of five protein complexes embedded in the inner membrane of the mitochondria including complex I (NADH dehydrogenase), complex II (succinate dehydrogenase of the TCA cycle), complex III (cytochrome-c reductase), complex IV (cytochrome-c oxidase), and complex V (ATP synthase). Thus, Figure 4.3 show a biological network involved in the production of energy in mitochondria reconstructed by KMF. For many other modules, their interaction is relatively



Module 29
S-adenosylmethionine decarboxylase 1 (AMD1)
isocitrate dehydrogenase 3 (NAD+) alpha (IDH3A)
fumarate hydratase (FH)
ornithine decarboxylase 1 (ODC1)
ornithine decarboxylase antizyme 1 (OAZ1)
S-adenosylmethionine decarboxylase 1 (AMD1)
citrate synthase (CS) (N67639)
citrate synthase (CS) (AA416759)
succinate dehydrogenase complex, subunit C
succinate-CoA ligase, GDP-forming, alpha subunit (SUCLG1)
succinate-CoA ligase, GDP-forming, beta subunit

Figure 4.2: An example of the functional modules identified by KMF. Left: KMF uncovered TCA cycle. 5 out of 6 enzymes involved in TAC cycle were found in module 29. Right: TCA genes were enhanced in Module 29.

weak according C matrix, and therefore their biological meaning is rather unclear from our study.

Therefore, KMF is able to identify highly enriched gene modules with distinct cellular functions and the interactions between the modules. In summary, KMF is an approach that can be applied to uncover pathways specific to a phenotype and potentially be used to elucidate mechanisms involved in diseases by integrating gene expression and *a priori*

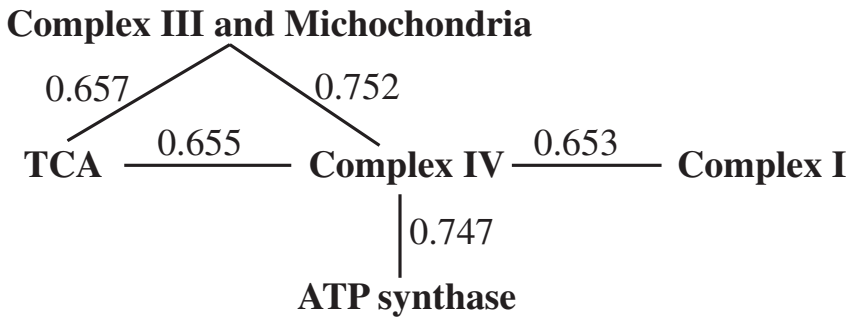


Figure 4.3: KMF uncovered the connections between energy production modules of TCA, Complex I, III, IV and V. Numbers on the edges are the interaction strengths between the modules from the interaction matrix (C matrix).

knowledge.

4.6 Conclusion

A novel framework is presented in this chapter to meet the challenging problem of reconstructing gene networks from multiple information sources. The advantage of our proposed framework is that it derives both the gene modules and their interactions in a unified framework of matrix factorization, and it incorporates the prior knowledge of co-regulation relationships from GO information into the network reconstruction process. We also present an efficient algorithm to solve the related optimization problem. Experiments show that the proposed framework performs significantly better in identifying gene modules than several state-of-the-art algorithms, and the interactions among modules uncovered by our algorithm are proved to be biologically meaningful.

CHAPTER 5

Exclusive Lasso for Multi-task Feature Selection

In this chapter we discuss learning grouped and hierarchical structures using regularization.

Although the structure information often appears explicitly in many situations like the conditional dependency in a Bayesian network which can be visualized using a directed graph, it may be implicit sometimes and thus difficult to explore. For example, a group of variables show co-varying property in group lasso regularization, i.e., they either co-exist or diminish together. Several algorithms including group Lasso [Yuan and Lin, 2005], hierarchical penalization [Szafranski et al., 2007] and Composite Absolute Penalties (CAP) [Zhao et al., 2009] have been proposed to exploit this kind of hidden structures within groups.

These group and hierarchical regularizations are mostly restricted to discover positive correlations among the variables within groups. Specifically, they assume that if a few variables in a group are important, then most of the variables in the same group should also be important. However, in many real-world applications, we may come to the opposite observation, i.e., variables in the same group exhibit negative correlations by competing with each other. For instance, in visual object recognition, the signature visual patterns of different objects tend to be negatively correlated, i.e., visual patterns useful for recognizing one object tent to be useless for other objects.

To address the problem of discovering negative correlations within groups, we propose a new regularization which we call *exclusive lasso*. Different from the existing grouped regularizer, if one feature in a group is given a large weight, the exclusive lasso regularizer

tends to assign small or even zero weights to the other features in the same group. We will present a theoretical analysis to verify the exclusive nature of the proposed regularizer in detail.

5.1 Grouped and Hierarchical Regularization

We briefly review the related work in grouped and hierarchical regularization.

5.1.1 Group Lasso

Group lasso [Yuan and Lin, 2005] has been studied extensively and applied to a number of machine learning problems. It uses the L_1 norm, which is theoretically proven to generate sparse solutions [Tibshirani, 1996], to select groups of variables that are grouped by the L_2 norm. Consider a linear regression setup where we have a continuous response $Y \in \mathbb{R}^n$, an $n \times p$ design matrix X and a parameter vector $\beta \in \mathbb{R}^p$, the group lasso estimator is defined as

$$\hat{\beta} = \arg\min_{\beta} \|Y - X\beta\|_2^2 + \lambda \sum_{g=1}^{G} \|\beta_{\mathcal{I}_g}\|_2$$

where $\mathcal{I}_g$ is the index set belonging to the g-th group of variables, $g = 1, \cdots, G$. The group lasso can be viewed as an intermediate between the L_1 and L_2 penalty. It is invariant under (groupwise) orthogonal transformation like ridge regression.

5.1.2 Hierarchical Penalization

Szafranski et al. [2007] proposed a hierarchical penalization which uses shrinking coefficients to transform the ridge-like penalty into a sparse penalizer. The model parameterized by β is fitted by minimizing a different a differentiable loss function $J(\cdot)$, subject to a ridge penalty

with adaptive coefficients that encourages sparseness among and within groups:

$$\begin{aligned} \min_{\beta,\sigma_1,\sigma_2} \quad & J(\beta) + \lambda \sum_{\ell} \sum_{m \in G_\ell} \frac{\beta_m^2}{\sqrt{\sigma_{1,l}\sigma_{2,m}}} \\ \text{s.t.} \quad & \sum_{\ell} d_\ell \sigma_{1,\ell} = 1, \quad \sigma_{1,\ell} \geq 0, \quad 1 \leq \ell \leq L \\ & \sum_{m} \sigma_{2,m} = 1, \quad \sigma_{2,m} \geq 0, \quad 1 \leq m \leq L \end{aligned} \tag{5.1}$$

The Lagrange parameter λ controls the amount of shrinkage, and d_ℓ is the size of the group ℓ. The constraints encourage sparseness in $\sigma_{1,\ell}$ and $\sigma_{2,m}$, which in turn induces sparseness in β_m.

In Equation 5.2, the groups G_ℓ form a partition of I, and the hierarchy refers to the tree-structure of the shrinking coefficients: $\sigma_{2,m}$ shrinks parameter β_m, while $\sigma_{1,\ell}$ shrinks the parameters for group G_ℓ. The minimizer of the problem equivalent to:

$$\min_{\beta} \quad J(\beta) + \lambda \left(\sum_{\ell} d_\ell^{1/4} \Big(\sum_{m \in G_\ell} |\beta_m|^{4/3} \Big)^{3/4} \right)^2$$

As we can see, this is a special case of the CAP regularizer. The parameter d_ℓ accounts for the groups sizes. The inner $L_{4/3}$ norm and the outer L_1 norm form a mixed-norm penalty that will be denoted $L_{(4/3,1)}$. The overall regularization generates sparse solutions at the group level, with few leading coefficients within the selected groups.

5.1.3 Composite Absolute Penalties

Zhao et al. [2009] further extended the idea of group lasso and proposed a general Composite Absolute Penalties (CAP) family, which allows for (i) different norms for combining variables within the same groups, and (ii) overlapping in variables between groups. Let $\beta = (\beta_1, \cdots, \beta_p)^\top$ be the p variables to be regularized. Given the grouping structure $G = \{G_k \subset \{1, \ldots, p\}, k = 1, \cdots, K\}$, and a vector of norm parameters $\gamma =$

γ_k	Regularizer
$\gamma_k = 1$	Lasso [Tibshirani, 1996]
$\gamma_k = 4/3$	Hierarchical Penalization [Szafranski et al., 2007]
$\gamma_k = 2$	Group Lasso [Yuan and Lin, 2005]
$\gamma_k = \infty$	iCAP penalty [Zhao et al., 2009]

Table 5.1: Several regularizers can be seen as special cases of the CAP family when $\gamma_0 = 1$ and γ_k takes different values.

$(\gamma_0, \gamma_1, \cdots, \gamma_K) \in \mathbb{R}_+^{K+1}$, the regularizer $T_{G,\gamma}(\beta)$ is defined as follows

$$T_{G,\gamma}(\beta) = \sum_k \left(\sum_{m \in G_k} |\beta_m|^{\gamma_k} \right)^{\gamma_0/\gamma_k}$$

Mixed-norms corresponds to groups defined as a partition of the set of variables. A CAP may also rely on nested groups, $G_1 \subset G_2 \subset \cdots \subset G_K$, and $\gamma_0 = 1$, in which case it favors hierarchical selection, i.e., the selection of groups of variables in the predefined order $\{I \setminus G_K\}, \{G_L \setminus G_{K-1}\}, \cdots, \{G_2 \setminus G_1\}, G_1$.

The CAP penalty in Equation 5.2 gives a very general family of regularizations. By reviewing existing grouped and hierarchical regularizations, we find they all consider $\gamma_0 = 1$ to enforce sparseness at the group level and identical norms $\|\gamma_k\|$ at the parameter level, as listed in Table 5.1.

In Table 5.1, the iCAP penalty limits the maximum magnitude of the coefficients within groups.

Figure 5.1 gives a visualization of the admissible sets for various regularizers.

5.1.4 Limitations with Existing Group Regularizers

Although various regularizers have been proposed as reviewed above to infer implicit group and hierarchical structures, these regularizers all assume the covarying property within groups, i.e., a key assumption behind these regularizers is that if a few features in a group are

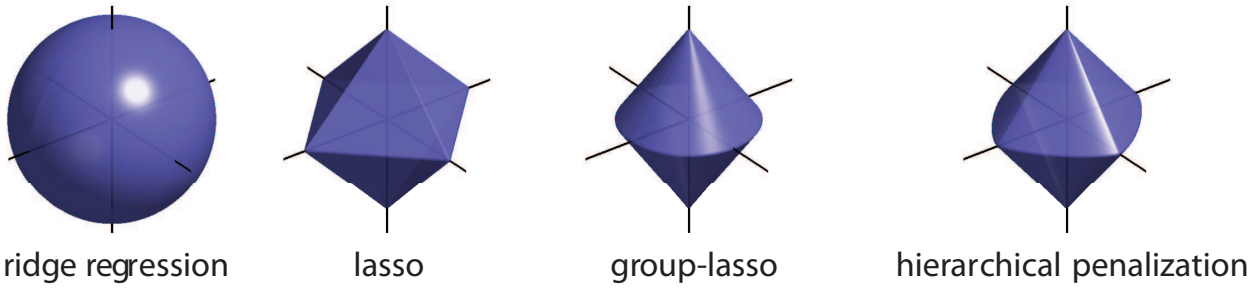


Figure 5.1: Visualization of the admissible sets for various regularizers. Ridge regression: $\beta_1^2 + \beta_2^2 + \beta_3^2 \leq 1$. Lasso: $|\beta_1| + |\beta_2| + |\beta_3| \leq 1$. Group lasso: $2(\beta_1^2 + \beta_2^2)^{\frac{1}{2}} + |\beta_3| \leq 1$. Hierarchical penalization: $2^{\frac{1}{4}}(|\beta_1|^{\frac{4}{3}} + |\beta_2|^{\frac{4}{3}})^{\frac{3}{4}} + |\beta_3| \leq 1$.

important, then most of the features in the same group should also be important. However, in many real-world applications, we may come to the opposite observation. Consider the problem of multi-category document classification. The existing approaches for multi-task feature selection usually assume a positive correlation among the categories, namely, when one keyword is important for several categories, it is also expected to be important for the other categories. This positive correlation is usually captured by a group lasso regularizer, where a group is defined for every word w that includes the feature weights of all the categories for word w. However, when our objective is to differentiate the related categories, we may expect a negative correlation among categories, namely, if word w is deemed to be important for one category, it becomes less likely for w to be an important word for the other categories. This kind of negative correlations appear in many real world applications. Here we give two more examples:

- In visual object recognition, the signature visual patterns of different objects tend to negatively correlated.
- In bioinformatics, many cellular processes share different signal pathways, leading to negatively correlated features (i.e., genes).

It is clear that such a negative correlation structure violates the underlying modeling of the existing group regularizers.

5.2 Exclusive Lasso

In order to capture the implicit structure of negative correlation among variables, we propose the exclusive lasso regularizer. Different from the grouped regularizer reviewed above, if one feature in a group is given a large weight, the exclusive lasso regularizer tends to assign small or even zero weights to the other features in the same group. We present a simple analysis to verify the exclusive nature of the proposed regularizer. Based on the proposed exclusive lasso regularizer, we present a framework for kernel-based multi-task learning. An efficient algorithm is derived to solve the related optimization problem. Empirical studies with document categorization verify that the proposed regularizer is effective for multi-task feature selection.

Although exclusive lasso proposed in our work can also be viewed as a special case of mixed norm and the general CAP family, this study is distinguished from the existing ones in two aspects:

1. Unlike the previous studies that only emphasize the sparsity of solutions caused by the regularization, our in depth analysis also reveals that the exclusive lasso is able to introduce competitions among variables within the same group, which is a key property for capturing the negative correlation among the tasks;

2. We apply the exclusive lasso regularization method to kernel based multi-task learning problem. It results in a sophisticated min-max optimization problem that is beyond the capability of the existing algorithms for group regularization. We present an efficient algorithm for solving the related min-max optimization based on the subgradient descent method.

Notations: We use index i for instances, j for features, and k for tasks. We use n to denote the total number of instances, d for the number of features, and m for the number of tasks.

5.2.1 Multi-task Feature Selection

We introduce our exclusive lasso regularizer in the context of multi-task learning (MTL) [Caruana, 1997]. Multi-task Learning has proven to be useful both theoretically [Baxter, 2000, Ben-David and Schuller, 2003] and experimentally [Evgeniou et al., 2005, Jebara, 2004, Torralba et al., 2004]. Most MTL algorithms assume a positive correlation among tasks. For example, Evgeniou et al. [2005], Bakker and Heskes [2003] assume that functions for different tasks are similar to each other, and Baxter [2000], Ben-David and Schuller [2003], Caruana [1997] assume a common representation of data that is shared by all the tasks.

