
Coregulation Analysis via Spectra of a Hypercube

Sergey M. Plis*
Computer Science Dept.
University of New Mexico
Albuquerque, NM 87131

Vamsi K. Potluru

Sushmita Roy
Terran Lane

Abstract

Traditional clustering treats data as instances with sets of attributes and commonly uses pairwise criteria. Graphical models, on the other hand, treat the data as random variables (attributes) with respective samples and allow multivariate interaction among these variables. It is common in graphical model learning to sacrifice multivariate interactions for the sake of a detailed (in)dependence structure. We present an alternative approach. We partition random variables into coregulated clusters focusing on arbitrary multivariate relations and ignoring their detailed structure. Multivariate interactions among groups of random variables are hard to compute and even the best estimators are not very fast. To address this, we show that the problem can be cast in the form suitable to the recently proposed framework for smooth function approximation over a hypercube. This formulation allows us to approximate the multiway clustering objective with a few samples, thus providing runtime benefits. We also improve the running time of the original approximation framework, and demonstrate the new clustering framework using a stochastic global search algorithm.

1 Introduction

Unsupervised analysis of data can be roughly split into two major groups: one treats the data as instances with corresponding sets of attribute values, and the other treats the data as random variables with their respective data samples. The most prominent analysis technique in the first group is clustering [2], whereas the second group is dominated by graphical models [9].

Clustering is concerned with partitioning data instances according to similarities in their attributes, which is prevalently done using pairwise similarity criteria [2]. Graphical model learning (structure search) involves discovery of multivariate statistical (in)dependence structure among the complete set of random variables (attributes).

Due to complexity, the algorithms for graphical model learning often step away from searching for multivariate dependencies and use pairwise criteria [4, 6, 13]. The problem is even more difficult when hidden variables are involved [3, 5]. When successful, structure search produces a detailed description of random variable interactions.

In this paper, we present an alternative approach to the structure search unsupervised learning problem: coregulation analysis (CA). Rather than restricting the number of variables in a relationship, or the form of the statistical model among random variables, we instead relax the requirement of a detailed structural description. That is, instead of searching for the fine-grained (in)dependence structure, we cluster random variables in groups, similar to the partitioning approach of conventional clustering. We abandon the fine dependencies of the graphical structure but allow arbitrarily multivariate and complex statistical relationships within our random variable partitions.

Our contributions include developing an objective function for random variable partitioning, presenting a way to compute this objective and formulating a global stochastic search algorithm to optimize the objective.

Since evaluation of the objective is computationally expensive, it is difficult to use in a practical search procedure. We overcome the difficulty by developing a transformation of the problem into the function approximation framework of Yackley et al. [21]. The resulting algorithm additionally does not require manual setting of the exact number of clusters, but rather an upper bound on this number. We further overcome

*pliz@cs.unm.edu

a known limitation of this framework that it becomes slow when number of random variables and clusters is large. We present results that significantly improve the speed of this approximation framework.

2 Partitioning objective

Given a set of n random variables $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ the task is to partition it into $k \leq m$ clusters $\{\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_k\}$. The number of clusters needs to be found automatically given the manually set upper bound m , but there are well known ways to estimate m such as penalized search [2] or Chinese Restaurant Process-based Bayesian estimates [20]. The assignment of random variables to clusters should satisfy the following:

1. The assignment forms a partition of \mathbf{X} .
2. Variables in a cluster are maximally dependent.
3. Clusters are maximally independent.

As described, the partitioning problem generalizes several possible scenarios of interdependence among random variables. Example scenarios using a directed graphical model are shown in Figure 1. Our assumptions are illustrated in Figure 1a, where random variables are grouped into 3 tight clusters with high interconnectivity within them and weak relations across them. Figure 1b shows a scenario where each cluster is regulated by a single latent variable. This is similar to assumptions of independent component analysis (ICA) [2], but we assume partition instead of a cover in ICA. Figure 1c shows the most general case, where interactions among variables within each cluster are governed by a complex unknown network of hidden variables. In all cases, variables in a single cluster are coregulated either by direct interactions among each other or by a hidden isolated variable/network. Hence we call the partition process Coregulation Analysis (CA).

If we denote by X_i^C an i^{th} random variable assigned to cluster C , and by \mathcal{D} the criterion used to evaluate statistical dependence (\mathcal{D} , for example, can be mutual information or multiinformation (see Section 3)), we can express the second condition in terms of the dependence criterion:

$$\max \mathcal{J}_2 = \sum_{i=1}^k \mathcal{D}(\mathbf{C}_i) = \sum_{i=1}^k \mathcal{D}(X_1^{\mathbf{C}_i}, \dots, X_{|\mathbf{C}_i|}^{\mathbf{C}_i}) \quad (1)$$

The third condition of the maximal independence of the clusters can be guaranteed by minimizing the dependence criterion among them:

$$\min \mathcal{J}_1 = \mathcal{D}(\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_m) \quad (2)$$

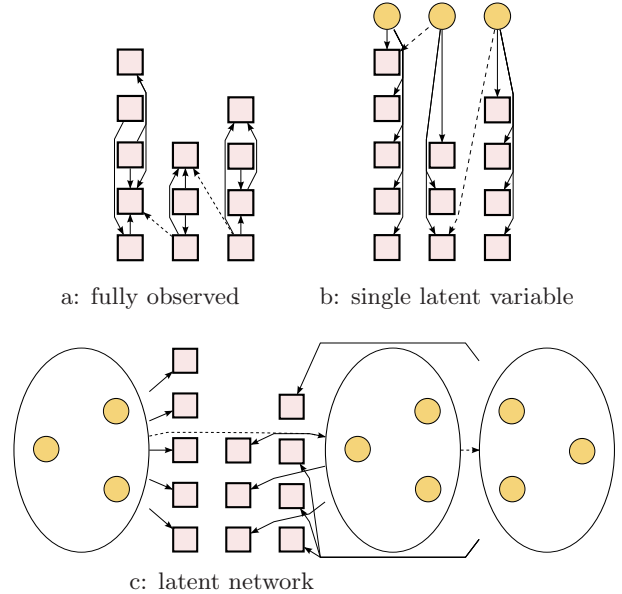


Figure 1: Scenarios of interdependence among random variables which are covered by our model. A directed graphical model is picked as an example only and it may as well be an undirected or a mixed type relationship. Solid and dashed arrows denote strong and weak statistical interactions respectively. Squares are the observed random variables and circles are the hidden variables.