Guyon and Elisseeff [2003] classify feature selection methods into three types: *filter*, *wrapper* and *embedded*. Filters select subsets of variables as a pre-processing step, independently of the chosen predictor; they are cheap but not very effective. Wrappers utilize the learning machine of interest as a black box to score subsets of variable according to their predictive power; they are good but very expensive. Embedded methods perform variable selection in the process of training and are usually specific to given learning machines.

Many algorithms have been proposed for multi-task feature selection, an important problem in multi-task learning. Xiong et al. [2007] imposed an automatic relevance determination prior on the hypothesis classes associated with individual tasks and regularized the variance of the hypothesis parameters. Argyriou et al. [2006] and Obozinski et al. [2006] used the $L_{1,2}$ norm, similar to group lasso, for regularizing features of different tasks. It encourages multiple predicators to have similar parameter sparsity patterns. Jebara [2004] introduced a common vector of binary feature selection switches for all the tasks, and exploited the maximum entropy discrimination formulation to identify the most informative features as well as the discriminant functions. Lee et al. [2007] introduced meta-features for feature selection in related tasks. Guinney et al. [2007] utilized the gradient of the multiple related regression functions over the tasks for dimension reduction and inference of dependencies both across

tasks and specifically for each task. All the existing algorithms for multi-task feature selection assume a positive correlation among the tasks, and they aim to learn a common subset of features for all the tasks. In contrast, our proposed exclusive lasso regularizer assumes a negative correlation among tasks, and aims to introduce competition among variables within the same group.

5.2.2 Multiple Kernel Learning

The proposed exclusive lasso regularizer will be also be used in the multiple kernel learning [Lanckriet et al., 2004] setting, which has been studied extensively in recent years [Lanckriet et al., 2004, Bach et al., 2004, Sonnenburg et al., 2006, Rakotomamonjy et al., 2007]. The key idea of multiple kernel learning is to search for the best combination of multiple kernel functions that will result in the optimal classification performance. A number of algorithms have been developed for multiple kernel learning, including the SDP formulation [Lanckriet et al., 2004], the QCQP formulation [Bach et al., 2004], the semi-infinite linear programming approach [Sonnenburg et al., 2006], the subgradient approach [Rakotomamonjy et al., 2007], and the level method [Xu et al., 2008]. In this paper, we apply the exclusive lasso to the kernel based multiple task learning, in which the combination of multiple kernel functions is learned simultaneously for all the tasks. We derive an efficient gradient-based algorithm to solve the large-scale optimization problem related to the multi-task multiple kernel learning problem.

5.2.3 Multi-Task Learning with Linear Classifiers

We consider a multi-task classification problem. Let $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, n\}$ be the training data, where $x_i \in \mathbb{R}^d$ is the input pattern and $y_i = (y_i^1, \ldots, y_i^m) \in \{-1, +1\}^m$ is the assigned categories with $y_i^k = 1$ if x_i is assigned to category k and $y_i^k = -1$ otherwise. For simplicity, we assume a linear classifier $f_k(x) = \beta_k^\top x$ where $\beta_k = (\beta_k^1, \ldots, \beta_k^d) \in \mathbb{R}^d$ is the combination weights. We thus have the following optimization problem for multi-task

learning:

$$\min_{\beta} \quad \frac{1}{2}V(\beta) + C\sum_{i=1}^{n}\sum_{k=1}^{m}\ell(y_i^k, f_k(x_i)) \tag{5.2}$$

where $\ell(z)$ is a loss function that measures the mismatch between y_i^k and the predicted value $f_k(x_i)$. $V(\beta)$ is a regularizer that controls the complexity of combination weights β. We assume a competitive nature among the features shared by all the tasks, i.e., if a very large weight is assigned to the jth feature for one task, we expect the weights for the same feature to be small or even zero for the other tasks. To this end, we introduce the following regularizer:

$$V(\beta) = \sum_{j=1}^{d}\left(\sum_{k=1}^{m}\left|\beta_k^j\right|\right)^2 \tag{5.3}$$

As indicated in the above expression, we introduce an L_1 norm to combine the weights for the same feature used by different tasks and an L_2 norm to combine the weights of different features together. Since L_1 norm tends to achieve a sparse solution, the construction in $V(\beta)$ essentially introduces a competition among different tasks for the same feature. We refer to the above regularizer as **exclusive lasso**. Using the exclusive lasso as a regularizer, we have the overall optimization problem written as

$$\min_{\beta} \ \frac{1}{2}\sum_{j=1}^{d}\Big(\sum_{k=1}^{m}\left|\beta_k^j\right|\Big)^2 + C\sum_{i=1}^{n}\sum_{k=1}^{m}\ell\left(y_i^k, f_k(x_i)\right) \tag{5.4}$$

An alternative approach to the regularizer shown above is to introduce a constraint for β:

$$\min_{\beta}\left\{\sum_{i=1}^{n}\sum_{k=1}^{m} l\left(y_i^k, f_k(x_i)\right) : \sqrt{\sum_{j=1}^{d}\left(\sum_{k=1}^{m}\left|\beta_k^j\right|\right)^2} \le \gamma\right\} \tag{5.5}$$

where γ is a predefined constant.

5.2.4 Understanding the Exclusive Lasso Regularizer

One of the fundamental questions is how the exclusive lasso regularizer introduces the competition among different tasks for the same feature. To illustrate this point, we consider the

following projection problem,

$$\min_{\beta \in \mathcal{G}} |\beta - \bar{\beta}|_2^2 \tag{5.6}$$

where $\bar{\beta}$ is an existing solution and domain $\mathcal{G}$ is defined as

$$\mathcal{G} = \Bigg\{ \beta = (\beta_1; \ldots; \beta_m) : \beta_k = (\beta_k^1, \ldots, \beta_k^d) \in \mathbb{R}^d, \\ k = 1, \ldots, m, \sqrt{\sum_{j=1}^{d} \left(\sum_{k=1}^{m} \left| \beta_k^j \right| \right)^2} \leq \gamma \Bigg\} \tag{5.7}$$

The projection problem in (5.6) directly demonstrates how the domain $\mathcal{G}$ shapes a solution $\bar{\beta}$, which essentially illustrates the effect of the exclusive lasso regularizer. Projection is an important operation that is used by many optimization algorithms (e.g., subgradient descent). In addition, problems such as constrained least square regression can be cast into a projection problem.

We first convert Equation 5.6 into a convex-concave problem,

$$\min_{\beta} \max_{\lambda \geq 0} |\beta - \bar{\beta}|_2^2 + 2\lambda \left(\sqrt{\sum_{j=1}^{d} \left(\sum_{k=1}^{m} \left| \beta_k^j \right| \right)^2} - \gamma \right) \tag{5.8}$$

The following proposition allows us to simplify the problem in Equation 5.8.

Proposition 1.

$$\sqrt{\sum_{j=1}^{d} \left(\sum_{k=1}^{m} \left| \beta_k^j \right| \right)^2} = \max_{\alpha \in \Delta} \alpha^\top \beta \tag{5.9}$$

where domain Δ *is defined as*

$$\Delta = \Bigg\{ \alpha = (\alpha_1; \ldots; \alpha_m) : \alpha_k = (\alpha_k^1, \ldots, \alpha_k^d) \in \mathbb{R}^d, \\ k = 1, \ldots, m, \sum_{j=1}^{d} \max_{1 \leq k \leq m} |\alpha_k^j| \leq 1 \Bigg\} \tag{5.10}$$

Using the above proposition, we have the following lemma that simplifies Equation 5.8.

Lemma 5.2.1. *Problem 5.8 is equivalent to the following optimization problem*

$$\min_{\tau} \left\{ 2\gamma|\tau|_2 + \sum_{j=1}^{d}\sum_{k=1}^{m}[|\bar{\beta}_k^j| - \tau_j]_+^2 \right\} \tag{5.11}$$

where $[x]_+ = \max(x, 0)$. *The optimal solution of* β *is computed as*

$$\beta_k^j = \ [\bar{\beta}_k^j - \tau_j]_+, \quad j = 1, \ldots, d, \quad k = 1, \ldots, m$$

Proof. Using the Proposition 1, we rewrite Equation 5.8 as

$$\max_{\alpha\in\Delta} \min_{\beta} |\beta - \bar{\beta}|_2^2 + 2\lambda \left(\alpha^\top \beta - \gamma\right)$$

Taking the minimization over β, we have

$$\max_{\alpha} \left\{ -2\lambda\gamma - |\bar{\beta} - \lambda\alpha|_2^2 : \sum_{j=1}^{d} \max_{1\le k\le m} [\alpha_k^j]^2 \le 1 \right\}$$

with $\beta = \bar{\beta} - \lambda\alpha$. Define $\tau_j = \max_{1\le k\le m} \lambda|\alpha_k^j|$ and the above equation can be written as

$$\min_{\tau,\lambda} \left\{ 2\lambda\gamma + \sum_{j=1}^{d}\sum_{k=1}^{m}[|\bar{\beta}_k^j| - \tau_j]_+^2 : |\tau|_2 \le \lambda \right\}$$

or

$$\min_{\tau} \left\{ 2\gamma|\tau|_2 + \sum_{j=1}^{d}\sum_{k=1}^{m}[|\bar{\beta}_k^j| - \tau_j]_+^2 \right\}$$

□

As indicated by the above lemma, whenever $\bar{\beta}_k^j$ is smaller than threshold τ_j, we have β_k^j become zero. The following proposition shows a necessary condition for $\beta_k^j = 0$.

Proposition 2. *For any feature* j, *if* $\sum_{k=1}^{m} \left(|\bar{\beta}_k^j| - \min_{1\le k\le m} |\bar{\beta}_k^j| \right) > \gamma$, *we have* $\beta_k^j = 0$ *if* $\bar{\beta}_k^j \le \left(\sum_{k=1}^{m} |\bar{\beta}_k^j| - \gamma \right) / m$.

Proof. We consider the first order optimality condition for τ, i.e.,

$$\gamma \frac{|\tau_j|}{|\tau|_2} + \sum_{k=1}^{m} [|\bar{\beta}_k^j| - \tau_j]_+ \partial_{\tau_j} [|\bar{\beta}_k^j| - \tau_j]_+ = 0$$

where $\partial_x f(x)$ is the subgradient of function $f(x)$. Notice that $\partial_{\tau_j}[|\bar{\beta}_k^j| - \tau_j]_+ \in [-1, 0]$, and is -1 when $|\bar{\beta}_k^j| < \tau_j$. Hence, the above optimality condition implies that

$$\sum_{k=1}^{m}[|\bar{\beta}_k^j| - \tau_j]_+ \leq \gamma \frac{|\tau_j|}{|\tau|_2} \leq \gamma$$

Since $[|\bar{\beta}_k^j| - \tau_j]_+ \geq |\bar{\beta}_k^j| - \tau_j$, we have

$$\sum_{k=1}^{m}|\bar{\beta}_k^j| - \tau_j \leq \gamma,$$

which leads to the result in the proposition. □

As indicated in the above proposition, when some tasks take significantly smaller weights for feature j than the other tasks, the regularizer will enforce the weights of feature j to be zero for these tasks, leading to the competition of feature j among tasks. Parameter γ is used to control the degree of domination. A large γ requires a large gap among the weights for the same feature before the small weights can be reduced to zero; similarly, a small γ allows us to reduce small weights to zero even when the gap among the weights for the same feature is still small.

5.2.5 Multi-Task Learning with Kernel Classifiers

We extend the exclusive lasso discussed above to the kernel case. In particular, we consider there are d kernels at our disposal, denoted by $\mathcal{W} = \{W^j \in \mathbb{R}^{n\times n}, j = 1, \ldots, d\}$. We assume that each kernel matrix in $\mathcal{W}$ is appropriately normalized (e.g., $\text{tr}(W^j) = 1$). For each task k, we assume that its kernel matrix, denoted by K^k, is a linear combination of the kernel matrices in $\mathcal{W}$, i.e.,

$$K^k = \sum_{j=1}^{d}\lambda_k^j W^j$$

where $\lambda_k = (\lambda_k^1, \ldots, \lambda_k^d) \in \mathbb{R}_+^d$ is the combination weights. For each individual task, the learning of combination weights λ_k, often referred to as multiple kernel learning, is cast into

the following optimization problem

$$\min_{\lambda_k \in \mathbb{R}^d_+} \max_{\gamma_k \in [0,C]^n} \left\{ \gamma_k^\top \mathbf{1} - \frac{1}{2}(\gamma_k \circ z_k)^\top \left(\sum_{j=1}^{d} W^j \lambda_k^j \right) (\gamma_k \circ z_k) \right\} \tag{5.12}$$

where $z_k = (y_1^k, y_2^k, \ldots, y_n^k)$ and $\circ$ is the element-wise dot product. Similar to the linear case, by assuming the exclusive nature among tasks in competing for kernels in $\mathcal{W}$, we introduce the exclusive lasso for regularizing the kernel weights $\lambda = (\lambda_1; \ldots; \lambda_m)$ assigned to different tasks, leading to the following optimization problem:

$$\begin{aligned} \min_{\lambda_k \in \mathbb{R}^d_+} \max_{\gamma_k \in [0,C]^n} \quad & \sum_{k=1}^{m} \left(\gamma_k^\top \mathbf{1} - \frac{1}{2}(\gamma_k \circ z_k)^\top \left[\sum_{j=1}^{d} W^j \lambda_k^j \right] (\gamma_k \circ z_k) \right) \\ & + \frac{r}{2} \sum_{j=1}^{d} \left(\sum_{k=1}^{m} \lambda_k^j \right)^2 \end{aligned} \tag{5.13}$$

where r is a predefined parameter that weights the importance of the regularizer. The following theorem shows the sparsity in the solution of λ and the competition among tasks for kernels caused by the exclusive lasso regularizer.

Theorem 1. *Provided the solution γ, for each kernel W^j, we have $\lambda_k^j > 0$ only if*

$$k = \arg\max_{1 \le k' \le m} (\gamma_{k'} \circ z_{k'})^\top W^j (\gamma_{k'} \circ z_{k'})$$

This theorem follows directly from the result in Proposition 4, which will be stated later.

5.2.6 Algorithm

We focus on solving the problem in (5.13). A straightforward approach is the subgradient method. Define

$$\begin{aligned} g(\gamma, \lambda) &= \sum_{k=1}^{m} \left(\gamma_k^\top \mathbf{1} - \frac{1}{2}(\gamma_k \circ z_k)^\top \left[\sum_{j=1}^{d} W^j \lambda_k^j \right] (\gamma_k \circ z_k) \right) \\ &\quad + \frac{r}{2} \sum_{j=1}^{d} \left(\sum_{k=1}^{m} \lambda_k^j \right)^2 \end{aligned} \tag{5.14}$$

$$f(\gamma) = \min_{\lambda_k \in \mathbb{R}^d_+} g(\gamma, \lambda) \tag{5.15}$$

Hence, the problem in (5.13) can be viewed as a maximization problem

$$\gamma = \underset{\gamma_k \in [0,C]^n}{\arg\max}\ f(\gamma)$$

We thus can apply the subgradient ascent approach to directly maximizing $f(\gamma)$. In each iteration of the subgradient ascent method, we compute the gradient of $f(\gamma)$, denoted by $\nabla f(\gamma)$, and the new solution is obtained by moving the existing solution γ along the direction of $\nabla f(\gamma)$, i.e.,

$$\gamma \leftarrow \pi_G\left(\gamma + s\nabla f(\gamma)\right)$$

where

$$G = \{\gamma = (\gamma_1; \ldots; \gamma_m) \in \mathbb{R}^{mn} : \gamma_k \in [0, C]^n, k = 1, \ldots, m\}$$

and $\pi_G(x)$ projects solution x onto the domain G. Evidently, there are two key parameters that need to be computed efficiently, i.e., step size s and $\nabla f(\gamma)$. The following proposition allows us to compute $\nabla f(\gamma)$, similar to [Xu et al., 2008].

Proposition 3. *We have the gradient of $f(\gamma)$ computed as*

$$\nabla_{\gamma_k} f(\gamma) = \mathbf{1} - \left[\sum_{j=1}^{d} \lambda_k^j \left(W^j \circ z_k z_k^\top\right)\right] \gamma_k \tag{5.16}$$

where λ_k^j is the minimizer of $g(\gamma, \lambda)$, i.e., $\lambda = \underset{\lambda_k \in \mathbb{R}_+^d}{\min}\ g(\gamma, \lambda)$.

As indicated in the above proposition, to compute the gradient of $f(\gamma)$, it is important to efficiently compute λ that minimizes $g(\gamma, \lambda)$. To this end, we rewrite $g(\gamma, \lambda)$ to highlight its dependency on λ:

$$g(\gamma, \lambda) = a - \sum_{k=1}^{m}\sum_{j=1}^{d} b_k^j \lambda_k^j + \frac{r}{2}\sum_{j=1}^{d}\left(\sum_{k=1}^{m} \lambda_k^j\right)^2 \tag{5.17}$$

where

$$a = \sum_{k=1}^{m} \gamma_k^\top \mathbf{1}, \quad b_k^j = \frac{1}{2}(\gamma_k \circ z_k)^\top W^j (\gamma_k \circ z_k) \tag{5.18}$$

In order to minimize $g(\gamma, \lambda)$ with respect to λ, we define h_j as

$$h_j = -\sum_{k=1}^{m} b_k^j \lambda_k^j + \frac{r}{2}\left(\sum_{k=1}^{m} \lambda_k^j\right)^2 \tag{5.19}$$

Since $g(\gamma, \lambda) = a + \sum_{j=1}^{d} h_j$ and each h_j only involves variables $\lambda_k^j, k = 1, \ldots, m$, we could optimize h_i separately. The following proposition gives the optimal solution that minimizes h_j.

Proposition 4. *Assume $b_k^j \neq b_{k'}^j$ for any $k \neq k'$ and any j. The optimal $\lambda_k^j, k = 1, \ldots, m$ that minimizes h_j is*

$$\lambda_k^j = \begin{cases} \bar{\lambda}^j & k = \arg\max\limits_{1 \leq k' \leq m} b_{k'}^j \\ 0 & \textit{otherwise} \end{cases}$$

where $\bar{\lambda}^j$ is computed as $\bar{\lambda}^j = \frac{1}{r} \max\limits_{1 \leq k \leq m} b_k^j$.

Proof. For the sake of simplicity, we drop index j and consider a general problem as follows

$$\min_{\lambda \in \mathbf{R}_+^m} \quad -\sum_{k=1}^{m} b_k \lambda_k + \frac{r}{2}\left(\sum_{k=1}^{m} \lambda_k\right)^2$$

We define $\lambda_k = \eta_k + \bar{\lambda}$ and $\bar{\lambda} = \sum_{k=1}^{m} \lambda_k / m$. We therefore have $\eta_k \geq \bar{\lambda}$ and $\sum_{k=1}^{m} \eta_k = 0$. Thus the original problem can be transformed into a problem of $\bar{\lambda}$ and η, i.e.,

$$\begin{aligned} \min_{\bar{\lambda}, \eta} \quad & \frac{rm^2}{2}\bar{\lambda}^2 - \sum_{k=1}^{m} b_k \eta_k - \bar{\lambda} \sum_{k=1}^{m} b_k \\ \text{s. t.} \quad & \bar{\lambda} \geq 0, \ \sum_{k=1}^{m} \eta_k = 0, \quad \eta_k \geq -\bar{\lambda}, \ k = 1, \ldots, m \end{aligned}$$

We consider the solution for η when $\bar{\lambda}$ is fixed, which leads to the following linear programming problem:

$$\begin{aligned} \min_{\eta} \quad & -\sum_{k=1}^{m} b_k \eta_k \\ \text{s. t.} \quad & \sum_{k=1}^{m} \eta_k = 0, \quad \eta_k \geq -\bar{\lambda}, \ k = 1, \ldots, m \end{aligned}$$

Since $b_k \geq 0$, it is clear that the optimal solution for the above linear programming problem is

$$\eta_k = \begin{cases} (m-1)\bar{\lambda} & k = \arg\max_{1 \leq k' \leq m} b_{k'} \\ -\bar{\lambda} & \text{otherwise} \end{cases}$$

Using the solution for η, we have the following problem for $\bar{\lambda}$

$$\min_{\bar{\lambda} \geq 0} \quad \frac{rm^2}{2}\bar{\lambda}^2 - m\bar{\lambda} \max_{1 \leq k \leq m} b_k$$

It is obvious that $\bar{\lambda} = \max_{1 \leq k \leq m} b_k/(rm)$. □

Note that Proposition 4 only addresses the situation when there is a unique element for $k = \arg\max_{1 \leq k' \leq m} b^j_{k'}$. Similar results can be easily derived when multiple elements tie for the maximum value of b^j_k. This proposition clearly demonstrates the competition of kernels among tasks resulting from the exclusive lasso regularizer. Using the result from Proposition 4, we can efficiently compute the optimal λ for a given γ, which allows us to efficiently compute the gradient of $f(\gamma)$ in (5.16).

We determine the step size s by the backtracking line search [Boyd and Vandenberghe, 2004]. Finally, the duality gap is used to check the convergence. Given the solution λ^* and γ^*, the duality gap is defined as

$$\delta = \min_{\lambda_k \in \mathbb{R}^d_+} g(\gamma^*, \lambda) - \max_{\gamma_k \in [0,C]^n} g(\gamma, \lambda^*), \tag{5.20}$$

where $\min_{\lambda \in \mathbb{R}^d_+} g(\gamma^*, \lambda)$ can be computed efficiently using Proposition 4, and $\max_{\gamma_k \in [0,C]^n} g(\gamma, \lambda^*)$ is solved by a kernel SVM.

5.3 Experiments

We evaluate the efficacy of the proposed exclusive lasso regularizer by multi-task feature selection. We use the Yahoo dataset [Ueda and Saito, 2003] in our experiments. This multi-topic web page categorization dataset was collected from 11 top-level categories ("Arts",

Dataset	m	N	MaxNPI	MinNPI
Arts	19	7441	1838	104
Business	17	11182	9723	110
Computers	23	12371	6559	108
Education	14	11817	3738	127
Entertainment	14	12691	3687	221
Health	14	9109	4703	114
Recreation	18	12797	2534	169
Reference	15	7929	3782	156
Science	22	6345	1548	102
Social	21	11914	5148	104
Society	21	14507	7193	113

Table 5.2: Metadata of the Yahoo data collection.

"Business", "Computers", etc.) in the "yahoo.com" domain. Each top-level category is further divided into a number of second-level subcategories. Each subcategory is an individual task in our multi-task classification algorithm. We preprocessed the datasets by removing topics with less than 100 documents and documents with no topics. 300 keywords are selected for each dataset based on their term frequency. After preprocessing, the number of subcategories ranges from 14 to 23 for the 11 datasets, and the number of data samples ranges from 6345 to 14507. Detailed statistics of the datasets can be found in Table 5.2. By constructing a kernel for each individual keyword, we apply the proposed method for kernel based multi-task task learning to document categorization. Throughout this study, a linear kernel is used by all the methods and for all the experiments because it is proven to be effective for document categorization.

5.3.1 Evaluation

We use the following two algorithms as baselines in our experiments to compare with the proposed exclusive lasso algorithm:

- SVM feature selection [Bradley and Mangasarian, 1998]. We train a linear SVM classifier for each category and select the features that have the largest absolute values in their coefficients. We use this feature selection method instead of the L_1 regularization path [Zhu et al., 2003] because we want to be consistent with the approach used in our proposed algorithm. Note that the SVM classifiers are trained independently in this case, and therefore features are selected independently for each task.

- Multi-task Feature Learning (MTFL) [Argyriou et al., 2006]. MTFL assumed that the functions f_t for T all share a small set of features. It used the group lasso to jointly penalize the features used by different tasks. It encourages multiple predicators to have similar parameter sparsity patterns, and aims to learn a subset of features common to all the tasks. Formally, MTFL aims to solve the following convex optimization problem in the feature selection setting:

$$\min_{A \in \mathbb{R}^{d \times t}} \quad \sum_{t=1}^{T} \sum_{i=1}^{m} L(y_{ti}, \langle a_t, x_{ti} \rangle) + \gamma \|A\|_{2,1}^2$$

  where m is the number of features and $\|A\|_{2,1}$ is the $(2,1)$-norm of matrix A. It is obtained by first computing the 2-norm of the (across the tasks) rows a^i (corresponding to feature i) of matrix A and then the 1-norm of the vector $b(A) = (\|a^1\|_2, \cdots, \|a^d\|_2)$. This norm combines the tasks and ensures that common features will be selected across them. We use the hinge loss function for $L(\cdot)$ for the MTFL algorithm in our experiments because our work follows directly the SVM framework.

To evaluate the efficacy of feature selection, we randomly sample 10 examples from each subcategory for training and use the remaining documents for testing. We use a small number of training examples because it is well known that in document categorization, with sufficient numbers of training documents, any feature selection method works well. After training the classification models, we choose the top features for each subcategory that have the largest weights. An SVM classifier is constructed for each subcategory by using the selected features, and its classification accuracy computed over the test documents is used

to evaluate the efficacy of feature selection algorithms. The hypothesis is that the more effective the feature selection algorithm is, the more accurate the SVM classifier will be. The area under the receiver operating characteristic curve (AUC) is used in our study as performance metric. We vary the number of selected features from one to ten, and repeat each experiment ten times. The reported AUC for each dataset is averaged over ten random trials.

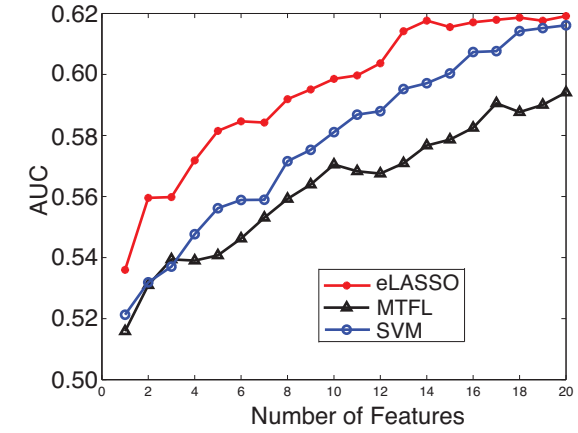
The regularization constant C of SVM is set to be 10 for all the SVM classifiers in the experiments according to our experience. The regularizer parameter r in Equation (5.13) is set to be 1 in all the experiments. Note that we did not employ cross validation to determine the parameters because of the small number of training samples.

5.3.2 Results

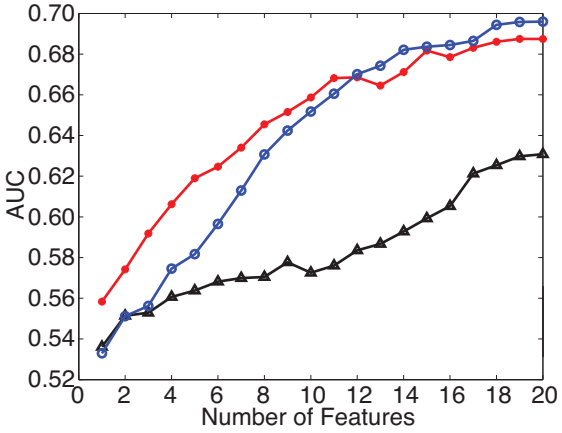
Figure 5.2 shows the average AUC of the 11 datasets of the Yahoo data collection for the three feature selection methods in comparison. We observe that (i) the MTFL method performs noticeably worse than the simple SVM based feature selection method, and (ii) the proposed algorithm for multi-task feature selection outperforms the other two baseline algorithms. This is not surprising given the topic structure in the Yahoo data collection. Although documents within each dataset belong to a common topic and therefore are expected to share many common terms, our goal is to classify documents in each dataset further into subcategories. As a result, we need to select discriminative terms that are sufficient to differentiate the subcategories, not the terms that are commonly shared among subcategories. These discriminative terms are more likely to be discovered by the proposed exclusive lasso algorithm since a discriminative term for a given subcategory is unlikely to be also discriminative for another subcategory. Finally, we observed that the advantage of the proposed algorithm over the other comparative methods tends to diminish as the selected number of features is increased. This is indeed within our expectation, as any feature selection method will work well if we aim to select most of the features.

5.4 Conclusion

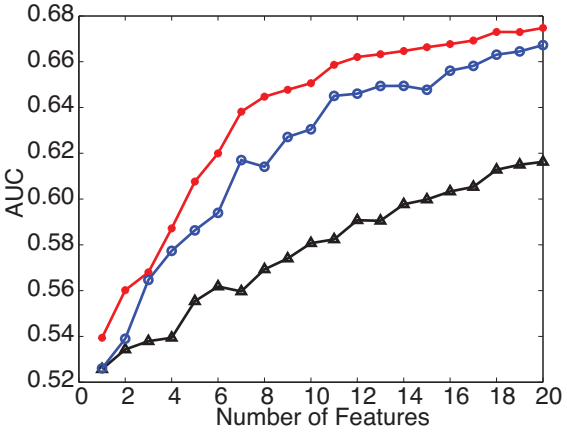
We introduce a new regularization which we call exclusive lasso in this chapter. We give detailed theoretical analysis to illustrate that the proposed exclusive lasso regularizer is able to introduce competitions among variables and thus generate sparse solutions. This regularizer is applied to a multi-task feature selection setting and an efficient algorithm is derived to solve the related optimization problem. Empirical study shows that our proposed algorithm outperforms the baseline algorithms on benchmark datasets.



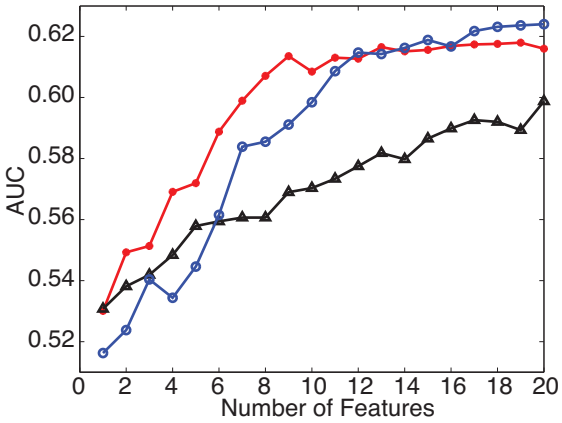
(a) Arts



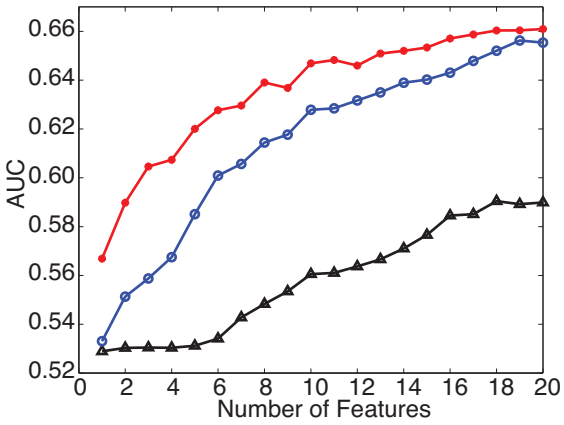
(b) Business



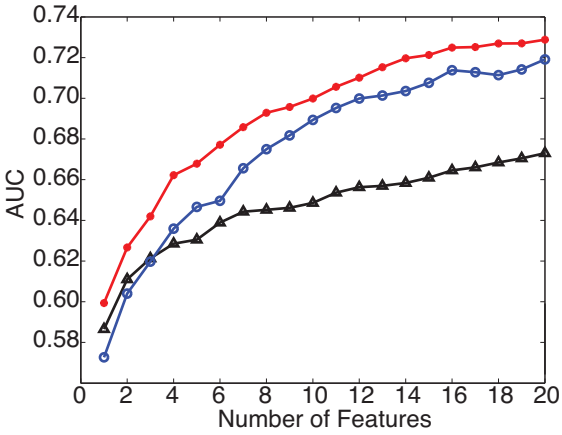
(c) Computers



(d) Education

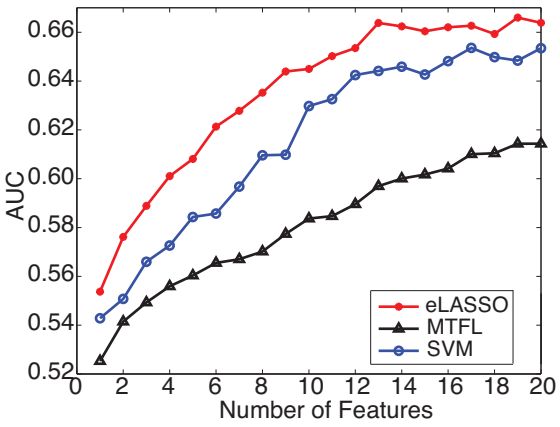


(e) Entertainment

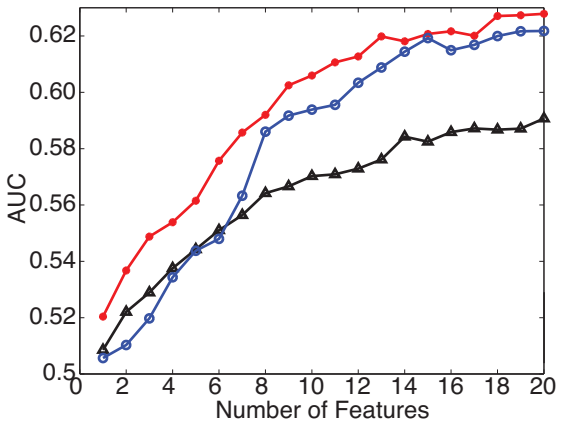


(f) Health

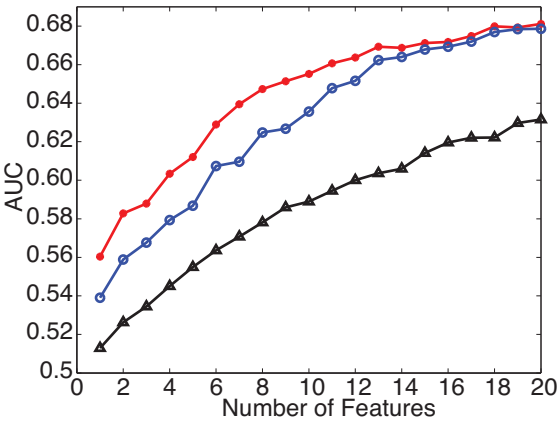
Figure 5.2



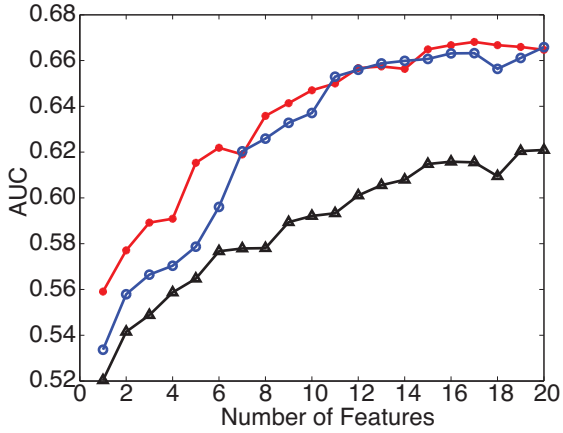
(g) Recreation



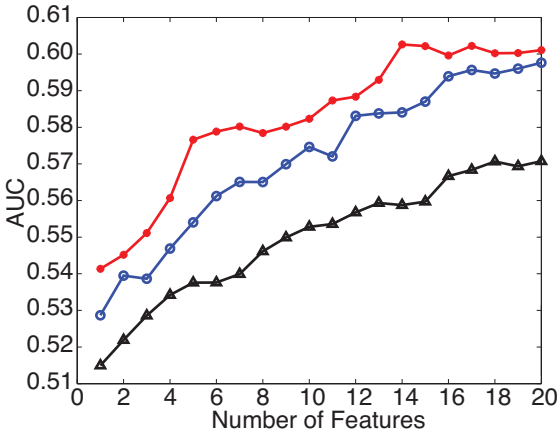
(h) Reference



(i) Science



(j) Social



(k) Society

Figure 5.2: (cont'd) AUC of exclusive lasso (eLASSO), SVM feature selection and MTFL on the 11 datasets of Yahoo data collection. The x axis is the number of selected features from each category used in the testing phase, and the y axis is the corresponding AUC measure. All performances are averaged for 10 runs each with a random sampling of training instances.

CHAPTER 6

Sparse Online Feature Selection

In this chapter we further investigate the sparse structures, specifically in a problem setting that we call *online feature selection*. Most online learning studies assume that the learning has full access to all input features. However, in many real world applications, it is expensive, either computationally or money wise to acquire and use all the input attributes. In this case it is desirable to develop online learning algorithms that only need to sense a small number of attributes before the reliable decision can be made. We develop theories and algorithms for sparse online feature selection. Specifically, we design general algorithms for online feature selection, and examine their theoretic properties such as upper and lower bounds for the regret bound. We evaluate the proposed algorithms by experiments on benchmark datasets.

6.1 Introduction

Unlike batch mode learning where the goal is to learn a single statistical model with small generalization error, online learning focuses on making sequential decisions that minimize the overall regret/loss. Specifically, online learning is performed in a sequence of rounds. In each round t, the algorithm observes an instance $\mathbf{x}_t$ which is drawn from some predefined instance domain $\mathcal{X}$. The algorithm then predicts the binary label of the instance and then receives the true label. The algorithm may use the new example $(\mathbf{x}_t, y_t)$ to improve its prediction model for future rounds. The online algorithms make no assumption about how the sequence of examples are generated, which is a key different from the batch model learning. The goal of the algorithm is to make as many correct predictions as possible.

The online learning algorithm uses a hypothesis $f_t : \mathcal{X} \to \mathbb{R}$ to generate the predictions

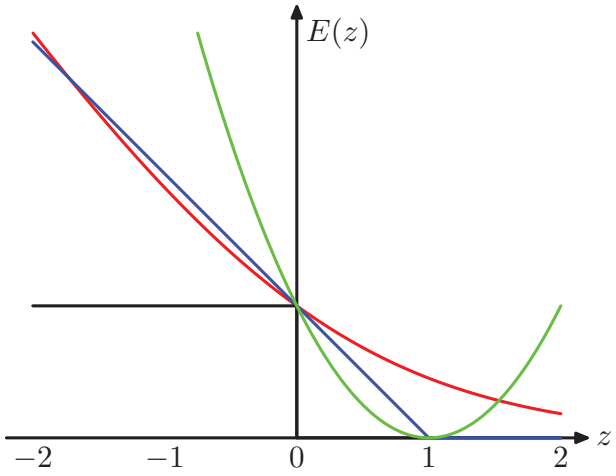


Figure 6.1: Different loss functions: zero-one loss (black), hinge loss (black), square loss (green) and logistic loss (red). The logistic loss is rescaled by a factor of $1/\ln(2)$ to pass (0,1).

in each round t, where $\text{sgn}(f_t(\mathbf{x}_t))$ is the prediction. We use a loss function $\ell(y_t, f_t(\mathbf{x}_t))$ to evaluate the predictions of a hypothesis. The commonly used loss functions include the following which are illustrated in Figure 6.1.

- Zero-one loss:

$$\ell(y, f(\mathbf{x})) = \begin{cases} 0 & yf(\mathbf{x}) > 0 \\ 1 & \text{otherwise} \end{cases}$$

- Hinge loss:

$$\ell(y, f(\mathbf{x})) = \max(0, 1 - yf(\mathbf{x}))$$

- Square loss:

$$\ell(y, f(\mathbf{x})) = (1 - yf(\mathbf{x}))^2$$

- Logistic loss:

$$\ell(y, f(\mathbf{x})) = \ln(1 + \exp(-yf(\mathbf{x})))$$

The objective of the online learning algorithm is to minimize the total loss $\sum_{t=1}^{T} \ell(y_t, f_t(\mathbf{x}_t))$ in the long run.

Algorithm 7 Perceptron Algorithm

Initialization
- $\mathbf{w}_1 = 0$

for $t = 1, 2, \ldots, T$ **do**
  Receive $\mathbf{x}_t$
  Make prediction $\text{sgn}(\mathbf{x}_t^\top \mathbf{w}_t)$
  Receive y_t
  if $y_t \mathbf{x}_t^\top \mathbf{w}_t \leq 0$ **then**
    $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y_t \mathbf{x}_t$
  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$
  end if
end for

The epic Rosenblatt's Perceptron algorithm [Rosenblatt, 1958], despite of its simplicity, has a nice error bound and works well in experiments. The algorithm is described in Algorithm 7. The early study of online learning was focused on linear classification model, which was later on extended to nonlinear classifier by using the kernel trick. Inspired by the success of maximum margin classifiers, various algorithms have been proposed to incorporate classification margin into online learning. The connection between online learning and repetitive game has also inspired numerous algorithms for online learning. Recent studies explored the close relationship between online learning and optimization theories, particularly stochastic approximation (e.t., stochastic gradient descent) and first order methods (e.g., subgradient descent).

Since the debut of Perceptron over half a century ago, many algorithms have been proposed for online learning [Cesa-Bianchi and Lugosi, 2006]. Despite the success, most online learning studies assume that the learner has the full access to all input features. However, in many real-world applications, objects are represented by thousands or even millions of features and measure all input features can be computationally expensive. One example is visual object recognition, for which the most popular approach is bag-of-words model (BoW). The BoW model introduces the visual vocabulary, with each visual word in the vocabulary represent a distinct visual pattern, and represents the visual content of images by histograms of visual

words. In order to capture diverse visual patterns of objects, the BoW model often has to introduce thousands even millions of visual words, leading to a high computational cost in computing visual word histograms. Specifically, a sparse online learning model is desirable for the following reasons:

- Space constraints: online algorithms are often applied to high dimensional data. As a result, the model of the online learning algorithm itself might overflow the memory and makes impossible to implement the algorithm.
- Test time constraints and computation: substantially reducing the number of features may significantly reduce the computational time to evaluate new samples.
- Overfitting: it has been theoretically and experimentally proved [Vapnik et al., 1994] that by regularizing the model and introducing sparsity, we can avoid the overfitting problem and increase the accuracy of the model.
- Cost to acquire the data: sometimes acquiring the full knowledge of the examples is costly. With limited budgets, it is desirable if we have a sparse model, as a result we only need to acquire the features corresponding to the non-zero features in the model.

In this chapter, we focus on the online learning problems where objects are represented by many attributes. Since it is expensive, either computationally or money wise, to acquire all the input attributes of an object, it is therefore desirable to develop online learning algorithms that only need to sense a small number of attributes before the reliable decision can be made.

To address this challenge, we will investigate the following protocol of online learning: Let $\mathbf{x}_1, \ldots, \mathbf{x}_T$ be a sequence of input patterns received over the trials, where each $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of d dimension. In our study, we assume d is a very large number and for computational efficiency we need to select a relatively small number of features for linear classification. More specifically, in each trial t, the learner presents a classifier $\mathbf{w}_t \in \mathbb{R}^d$ that

will be used to classify instance $\mathbf{x}_t$ by a linear function $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_t)$. Instead of using all the features for classification, we require the classifier $\mathbf{w}_t$ to have at most B non-zero elements and consequently at most B features of $\mathbf{x}_t$ will be used for classification. We refer to this problem as *online feature selection*. Our goal is to design an effective strategy for online feature selection that is able to make a small number of mistakes. Throughout the following text, we assume $|\mathbf{x}_t|_2 \leq 1, t = 1, \ldots, T$.

6.2 Related Work

It is worthwhile to note that the *online feature selection* defined above is a novel problem, and is not yet addressed in any research work so far at the author's awareness. It is related to, but significantly different from several existing machine learning tasks:

- *Batch mode feature selection:* Unlike the batch model feature selection [Jain and Zongker, 2002] that learns from a collection of training examples where all the input features are provided, in every trial of online feature selection, only a small number of input features are acquired, making it significantly more challenging problem.

- *Sparse online learning:* Sparse online learning aims to learn a statistical model that only utilizes a small number of input features. Similar to batch mode feature selection, sparse online learning assumes all the input features of training examples are provided. Several sparse online learning algorithms are proposed recently. In [Langford et al., 2009], the authors proposed an algorithm of *truncated gradient descent*. The basic idea is that after the gradient decent step, the coefficients of the linear classifier are shrunk by a small amount if its value is within certain threshold, and thus sparsity is achieved. Recently in [Duchi and Singer, 2009] the authors proposed a framework for empirical risk minimization with regularization called *forward looking subgradients*. The basic idea is to solve a regularized optimization problem after every gradient-descent step. Note that our work differs from these studies in that we impose a hard constraint on

the number of non-zero elements in classifier $\mathbf{w}$, while all the studies on sparse online learning only have soft constraints on the sparsity of the classifier.

- *Learning with missing features:* Similar to online feature selection, in learning with missing features [Ghahramani and Jordan, 1997], only part of features of training examples are observed. However, unlike online feature selection where the learner actively selects a subset of features for measuring, in learning with missing features, the learner does not control which features to measure.

- *Budget online learning:* Our work is also related to budget online learning [Cavallanti et al., 2007, Dekel et al., 2008, Orabona et al., 2008] in that the number of support vectors is bounded by a predefined number. A common strategy behind many budget online learning algorithms is to remove the "oldest" support vector when the maximum number of support vectors is reached. This simple strategy however is not applicable to online feature selection, because in the case of linear kernel, the combination of limited number of support vectors does not guarantee a sparse classifier.

- Finally, our work is closely related to the online learning algorithm that aims to learn a classifier that performs as well as the best subset of experts [Warmuth and Kuzmin, 2006], which is in contrast to most online learning work on prediction with expert advice that only compares to the best expert in the ensemble [Cesa-Bianchi and Lugosi, 2006]. Unlike the work [Warmuth and Kuzmin, 2006] where only positive weights are assigned to individual experts, in our study, the weights assigned to individual features can be both negative and positive, making it more flexible.

6.3 A Naive Approach

A naive approach for online feature selection is to modify the Perceptron algorithm. In the t-th trial, when asked to make prediction, we will truncate the classifier $\mathbf{w}_t$ by setting

Algorithm 8 A Modified Perceptron for Online Feature Selection

Input
- B: the number of selected features

Initialization
- $\mathbf{w}_1 = 0$

for $t = 1, 2, \ldots, T$ **do**
  Receive $\mathbf{x}_t$
  Make prediction $\text{sgn}(\mathbf{x}_t^\top \mathbf{w}_t^B)$ where $\mathbf{w}_t^B$ is $\mathbf{w}_t$ with everything but the B largest elements set to zero.
  Receive y_t
  if $y_t \mathbf{x}_t^\top \mathbf{w}_t^B \leq 0$ **then**
    $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t$
  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$
  end if
end for

everything but the B largest elements in $\mathbf{w}_t$ to be zero. This truncated classifier, denoted by $\mathbf{w}_t^B$, is then used to classify the received instance $\mathbf{x}_t$. Similar to the Perceptron algorithm, when the instance is misclassified, we will update the classifier by adding the vector $y_t \mathbf{x}_t$ where $(\mathbf{x}_t, y_t)$ is the misclassified training example. Algorithm 8 shows the pseudo code of this approach.

Unfortunately, this simple approach does not work: it can not guarantee a decent bound for mistakes. To see this, consider the case where the input pattern $\mathbf{x}$ can only take two possible values, either $\mathbf{w}_a$ or $\mathbf{w}_b$. For $\mathbf{w}_a$, we set its first B elements to be one and the remaining elements to be zero. For $\mathbf{w}_b$, we set its $B + 1$ to $2B$ elements to be one and the remaining elements to be zero. An instance $\mathbf{x}$ is assigned to the positive class (i.e., $y = +1$) when it is $\mathbf{w}_a$, and assigned to the negative class (i.e., $y = -1$) when it is $\mathbf{w}_b$. Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{2T}, y_{2T})$ be a sequence of $2T$ examples, with $\mathbf{x}_{2k+1} = \mathbf{w}_a, y_{2k+1} = 1, k = 0, \ldots, T-1$ and $\mathbf{x}_{2k} = \mathbf{w}_b, y_{2k} = -1, k = 1, \ldots, T$. It is clear that Algorithm 8 will always make the mistake while a simple classifier that uses only two attributes (i.e., the first feature and the $B + 1$ feature) will make almost no mistakes.

6.4 $L1$ Projection for Online Feature Selection

One reason for the failure of Algorithm 9 is that although it selects the B largest elements for prediction, it does not guarantee that the numerical values for the unselected attributes are sufficiently small, which could potentially lead to many classification mistakes. We can avoid this problem by exploring the sparsity property of $L1$ norm, given in the following proposition.

Proposition 5. *(from [Donoho, 2006]) For $q > 1$ and $\mathbf{x} \in \mathbb{R}^d$, we have*

$$|\mathbf{x} - \mathbf{x}^m|_q \leq \xi_q |\mathbf{x}|_1 (m+1)^{1/q-1}, m = 1, \ldots, d$$

where ξ_q is a constant depending only on q and $\mathbf{x}^m$ stands for the vector $\mathbf{x}$ with everything but the m largest elements set to 0.

This proposition indicates that when a vector $\mathbf{x}$ lives in a $L1$ ball, most of its numerical values are concentrated in its largest elements, and therefore removing the smallest elements will result in a very small change to the original vector measured by the L_q norm. Thus, we will enforce the classifier to be restricted to a $L1$ ball, i.e.,

$$\Delta_R = \{\mathbf{w} \in \mathbb{R}^d : |\mathbf{w}|_1 \leq R\} \tag{6.1}$$

Based on this idea, algorithm 9 gives a method for online feature selection. The learner maintains a linear classifier $\mathbf{w}_t$ that has at most B non-zero elements. When a training instance $(\mathbf{x}_t, y_t)$ is misclassified, the learner will update the classifier by first adding the training instance $y_t \mathbf{x}_t$ to the existing classifier $\mathbf{w}_t$ and then projecting it into the $L1$ ball Δ_R. If the resulting classifier $\widehat{\mathbf{w}}_{t+1}$ has more than B non-zero elements, we will simply keep the B elements in $\widehat{\mathbf{w}}_{t+1}$ with the largest absolute weights. The main computational challenge in Algorithm 9 is step 8 that projects a vector into $L1$ ball Δ_R. It requires solving the following optimization problem

$$\min_{|\mathbf{x}|_1 \leq R} |\mathbf{x} - \mathbf{a}|_2^2$$

Algorithm 9 Online Feature Selection using L_2 Norm

Input
- R: maximum $L1$ norm
- B: the number of selected features

Initialization
- $\mathbf{w}_1 = 0$

for $t = 1, 2, \ldots, T$ **do**
  Receive $\mathbf{x}_t$
  Make prediction $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_t)$
  Receive y_t
  if $y_t \mathbf{w}_t^\top \mathbf{x}_t \leq 0$ **then**
    $\widehat{\mathbf{w}}_{t+1} = \pi_{\Delta_R}(\mathbf{w}_t + y_t \mathbf{x}_t)$ where $\pi_{\Delta_R}(\mathbf{x})$ projects $\mathbf{x}$ into domain Δ_R using $L2$ norm.
    if $|\widehat{\mathbf{w}}_{t+1}|_0 > B$ **then**
      $\mathbf{w}_{t+1} = \widehat{\mathbf{w}}_{t+1}^B$ where $\widehat{\mathbf{w}}_{t+1}^B$ is vector $\widehat{\mathbf{w}}_{t+1}$ with everything but the B largest elements set to be zero.
    else
      $\mathbf{w}_{t+1} = \widehat{\mathbf{w}}_{t+1}$
    end if
  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$
  end if
end for

where $\mathbf{a}$ is a given vector. The solution to the above optimization problem is given by

$$x_i = [|a_i| - \lambda]_+ \text{sgn}(a_i), i = 1, \ldots, d$$

where $[x]_+ = \max(0, x)$. Dual variable $\lambda \geq 0$ is either 0 if $\sum_{i=1} |a_i| \leq R$ or given by the following nonlinear equation

$$\sum_{i=1}^{d} [|a_i| - \lambda]_+ = R,$$

which can be solved efficiently using bisection search.

The following theorem gives the mistake bound for the Algorithm 9.

Theorem 2. *After running Algorithm 9 over a sequence of training examples* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ *with* $|\mathbf{x}_t|_2 \leq 1, t \in [T]$, *we have the following the bound for the number*

of mistakes M made by Algorithm 9

$$M \leq \frac{1}{1 - \left[\frac{\xi_2^2 R^2}{B} + 4\frac{\xi_2 R^2}{\sqrt{B}}\right]} \left\{ \min_{\mathbf{w} \in \Delta} |\mathbf{w}|_2^2 + 2\sum_{t=1}^{T} \ell(y_t, \mathbf{w}^\top \mathbf{x}_t) \right\}$$

Proof. Consider any trial t when the received training example $(\mathbf{x}_t, y_t)$ is misclassified. For any classifier $\mathbf{w} \in \Delta$, we have

$$\ell(y_t \mathbf{w}_t^\top \mathbf{x}_t) \leq \ell(y_t \mathbf{w}^\top \mathbf{x}_t) + \frac{1}{2}|\mathbf{w} - \mathbf{w}_t|_2^2 - \frac{1}{2}|\mathbf{w} - \widehat{\mathbf{w}}_{t+1}|_2^2 + \frac{1}{2}|\mathbf{x}_t|_2^2$$

When the number of non-zero elements in $\widehat{\mathbf{w}}_{t+1}$ is more than B, we will generate $\mathbf{w}_{t+1}$ by only keeping the B largest elements in $\widehat{\mathbf{w}}_{t+1}$, which leads to an additional term $(|\mathbf{w} - \mathbf{w}_{t+1}|_2^2 - |\mathbf{w} - \widehat{\mathbf{w}}_{t+1}|_2^2)/2$ on the right hand side of the above inequality. Using Proposition 1, we have

$$\begin{aligned} & |\mathbf{w} - \mathbf{w}_{t+1}|_2^2 - |\mathbf{w} - \widehat{\mathbf{w}}_{t+1}|_2^2 \\ = \; & |\mathbf{w}_{t+1} - \widehat{\mathbf{w}}_{t+1}|_2^2 + 2(\widehat{\mathbf{w}}_{t+1} - \mathbf{w}_{t+1})^\top (\mathbf{w} - \widehat{\mathbf{w}}_{t+1}) \\ \leq \; & \frac{\xi_2^2 R^2}{B} + 4\frac{\xi_2 R^2}{\sqrt{B}} \end{aligned}$$

We thus have

$$\begin{aligned} & \ell(y_t \mathbf{w}_t^\top \mathbf{x}_t) \\ \leq \; & \ell(y_t \mathbf{w}^\top \mathbf{x}_t) + \frac{1}{2}|\mathbf{w} - \mathbf{w}_t|_2^2 - \frac{1}{2}|\mathbf{w} - \widehat{\mathbf{w}}_{t+1}|_2^2 + \frac{1}{2} + \frac{1}{2}\left(\frac{\xi_2^2 R^2}{B} + 4\frac{\xi_2 R^2}{\sqrt{B}}\right) \end{aligned}$$

We complete the proof by adding up the inequalities of all trials and using the fact that $\ell(y_t \mathbf{w}_t^\top \mathbf{x}_t) \geq 1$ when $y_t \mathbf{w}_t^\top \mathbf{x}_t \leq 0$. □

One limitation of Algorithm 9 is that according to Theorem 1, in order to give a meaningful bound of mistakes, we should have $R < O(B^{1/4})$, leading to a very small $L1$ ball for the comparator $\mathbf{w}$. We can improve Algorithm 9 by using L_{2K} norm, instead of $L2$ norm, as the potential function, where $K \geq 1$ is an integer equal to or larger than 1. The overall idea follows the framework of potential gradient descent approaches [Cesa-Bianchi and Lugosi,

2006]. In this algorithm, we maintain two sets of solutions: $\mathbf{u}_t \in \mathbb{R}^d$ is a solution in the dual space and $\mathbf{w}_t \in \mathbb{R}^d$ is a solution in the primary space. The mapping between solution $\mathbf{u}_t$ and $\mathbf{w}_t$ are defined as follows:

$$\mathbf{u}_t = \nabla\Phi(\mathbf{w}_t),\ \mathbf{w}_t = \nabla\Phi_*(\mathbf{u}_t) \tag{6.2}$$

where

$$\Phi(\mathbf{x}) = \frac{1}{2}|\mathbf{x}|_{2K}^2,\ \Phi_*(\mathbf{x}) = \frac{1}{2}|\mathbf{x}|_{2K/[2K-1]}^2 \tag{6.3}$$

For the convenience of presentation, we define

$$q = 2K/(2K-1) \tag{6.4}$$

and therefore $\Phi_*(\mathbf{x}) = |\mathbf{x}|_q^2/2$. In each trial, we first make prediction using $\mathbf{w}_t$ for the received instance $\mathbf{x}_t$. When the prediction is incorrect, we update the solution in the dual space and map it to the primary space by the transform $\nabla\Phi_*$. The mapped solution is projected into the $L1$ ball and further truncated if the projected solution has more than B non-zero entries. The final solution $\mathbf{w}_{t+1}$ will be mapped back to the dual space to generate the dual solution $\mathbf{u}_{t+1}$. Algorithm 10 gives the detailed steps of this algorithm. One of the key steps is to project solution into a $L1$ ball under the L_{2K} norm. The solution is given by the following optimization problem

$$\min_{|\mathbf{x}|_1 \le R} |\mathbf{x} - \mathbf{a}|_{2K}^{2K} \tag{6.5}$$

We have the following proposition for the solution to the optimization problem in (6.5).

Proposition 6. *The optimal solution to Equation 6.5 is given by*

$$x_i = sgn(a_i)[a_i - \lambda^{1/(2K-1)}]_+$$

where dual variable $\lambda \ge 0$ is either 0 or the solution to the following nonlinear inequality

$$\sum_{i=1}^{d}[a_i - \lambda^{1/(2K-1)}]_+ = R$$

Proof. We first convert Equation (6.5) into a convex-concave optimization problem

$$\max_{|\gamma|_\infty \le \lambda, \lambda \ge 0} \min_{\mathbf{x}} \frac{1}{2K} |\mathbf{x} - \mathbf{a}|_{2K}^{2K} + \gamma^\top \mathbf{x} - \lambda R$$

By minimizing over $\mathbf{x}$, we have

$$x_i = a_i - \gamma_i^{1/(2K-1)}, i = 1, \ldots, d$$

Using the above solution, we simplify the convex-concave optimization problem into the following maximization problem

$$\max_{\lambda, |\gamma|_\infty \le \lambda} \sum_{i=1}^{d} \left\{ -\frac{2K-1}{2K} \gamma_i^{2K/(2K-1)} + a_i \gamma_i \right\} - \lambda R$$

By maximizing over γ_i with λ fixed, we have

$$\gamma_i = \operatorname{sgn}(a_i) \min \left(\lambda, |a_i|^{2K-1} \right), i = 1, \ldots, d$$

Hence, the solution for x_i becomes

$$x_i = \operatorname{sgn}(a_i) \left[|a_i| - \lambda^{1/(2K-1)} \right]_+$$

Finally, λ is given by the following nonlinear inequality

$$\sum_{i=1}^{d} |x_i| = \sum_{i=1}^{d} \left[|a_i| - \lambda^{1/(2K-1)} \right]_+ \le R$$

We will first check if $\lambda = 0$ will satisfy the inequality. If not, we conduct bisection search to find $\lambda > 0$. □

We have the following theorem for the mistake bound when using L_{2K} norm.

Theorem 3. *After running Algorithm 10 over a sequence of training examples* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ *with* $|\mathbf{x}_t|_2 \le 1, t \in [T]$, *we have the following bound for the number of mistakes made by Algorithm 10*

$$M \le \frac{1}{1 - \left[\frac{\xi_{2K}^2 R^2}{B^{2-1/K}} + 4 \frac{\xi_{2K} R^2}{B^{1-1/[2K]}} \right]} \left\{ \min_{\mathbf{w} \in \Delta} |\mathbf{w}|_{2K}^2 + 2 \sum_{t=1}^{T} \ell(y_t \mathbf{w}^\top \mathbf{x}_t) \right\}$$

The proof is almost word-by-word copy of that for Theorem 1. Compared to Algorithm 9, we only require $R < O(B^{1/2-1/[4K]})$, which is close to $O(\sqrt{B})$ when K is sufficiently large.

Algorithm 10 Online Feature Selection using L_{2K} Norm

Input

- R: maximum $L1$ norm
- B: the number of selected features
- K: norm used for potential function

Initialization

- $\mathbf{u}_1 = \mathbf{w}_1 = 0$
- $\Phi(\mathbf{x}) = \frac{1}{2}|\mathbf{x}|_{2K}^2$ and $\Phi_*(\mathbf{x}) = \frac{1}{2}|\mathbf{x}|_q^2$, where $q = 2K/(2K-1)$.

for $t = 1, 2, \ldots, T$ **do**
  Receive $\mathbf{x}_t$
  Make prediction $\text{sgn}(\mathbf{w}_t^\top \mathbf{x}_t)$
  Receive y_t
  if $y_t \mathbf{w}_t^\top \mathbf{x}_t \leq 0$ **then**
    $\widehat{\mathbf{w}}_{t+1} = \pi_{\Delta_R}^K(\nabla \Phi_*(\mathbf{u}_t + y_t \mathbf{x}_t))$ where $\pi_{\Delta_R}^K(\mathbf{x})$ projects $\mathbf{x}$ into domain Δ_R using L_{2K} norm.
    if $|\widehat{\mathbf{w}}_{t+1}|_0 > B$ **then**
      $\mathbf{w}_{t+1} = \widehat{\mathbf{w}}_{t+1}^B$ where $\widehat{\mathbf{w}}_{t+1}^B$ is vector $\widehat{\mathbf{w}}_{t+1}$ with everything but the B largest elements set to be zero.
    else
      $\mathbf{w}_{t+1} = \widehat{\mathbf{w}}_{t+1}$
    end if
    $\mathbf{u}_{t+1} = \nabla \Phi(\mathbf{w}_{t+1})$
  else
    $\mathbf{w}_{t+1} = \mathbf{w}_t$, $\mathbf{u}_{t+1} = \mathbf{u}_t$
  end if
end for

6.5 Experiments

In this section we present the empirical studies of the proposed algorithms.

6.5.1 Datasets

We used two datasets in our experiments: 20 Newsgroup and Reuters dataset. Table 6.1 shows the information of the datasets we used.

- The 20 Newsgroup dataset is a collection of 20,000 messages collected from 20 different Usenet newsgroups – 1000 messages from each of the 20 newsgroups were chosen,

Name	# Samples	# Features
20news-autos	1978	8165
20news-space	1974	8165
reuters-money	1434	5180
reuters-acq	4738	5180

Table 6.1: Information of the 20 Newsgroup and Reuters datasets.

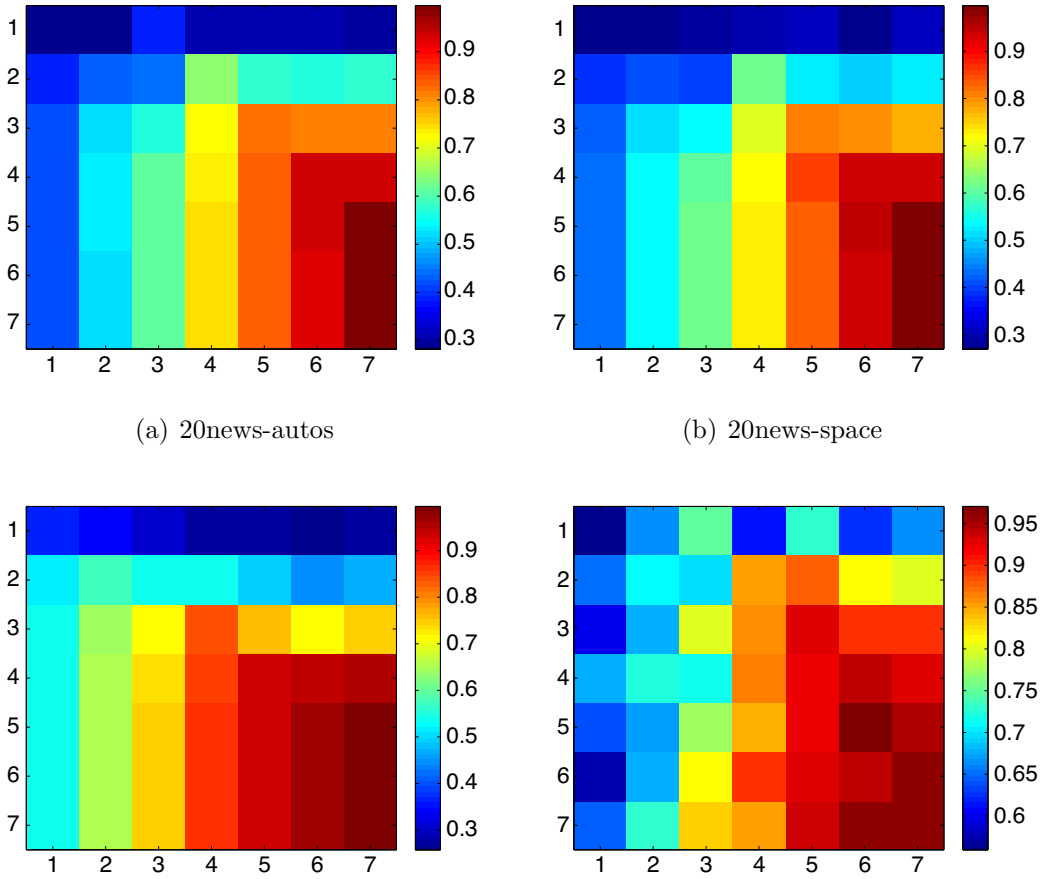
and the dataset was partitioned by the newsgroup name. We randomly chose group "rec.autos" and "sci.space" in our experiments.

- The Reuters-21578 dataset is a standard text categorization benchmark containing stories from Reuters news agency grouped into 135 classes. We randomly chose the group "money-fx" and "acq" in our experiments.

All documents are TF-IDF weighted. For each category in a dataset, we use all documents in this category as positive samples, and we sample the same amount documents from other categories and use as negative samples. Thus the data is balanced for positive and negative labels.

6.5.2 Experimental Results

For each of the four datasets, we randomly sample 90% of examples and apply Algorithm 9 for 100,000 iterations. We calculate the cumulative accuracy at the end of the horizon, and we do the offline evaluation by applying the resulting classifier to the remaining 10% examples. Two important parameters in Algorithm 9 are the number of features B and the norm ball size R. We vary the value of these two parameters. For each setting of parameters, we run the algorithm 5 times, each with a random sampling of training samples as well as the input sequences. The performance of the 5 runs are averaged. Figure 6.2 and Figure 6.3 illustrates the online and offline performance on the respectively.



(a) 20news-autos

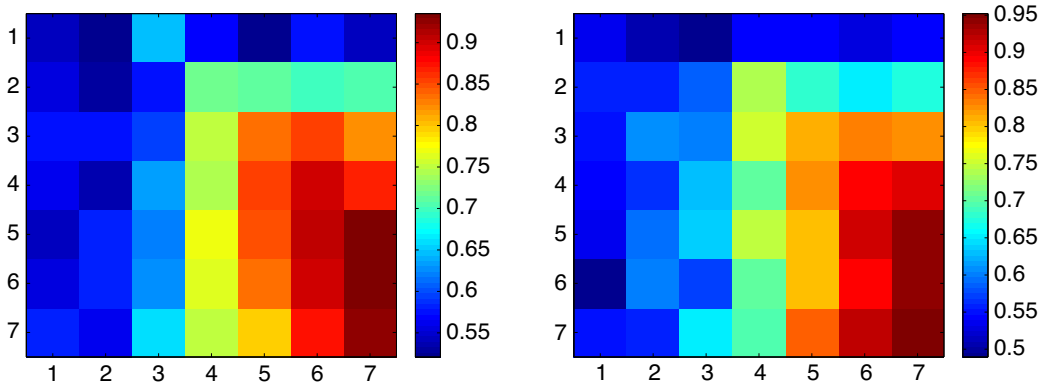
(b) 20news-space

(c) reuters-acq

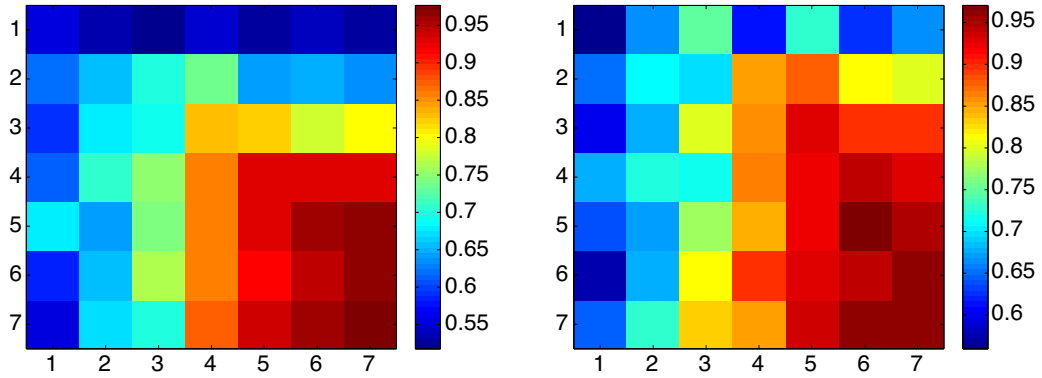
(d) reuters-money

Figure 6.2: Cumulative accuracy of Algorithm 6.2 at the end of horizon. The y axis represents varying B values: number 1 to 7 corresponds to 20, 50, 100, 200, 500, 1000 and 2000 respectively. The x axis represents varying R values: number 1 to 7 corresponds to 5, 10, 20, 50, 100, 200 and 500 respectively.

The experimental results is consistent with our theoretical analysis. The performance of the algorithm increases with increasing B and R both in the online evaluation and offline evaluation.



(a) 20news-autos (b) 20news-space



(c) reuters-acq (d) reuters-money

Figure 6.3: Offline evaluation using the model generated at the end of horizon from Algorithm 6.2. The y axis represents varying B values: number 1 to 7 corresponds to 20, 50, 100, 200, 500, 1000 and 2000 respectively. The x axis represents varying R values: number 1 to 7 corresponds to 5, 10, 20, 50, 100, 200 and 500 respectively.

6.6 Conclusion

In this chapter, we continue our interests in sparse structures, specifically in online feature selection. Despite the success of online learning algorithms, most studies assume that the learning has full access to all input features. However, in many real world applications, it is expensive, either computationally or money wise to acquire and use all the input attributes. As a result, it is desirable to develop online learning algorithms that only need to sense a

small number of attributes before the reliable decision can be made. We study the topic in this area, and develop theories and algorithms for sparse online feature selection. Specifically, we propose the general algorithms for online feature selection, and examine their theoretic properties such as the regret bound and the upper and lower bounds for the regret bound. We evaluate the proposed algorithms by experiments on benchmark datasets.

CHAPTER 7

Conclusion

The learning of structures is an important topic in the field of machine learning as it finds applications in numerous domains. The key contributions to this area that the author made in this dissertation are following.

- **Application of DBN for cortical network reconstruction:** In Chapter 3, the dynamic Bayesian networks is applied to identifying functional cortical networks from simultaneously recorded spike trains. An empirical study with cortical network reconstruction is presented to show the advantage of DBNs in comparison to the state-of-the-art approaches. In particular, the experiments demonstrate the strong power of DBN in inferring the topology of the networks with unknown latency and weak signals. The author and his collaborators are among the first to apply DBN to cortical network reconstruction.

- **A novel KMF framework for gene regulatory network reconstruction:** In Chapter 4, a novel knowledge driven matrix factorization (KMF) framework is presented to meet the challenging problem of reconstructing gene networks from multiple information sources. In KMF, gene expression data is initially used to estimate the correlation matrix. The gene modules and the interactions among the modules are derived by factorizing the correlation matrix. The prior knowledge in GO is integrated into matrix factorization to help identify the gene modules. The advantage of proposed framework is that it derives both the gene modules and their interactions in a unified framework of matrix factorization, and it incorporates the prior knowledge of co-regulation relationships from GO information into the network reconstruction pro-

cess. An alternating optimization algorithm is presented to efficiently find the solution of the related optimization problem. Experiments show that the proposed algorithm performs significantly better in identifying gene modules than several state-of-the-art algorithms, and the interactions among the modules uncovered by the algorithm are proved to be biologically meaningful.

- **A novel exclusive lasso regularization:** In Chapter 5, a novel group regularization called *exclusive lasso* is presented. Unlike the group lasso regularizer that assumes co-varying variables in groups, the proposed exclusive lasso regularizer models the scenario when variables in the same group compete with each other. Analysis is presented to illustrate the properties of the proposed regularizer. A framework of kernel-based multi-task feature selection algorithm based on the proposed exclusive lasso regularizer is also proposed for the application of proposed regularizer. An efficient algorithm is derived to solve the related optimization problem. Experiments with document categorization show that the proposed approach outperforms state-of-the-art algorithms for multi-task feature selection.

- **First step for solving a novel problem of online feature selection:** In Chapter 6, the author further investigate the sparse structure in online learning, specifically in online feature selection. Most online learning studies assume that the learning has full access to all input features. However, in many real world applications, it is expensive, either computationally or money wise to acquire and use all the input attributes. In this case it is desirable to develop online learning algorithms that only need to sense a small number of attributes before the reliable decision can be made. The author proposed a new problem of *online feature selection*. In the first step towards solving this problem, the author develops theories and algorithms for sparse online feature selection. Specifically, some general algorithms for online feature selection are developmed, and their theoretic properties such as the upper and lower bounds for the

regret bound are examined. The efficacy of the proposed algorithms are also examined by extensive experiments on benchmark datasets.

Learning with structures is a broad and interesting topic in the area of machine learning. Besides the successful studies conducted in this dissertation, there are a number of challenging problems that deserve further investigation in the future. For example, given the regulatory network, how to identify the important genes or miRNAs responsible for certain diseases like cancers. This information would be important for understanding the cause of the diseases as well as to design drugs and drug delivery for treating these diseases. Another example is the novel *online feature selection* problem proposed in Chapter 6. Although the author made some first steps towards solving it, it remains to be a tough problem and requires deeper investigation.

APPENDICES

APPENDIX A

Notations

Sets

$\mathbb{R}$	Real numbers
$\mathbb{R}^n$	Real n-vectors
$\mathbb{R}^{m\times n}$	Real $m \times n$ matrices
$\mathbb{R}_+, \mathbb{R}_{++}$	Nonnegative, positive real numbers
$\mathbb{S}^n$	Symmetric $n \times n$ matrices
$\mathbb{S}^n_+, \mathbb{S}^n_{++}$	Symmetric positive semi-definite, positive definite, $n \times n$ matrices

Vectors and matrices

$\mathbf{1}$	Vector with all components one
I	Identity matrix
$X^\top$	Transpose of matrix X
$\mathrm{tr}(X)$	Trace of matrix X
$\mathrm{rank}(X)$	Rank of matrix X

Norms and distances

$\|\cdot\|$	A norm
$\|\mathbf{x}\|_2$	Euclidean (or L_2-) norm of vector $\mathbf{x}$
$\|X\|_F$	Frobenius norm of matrix X
$\|X\|_{tr}$	Trace norm of matrix X

Generalized inequalities

$\mathbf{x} \preceq \mathbf{y}$	Componentwise inequality between vectors $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{x} \prec \mathbf{y}$	Strick componentwise inequality between vectors $\mathbf{x}$ and $\mathbf{y}$
$X \preceq Y$	Matrix inequality between matrices X and Y
$X \prec Y$	Strict matrix inequality between matrices X and Y

APPENDIX B

Pearson Correlation

Pearson correlation is obtained by dividing the covariance of the two variables by the product of their standard deviations:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

where X and Y are two random variables, μ_X and μ_Y are their expectations respectively, σ_X and σ_Y are their standard deviation respectively.

The Pearson correlation is defined only if both of the standard deviations are finite and both of them are non-zero.

The value of Pearson correlation is in $[-1, 1]$. It is 1 in the case of an increasing linear relationship, -1 in the case of a decreasing linear relationship, and some value between -1 and 1 in all other cases, indicating the degree of linear dependence between the two variables. The closer the coefficient is to either -1 or 1, the stronger the correlation between the two variables.

APPENDIX C

Bregman Divergence

Let $\varphi : S \to \mathbb{R}$ be a differentiable, strictly convex function of Legendre type ($S \in \mathbb{R}^d$), then the *Bregman divergence* $D_\varphi : S \times \text{relint}(S) \to \mathbb{R}$ is defined as

$$D_\varphi(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) - \varphi(\mathbf{y}) - (\mathbf{x} - \mathbf{y})^\top \nabla\varphi(\mathbf{y})$$

Figure C.1 is an illustration of Bregman divergence.

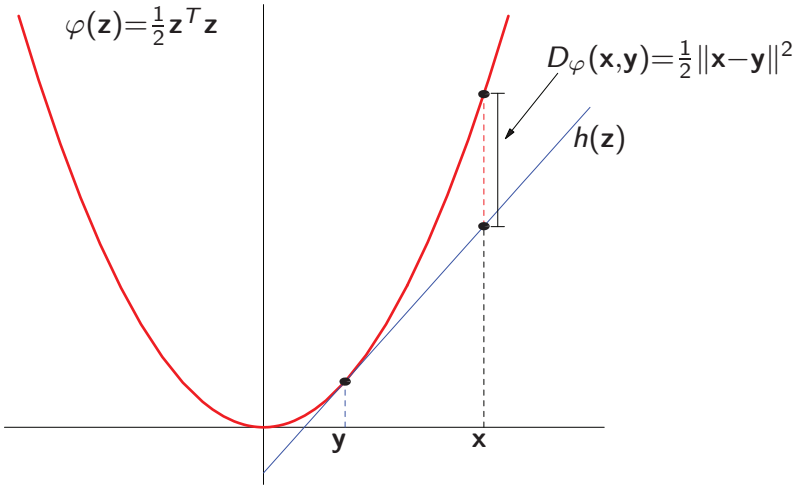


Figure C.1: Squared Euclidean distance is a Bregman divergence.

Bregman divergence has the following properties:

- Non-negativity: $D_\varphi(\mathbf{x}, \mathbf{y}) \geq 0$ and equals 0 iff $\mathbf{x} = \mathbf{y}$.
- Convexity: $D_\varphi(\mathbf{x}, \mathbf{y})$ is convex in its first argument, but not necessarily in the second argument.
- Bregman divergence is not a metric (symmetry, triangle inequality do not hold).

- Three point property generalizes the "Law of cosines":

$$D_\varphi(\mathbf{x}, \mathbf{y}) = D_\varphi(\mathbf{x}, \mathbf{z}) + D_\varphi(\mathbf{z}, \mathbf{y}) - (\mathbf{x} - \mathbf{z})^\top (\nabla\varphi(\mathbf{y}) - \nabla\varphi(\mathbf{z}))$$

APPENDIX D

Kullback-Leibler Divergence

Given two probabilistic distributions ν and μ, the *relative entropy* of ν respect to μ, or the *Kullback-Leibler divergence* of ν from μ, is

$$D_{\mathrm{KL}}(\mu\|\nu) = -E_{\mu}\left[\log\frac{d\nu}{d\mu}\right]$$

where E_{μ} is the expectation with respect to μ. The Kullback-Leibler divergence is non-negative, and it is zero iff $\mu = \nu$.

The Kullback-Leibler divergence can be interpreted as the expected extra message-length per datum that must be communicated if a code that is optimal for a given (wrong) distribution ν is used, compared to using a code based on the true distribution μ:

$$D_{\mathrm{KL}}(\mu\|\nu) = H(\mu,\nu) - H(\mu)$$

where $H(\mu,\nu)$ is the cross entropy of μ and ν, and $H(\mu)$ is the entropy of μ.

APPENDIX E

Mutual Information

The *mutual information* of two discrete random variables X and Y is defined as:

$$I(X;Y) = \int_Y \int_X p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right) \mathrm{d}x \, \mathrm{d}y,$$

where $p(x,y)$ is the joint probability density function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability density functions of X and Y respectively.

Mutual information can be intuitively interpreted as the information that X and Y share.

Mutual information can be expressed using entropy as:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \\ &= H(X,Y) - H(X|Y) - H(Y|X) \end{aligned}$$

Mutual information can also be expressed as a Kullback-Leibler divergence of the product $p(x) \times p(y)$ of the marginal distributions of the two random variables X and Y from the random variables' joint distribution $p(x,y)$:

$$I(X;Y) = D_{\mathrm{KL}} \left(p(x,y) \| p(x)p(y) \right)$$

BIBLIOGRAPHY

BIBLIOGRAPHY

A. M. Aertsen, G. L. Gerstein, M. K. Habib, and G. Palm. Dynamics of neuronal firing correlation: modulation of "effective connectivity". *Journal of Neurophysiology*, 61(5): 900–917, 1989.

Ash A. Alizadeh, Michael B. Eisen, R. Eric Davis, Chi Ma, Izidore S. Lossos, Andreas Rosenwald, Jennifer C. Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, John I. Powell, Liming Yang, Gerald E. Marti, Troy Moore, James Hudson, Lisheng Lu, David B. Lewis, Robert Tibshirani, Gavin Sherlock, Wing C. Chan, Timothy C. Greiner, Dennis D. Weisenburger, James O. Armitage, Roger Warnke, Ronald Levi, Wyndham Wilson, Michael R. Grever, John C. Byrd, David Botstein, Patrick O. Brown, and Louis M. Staudt. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769): 503–511, 2000.

Tim Van Allen and Russell Greiner. Model selection criteria for learning belief nets: An empirical comparison. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1047–1054, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2.

Orly Alter, Patrick O. Brown, and David Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences of the United States of America*, 97(18):10101–10106, 2000.

S. Amari. Measure of correlation orthogonal to change in firing rate. *Neural computation*, 21(4):960–972, 2009.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *NIPS*, 2006.

Adam Arkin, John Ross, and Harley H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected escherichia coli cells. *Genetics*, 149(4): 1633–1648, 1998.

Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004.

Liviu Badea and Doina Tilivea. Sparse factorizations of gene expression data guided by binding data. In *PSB*, 2005.

Bart Bakker and Tom Heskes. Task clustering and gating for Bayesian multitask learning. *J. Mach. Learn. Res.*, 4:83–99, 2003.

Onureena Banerjee, Laurent El Ghaoui, Alexandre d'Aspremont, and Georges Natsoulis. Convex optimization techniques for fitting sparse Gaussian graphical models. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 89–96, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2.

Ziv Bar-Joseph, Georg K. Gerber, Tong Ihn Lee, Nicola J. Rinaldi, Jane Y. Yoo, Franois Robert, D.Benjamin Gordon, Ernest Fraenkel, Tommi S. Jaakkola, Richard A. Young, and David K. Gifford. Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, 21(11):13371342, 2003.

Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

Katia Basso, Adam A. Margolin, Gustavo Stolovitzky, Ulf Klein, Riccardo Dalla-Favera, and Andrea Califano. Reverse engineering of regulatory networks in human b cells. *Nature Genetics*, 37(4):382–390, 2005.

L.E. Baum and G.R. Sell. Growth transformations for functions on manifolds. *Pacific J. Math*, 27(2):211–227, 1968.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

Jonathan Baxter. A model for inductive bias learning. *J. Artificial Intelligence Research*, 12:149–198, 2000.

Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *COLT*, 2003.

Benjamin P. Berman, Yutaka Nibu, Barret D. Pfeiffer, Pavel Tomancak, Susan E. Celniker, Michael Levine, Gerald M. Rubin, and Michael B. Eisen. Exploiting transcription factor binding site clustering to identify cis-regulatory modules involved in pattern formation in the Drosophila genome. *Proc. Natl. Acad. Sci. USA*, 99(2):757–62, 2002.

Ashish Bhan, David J. Galas, and Gregory T. Dewey. A duplication growth model of gene expression networks. *Bioinformatics*, 18(11):1486–1493, November 2002.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, 1998.

Wray Buntine. Theory refinement on Bayesian networks. In *UAI*, pages 52–60. Morgan Kaufmann, 1991.

A. J. Butte and I. S. Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pacific Symposium of Biocomputing*, pages 418–429, Children's Hospital Informatics Program, Boston, MA 02115, USA., 2000.

A. J. Cadotte, T.B. DeMarse, P. He, and M. Ding. Causal measures of structure and plasticity in simulated and living neural networks. *PLoS ONE*, 3(10), 2008.

P. Carmona-Saez, R.D. Pascual-Marqui, F. Tirado, J.M. Carazo, and A. Pascual-Montano. Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC bioinformatics*, 7(1):78, 2006.

Rich Caruana. Multi-task learning. *Machine Learning*, 28:41–75, 1997.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge Univ Press, 2006.

David M. Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *J. Mach. Learn. Res.*, 5:1287–1330, 2004. ISSN 1533-7928.

David Maxwell Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

David Maxwell Chickering and Craig Boutilier. Learning equivalence classes of Bayesian-network structures. In *Journal of Machine Learning Research*, pages 150–157. Morgan Kaufmann, 1996.

C. I. Chow, Sexior Member, and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

Jerome T. Connor, R. Douglas Martin, and L. E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5:240–254, 1994.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, October 1992.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, August 1991. ISBN 0471062596.

Djurdje Cvijovicacute and Jacek Klinowski. Taboo search: An approach to the multiple minima problem. *Science*, 267(5198):664–666, February 1995.

G. Czanner, U.T. Eden, S. Wirth, M. Yanike, W.A. Suzuki, and E.N. Brown. Analysis of between-trial and within-trial neural spiking dynamics. *Journal of neurophysiology*, 99(5): 2672, 2008.

J. de La Rocha, B. Doiron, E. Shea-Brown, K. Josi&cacute, and A. Reyes. Correlation between neural spike trains increases with firing rate. *Nature*, 448(7155):802–806, 2007.

Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Comput. Intell.*, 5(3):142–150, 1990. ISSN 0824-7935.

Dennis DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 249–256, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.*, 37(5):1342–1372, 2008. ISSN 0097-5397.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, page 41, 1999.

Inderjit S. Dhillon and Suvrit Sra. Generalized nonnegative matrix approximations with Bregman divergences. In *In: Neural Information Proc. Systems*, pages 283–290, 2005.

C. Ding, T. Li, and M.I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1):45–55, 2008.

David Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289 – 1306, 2006.

Alvin W. Drake. *Fundamentals of Applied Probability Theory*. Mcgraw-Hill College, 1967. ISBN 0070178151.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *J. Mach. Learn. Res.*, 10:2899–2934, 2009.

Miroslav Dudk, Steven J. Phillips, and Robert E. Schapire. Performance guarantees for regularized maximum entropy density estimation. In *Proceedings of the 17th Annual Conference on Computational Learning Theory*, pages 472–486. Springer Verlag, 2004.

D. Dueck, Q. D. Morris, and B. J. Frey. Multi-way clustering of microarray data using probabilistic sparse matrix factorization. *Bioinformatics-Oxford*, 21(1):144, 2005.

James Durbin and Siem J. Koopman. *Time series analysis by state space methods*. Oxford University Press, 2 edition, 2001.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci*, 95(25):14863–14868, December 1998. ISSN 0027-8424.

Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005.

Liming Fan. Structure learning with large sparse undirected graphs and its applications. Technical report, Carnegie Mellon University, 2006.

T. Feng, S. Z. Li, H-Y. Shum, and H. Zhang. Local non-negative matrix factorization as a visual representation. In *ICDL '02: Proceedings of the 2nd International Conference on Development and Learning*, page 178, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1459-6.

Mario A. T. Figueiredo and Anil K. Jain. Bayesian learning of sparse classifiers. In *In 2001 Conference on Computer Vision and Pattern Recognition (CVPR 2001*, pages 35–41. IEEE Press, 2001.

G. D. Forney. The Viterbi algorithms. *Proceedings of the IEEE*, 61(3):268–278, 1973.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. *Learning in Graphical Models*, pages 421–460, 1998.

Nir Friedman and Moises Goldszmidt. *Learning in Graphical Models*, chapter Learning Bayesian networks with local structure, pages 252–262. MIT Press, 1996.

Nir Friedman and Daphne Koller. Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1-2):95–125, January 2003. ISSN 0885-6125.

Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309. Springer-Verlag, 1999a.

Nir Friedman, Iftach Nachman, and Dana Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *UAI*, pages 206–215, 1999b.

R. Ganapathy, G. Rangarajan, and AK Sood. Granger causality and cross recurrence plots in rheochaos. *Physical Review E*, 75(1):16211, 2007.

T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, July 2003. ISSN 1095-9203.

Dan Geiger and David Heckerman. Parameter priors for directed acyclic graphical models and the characterization of several probability distributions. *The Annals of Statistics*, 30 (5):1412–1440, 2002.

Z. Ghahramani. Learning dynamic Bayesian networks. *Lecture Notes in Computer Science*, 1387:168–197, 1998.

Z. Ghahramani and M. Jordan. Mixture models for learning from incomplete data. *MIT Press*, 4:67–85, 1997.

J. Girgis, D. Merrett, S. Kirkland, G. A. S. Metz, V. Verge, and K. Fouad. Reaching training in rats with spinal cord injury promotes plasticity and task specific recovery. *Brain*, 2007.

Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

L. Glass and S. A. Kauffman. The logical analysis of continuous, non-linear biochemical control networks. *Journal of Theoretical Biology*, 39(1):103, 1973.

D. Guillamet, M. Bressan, and J. Vitria. A weighted non-negative matrix factorization for local representations. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1. IEEE Computer Society; 1999, 2001.

J. Guinney, Q. Wu, and S. Mukherjee. *Estimating variable structure and dependence in Multi-task learning via gradients*, 2007. working paper.

Yuhong Guo and Dale Schuurmans. Convex structure learning for Bayesian networks: polynomial feature selection and approximate ordering. In *UAI*, 2006.

M. Gustafsson, M. Hornquist, and A. Lombardi. Large-scale reverse engineering by the lasso. In *ICSB*, 2003. undirect graphical model.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003. ISSN 1533-7928.

J. D. Hamilton. *Time series analysis*. Princeton Univ Press, 1994.

Jing-Dong J. Han, Nicolas Bertin, Tong Hao, Debra S. Goldberg, Gabriel F. Berriz, Lan V. Zhang, Denis Dupuy, Albertha J. Walhout, Michael E. Cusick, Frederick P. Roth, and Marc Vidal. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, 430(6995):88–93, July 2004.

A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Pac Symp Biocomput*, pages 422–433, 2001.

Alexander J. Hartemink, David K. Gifford, Tommi S. Jaakkola, and Richard A. Young. Combining location and expression data for principled discovery of genetic regulatory networks. In *Pac. Symp. Biocomput.*, pages 437–449, 2002.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, September 1995. ISSN 0885-6125.

David Heckerman, David M. Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, October 2000.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.

Katsuhisa Horimoto and Hiroyuki Toh. Statistical estimation of cluster boundaries in gene expression profile data. *Bioinformatics*, 17(12):1143–1151, December 2001.

P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 557–565, 2002.

Patrik O. Hoyer and Peter Dayan. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19:2271–2282, 2003.

T. Ideker, V. Thorsson, J.A. Ranish, R. Christmas, J. Buhler, J.K. Eng, R. Bumgarner, D.R. Goodlett, R. Aebersold, and L. Hood. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 292:929–934, 2001.

J. Ihmels, G. Friedlander, S. Bergmann S, O. Sarig, Y. Ziv Y, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nat Genet.*, 31(4):370–7, 2002.

Su In Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of Markov networks using ℓ_1-regularization. In *NIPS*, 2006.

V. R. Iyer, M. B. Eisen, D. T. Ross, G. Schuler, T. Moore, J. C. Lee, J. M. Trent, L. M. Staudt, J. Hudson, M. S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P. O. Brown. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283 (5398):83–87, January 1999. ISSN 0036-8075.

A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(2):153–158, 2002.

Tony Jebara. Multi-task feature and kernel selection for SVMs. In *ICML*, 2004.

Frederick Jelinek. *Statistical methods for speech recognition*. MIT Press, Cambridge, MA, USA, 1997.

H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, October 2000. ISSN 0028-0836.

Rong Jin, Chris Ding, and Feng Kang. A probabilistic approach for optimizing spectral clustering. In *Proceedings of the Nineteenth Annual Conference on Neural Information Processing Systems (NIPS 2005)*, Cancouver, Canada, Dec 2005.

Rong Jin, Luo Si, Shireesh Srivastava, Zheng Li, and Christina Chan. A knowledge driven regression model for gene expression and microarray analysis. In *EMBS'06*, 2006.

Schafer Juliane and Strimmer Korbinian. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, March 2005. ISSN 1367-4803.

M. Kamiński, M. Ding, W.A. Truccolo, and S.L. Bressler. Evaluating causal relations in neural systems: Granger causality, directed transfer function and statistical assessment of significance. *Biological Cybernetics*, 85(2):145–157, 2001.

Robert E. Kass and Adrian E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

H. Kitano. Computational system biology. *Nature*, 420:206–210, 2002.

Keith Knight and Wenjiang Fu. Asymptotics for lasso-type estimators. *The Annals of Statistics*, 28(5):1356–1378, 2000.

Mikko Koivisto. Exact Bayesian structure discovery in Bayesian networks. *J. of Machine Learning Research*, 5:2004, 2004.

H. Koshino, P. A. Carpenter, N.J. Minshew, V.L. Cherkassky, T. A. Keller, and M. A. Just. Functional connectivity in an fMRI working memory task in high-functioning autism. *Neuroimage*, 24(3):810–821, 2005.

Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *J. Mach. Learn. Res.*, 10:777–801, 2009.

P. Larranaga, M. Poza, Y. Yurramendi, R. H. Murga, and C. M. H. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: a performance analysis of control parameters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9): 912–926, 1996.

D. D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562. MIT Press, 2001.

Su-In Lee, Vassil Chatalbashev, David Vickrey, and Daphne Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML*, 2007.

Fan Li and Yiming Yang. Recovering genetic regulatory networks from micro-array data and location analysis data. *Genome Informatics*, 15(2):131140, 2004.

Z. Li, S. Srivastava, S. Mittal, X. Yang, L. Sheng, and C. Chan. A three stage integrative pathway search framework to identify toxicity relevant genes and pathways. *BMC bioinformatics*, 8(1):202, 2007.

James C. Liao, Riccardo Boscolo, Young-Lyeol Yang, Linh M. Tran, Chiara Sabatti, and Vwani P. Roychowdhury. Network component analysis: Reconstruction of regulatory signals in biological systems. *Proceedings of the National Academy of Sciences of the United States of America*, 100(26):15522–15527, 2003.

Y. Liu, R. Jin, and A. K. Jain. Boostcluster: boosting clustering by pairwise constraints. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 450–459. ACM New York, NY, USA, 2007.

L. Ljung. *System identication: Theory for the user*. Prentice-Hall, 1999.

A. V. Lukashin, M. E. Lukashev, and R. Fuchs. Topology of gene expression networks as revealed by data mining and modeling. *Bioinformatics*, 19(15):1909–1916, October 2003. ISSN 1367-4803.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, July 2008. ISBN 0521865719.

D. Margaritis. *Learning Bayesian networks model structure from data*. PhD thesis, CMU, 2003.

Dimitris Margaritis and Sebastian Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12*, pages 505–511. MIT Press, 1999.

S. J. Martin and R. G. M. Morris. New life in an old idea: the synaptic plasticity and memory hypothesis revisited. *Hippocampus*, 12(5):609–636, 2002.

H. H. McAdams and A. Arkin. Simulation of prokaryotic genetic circuits. *Annu Rev Biophys Biomol Struct*, 27:199–224, 1998. ISSN 1056-8700.

C. Meek, D. M. Chickering, and D. Heckerman. Autoregressive tree models for time-series analysis. In *Proceedings of the Second International SIAM*, pages 229–244. SIAM, 2002.

Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *Ann. Statist.*, 34(3):1436–1462, 2006.

Carl D. Meyer. *Matrix analysis and applied linear algebra*. Society for Industrial Mathematics, 2000.

Andrew Moore and Weng Keen Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In *In Proceedings of the 20th International Conference on Machine Learning (ICML 03)*, pages 552–559. AAAI Press, 2003.

Andrew W. Moore and Mary S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, 1998.

Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, 2002.

Yu Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005. ISSN 0025-5610.

Andrew Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 78, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-5.

R. A. Normann, E. M. Maynard, P. J. Rousche, and D. J. Warren. A neural interface for a cortical vision prosthesis. *Vision Research*, 39(15):2577–2587, 1999.

G. Obozinski, B. Taskar, and M. I. Jordan. Multi-task feature selection. In *ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

Francesco Orabona, Joseph Keshet, and Barbara Caputo. The projectron: a bounded kernel-based perceptron. In *Proceedings of International Conference on Machine Learning*, pages 720–727, 2008.

Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, September 1988. ISBN 1558604790.

Simon Perkins, Kevin Lacker, and James Theiler. Grafting: fast, incremental feature selection by gradient descent in function space. *J. Mach. Learn. Res.*, 3:1333–1356, 2003. ISSN 1533-7928.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393, 1997.

Yitzhak Pilpel, Priya Sudarsanam, and George M. Church. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nat. Genet.*, 29:153–159, 2001.

David Pollard. *Convergence of stochastic process*. Springer-Verlag, 1984.

L.R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *ICML*, 2007.

E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabasi. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, August 2002.

S. Raychaudhuri, J. M. Stuart, and R. B. Altman. Principal components analysis to summarize microarray experiments: application to sporulation time series. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 455–466, 2000. ISSN 1793-5091.

Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 713–719, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5.

Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1989. ISBN 981020311X.

R. W. Robinson. Counting labeled acyclic digraphs. In F. Harary, editor, *New Directions in Graph Theory*. New York: Academic Press, 1973.

Dana Ron, Yoram Singer, and NAFTALI TISHBY. The power of amnesia: Learning probabilistic automata with variable memory length. In *Machine Learning*, pages 117–149, 1996.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65:386–408, 1958.

V. Roth. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15:16–28, 2004.

S. Russel and P. Norvig. *Articial Ingelligence: A Modern Approach*. Prentice-Hall., 1995.

Mehran Sahami. Learning limited dependence Bayesian classifiers. In *In KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338. AAAI Press, 1996.

K. Sameshima and L.A. Baccalá. Using partial directed coherence to describe neuronal ensemble interactions. *Journal of neuroscience methods*, 94(1):93–103, 1999.

R. Schachtner, D. Lutter, P. Knollmuller, A. M. Tome, F. J. Theis, G. Schmitz, M. Stetter, P. G. Vilda, and E. W. Lang. Knowledge-based gene expression classification via matrix factorizations. *Bioinformatics*, 24(15):1688, 2008.

M. Niculescu-Mizil Schmidt and K. A. Murphy. Learning graphical model structure using L1-regularization paths. In *AAAI*, 2007.

Bernhard Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press, 2002.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2): 461–464, 1978.

E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nat Genet*, 34(2):166–176, June 2003. ISSN 1061-4036.

A. K. Seth. Causal networks in simulated neural systems. *Cognitive Neurodynamics*, 2(1): 49–64, 2008.

Fei Sha, Yuanqing Lin, Lawrence K. Saul, and Daniel D. Lee. Multiplicative updates for nonnegative quadratic programming. *Neural Comput.*, 19(8):2004–2031, 2007. ISSN 0899-7667.

Farial Shahnaz, Michael W. Berry, V. Paul Pauca, and Robert J. Plemmons. Document clustering using nonnegative matrix factorization. *Inf. Process. Manage.*, 42(2):373–386, 2006. ISSN 0306-4573.

Ajit Singh and Andrew Moore. Finding optimal Bayesian networks by dynamic programming. Technical Report CMU-CALD-05-106, Carnegie Mellon University, 2005.

Anne V. Smith, Jing Yu, Tom V. Smulders, Alexander J. Hartemink, and Erich D. Jarvis. Computational inference of neural information flow networks. *PLoS Computational Biology*, 2(11):1436–1449, November 2006.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *J. Mach. Learn. Res.*, 7:1531–1565, 2006.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9(12): 3273–3297, December 1998. ISSN 1059-1524.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction,and Search*. The MIT Press, 2nd edition, 2001.

Henning Sprekeler, Christian Michaelis, and Laurenz Wiskott. Slowness: an objective for spike-timing-dependent plasticity? *PLoS computational biology*, 3(6), June 2007.

Nathan Srebro. Maximum likelihood bounded tree-width Markov networks. In *Artificial Intelligence*, pages 504–511, 2001.

Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

S. Srivastava and C. Chan. Hydrogen peroxide and hydroxyl radicals mediate palmitate-induced cytotoxicity to hepatoma cells: relation to mitochondrial permeability transition. *Free Radical Research*, 41(1):38–49, 2007.

Marie Szafranski, Yves Grandvalet, and Pierre Morizet-Mahoudeaux. Hierarchical penalization. In *NIPS*, 2007.

Pablo Tamayo, Donna Slonim, Jill Mesirov, Qing Zhu, Sutisak Kitareewan, Ethan Dmitrovsky, Eric S. Lander, and Todd R. Golub. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *PNAS*, 96(6):2907–2912, 1999.

S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nat Genet*, 22(3):281–285, July 1999. ISSN 1061-4036.

Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2005.

R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 63:411–423, 2001.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

Hiroyuki Toh and Katsuhisa Horimoto. Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling. *Bioinformatics*, 18(2):287–297, 2002.

P. Törönen, M. Kolehmainen, G. Wong, and E. Castren. Analysis of gene expression data using self-organizing maps. *FEBS letters*, 451(2):142–146, 1999.

Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, pages 762–769, 2004.

W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology*, 93(2):1074, 2005.

Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.*, 65(1):31–78, 2006. ISSN 0885-6125.

N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. In *NIPS*, 2003.

V. Vapnik, E. Levin, and Y. L. Cun. Measuring the vc-dimension of a learning machine. *Neural Computation*, 6(5):851–876, 1994.

S. Varma and R. Simon. Iterative class discovery and feature selection using minimal spanning trees. *BMC bioinformatics*, 5(1):126, 2004.

Martin J. Wainwright, Pradeep Ravikumar, and John D. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In *NIPS*, pages 1465–1472. MIT Press, 2006. ISBN 0-262-19568-2.

M. K. Warmuth and D. Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In *Advances in Neural Information Processing Systems 19 (NIPS 06)*, 2006.

D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393 (6684):440–442, June 1998. ISSN 0028-0836.

Kilian Q. Weinberger, Benjamin D. Packer, and Lawrence K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 381–388, 2005.

Xiling Wen, Stefanie Fuhrman, George S. Michaels, Daniel B. Carr, Susan Smith, Jeffery L. Barker, and Roland Somogyi. Large-scale temporal gene expression mapping of central nervous system development. *Proceedings of the National Academy of Sciences*, 95(1): 334–339, January 1998.

A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. von Rohr, L. Thiele, E. Zitzler, W. Gruissem, and P. Bühlmann. Sparse graphical Gaussian modeling of the isoprenoid gene network in Arabidopsis thaliana. *Genome Biol*, 5(11), 2004. ISSN 1465-6914.

Ransom Winder, Carlos R. Cortes, James A. Reggia, and M. A. Tagamets. Functional connectivity in fMRI: A modeling approach for estimation and for relating to local circuits. *NeuroImage*, 34(3):1093 – 1107, 2007. ISSN 1053-8119.

K. D. Wise, D. J. Anderson, J. F. Hetke, D. R. Kipke, and K. Najafi. Wireless implantable microsystems: high-density electronic interfaces to the nervous system. *Proceedings of the IEEE*, 92(1):76–97, 2004.

Y. Xiang, S. K. M. Wong, and N. Cercone. A "microscopic" study of minimum entropy search in learning decomposable Markov networks. *Mach. Learn.*, 26(1):65–92, 1997. ISSN 0885-6125.

Tao Xiong, Jinbo Bi, Bharat Rao, and Vladimir Cherkassky. Probabilistic joint feature selection for multi-task learning. In *SDM*, 2007.

Zenglin Xu, Rong Jin, Irwin King, and Michael R. Lyu. An extended level method for multiple kernel learning. In *NIPS*, 2008.

Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. Fourteenth Internation Conference on Machine Learning*, pages 412–420, 1997.

M. K. Yeung, J. Tegnér, and J. J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *Proc Natl Acad Sci U S A*, 99(9):6163–6168, April 2002. ISSN 0027-8424.

Jing Yu, V. Anne Smith, Paul P. Wang, Alexander J. Hartemink, and Erich Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20:35943603, 2004.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68:49–67, 2005.

Xuedong Zhang and Laurel H. Carney. Response properties of an integrate-and-fire model that receives subthreshold inputs. *Neural Comput.*, 17(12):2571–2601, 2005. ISSN 0899-7667.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, 37:3468–3497, 2009.

Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. In *Neural Information Processing Systems*. MIT Press, 2003.