The cumulative objective function for the proposed clustering of random variables is then

$$\min \mathcal{J} = \frac{\mathcal{J}_1^\alpha}{\mathcal{J}_2^{1-\alpha}}, \quad (3)$$

where $0 \leq \alpha \leq 1$ controls relative importance of intra-cluster dependence over inter-cluster independence. The first condition can be forced to be true by limiting the space of possible solutions while optimizing (3).

3 Estimating dependence

There are several options for the dependence criterion \mathcal{D} . Based on problem-dependent considerations any of them can be selected. A number of such criteria has been proposed; a full evaluation of their relative merits is beyond the scope of this paper.

A possible choice is the multiinformation (MI) function of [18]. Kullback-Leibler (KL) divergence [11] also is a plausible criterion for our purposes. Another possible choice is the Hilbert Schmidt Information Criterion (HSIC) [7].

If we use definition of multiinformation $\mathcal{D}_{\mathcal{I}}(X_1, \dots, X_n)$ presented in [17, eq. 1.12] for discrete variables, and extended to continuous

random variables as:

$$\int_{x_1, \dots, x_n} p(x_1, \dots, x_n) \log \frac{p(x_1, \dots, x_n)}{p(x_1) \dots p(x_n)} \dots dx_n, \quad (4)$$

then we can compute it as KL divergence $\mathcal{D}_{\mathcal{KL}}[p(x_1, \dots, x_n)|p(x_1) \dots p(x_n)]$. Furthermore, we employ Condition 1.2.14 from [17] (see also [10, eq. A3]):

$$\mathcal{D}_{\mathcal{I}}(X, Y, Z) = \mathcal{D}_{\mathcal{I}}(X, Y) + \mathcal{D}_{\mathcal{I}}((X, Y), Z), \quad (5)$$

where the second term on the right hand side denotes mutual information between a vector valued random variable $(X \times Y)$ and Z . With it we are able to compute \mathcal{J}_1 and \mathcal{J}_2 recursively:

$$\mathcal{J}_1 = \mathcal{D}_{\mathcal{I}}(\mathbf{C}_1, \mathbf{C}_2) + \dots + \mathcal{D}_{\mathcal{I}}((\mathbf{C}_1, \dots, \mathbf{C}_{m-1}), \mathbf{C}_m) \quad (6)$$

$$\mathcal{J}_2 = \sum_{i=1}^k \mathcal{D}_{\mathcal{I}}(X_1^{\mathbf{C}_i}, X_2^{\mathbf{C}_i}) + \dots + \mathcal{D}_{\mathcal{I}}((\dots), X_{|\mathbf{C}_i|}^{\mathbf{C}_i}) \quad (7)$$

Information theoretic criteria are difficult to estimate and require complicated bias correction terms in higher dimensions [15]. Fortunately, for our purpose, we are more interested in the discrepancy among random variables (their clusters) than in the exact MI value. A recent paper [16] demonstrates suitability of the k -nearest neighbor mutual information (MI) estimator [10] to measuring discrepancy in random variables, together with proving almost sure convergence for k -nn type estimators of MI, KL divergence and differential entropy. This new result allows us to use equations (6) and (7) directly.

4 Approximating objective

Estimation of the objective (3) is computationally expensive due to the complexity dependent on the number of variables, number of clusters and the sample size. In this section we show a way to overcome this problem by computing only a small number of values y_i of the objective and approximating the rest.

4.1 Framework

Yackley et al. [21] have introduced a framework for approximating Bayesian Network score using a meta-graph kernel. They present fast computation of arbitrary elements of eigenvectors of the Laplacian of the Hamming cube. This leads to an efficient solution of the following minimization problem:

$$\hat{\mathcal{J}} = \arg \min_{\mathcal{J}} \frac{1}{N} \sum_{i=1}^N \|\mathcal{J}(v_i) - y_i\|^2 + c\mathcal{J}^T L^l \mathcal{J}, \quad (8)$$

where \mathcal{J} is a function we use to approximate the data y_i (computed values of the objective (3) in our case), L is the Laplacian of the graph describing the domain on which \mathcal{J} is defined, N is the number of computed values y_i , c and l are regularization parameters, and v_i is a vertex of the hypercube.

The problem is formulated and solved within the Reproducible Kernel Hilbert Space framework [19, 21]. The authors show that one needs to compute values of the kernel matrix only at those columns where the data y_i is available and only at those rows where we need to approximate the value. For this, elements of the kernel are computed using:

$$(K_1)_{ij} = \sum_{k=1}^r \left(\frac{s}{2^k}\right)^l \sum_{|\mathbf{h}|_1=k} (-1)^{\langle \mathbf{b}_i \oplus \mathbf{b}_j, \mathbf{h} \rangle}. \quad (9)$$

\oplus denotes binary exclusive or, $|\cdot|_1$ denotes the number of nonzero elements in the binary expansion of the argument, $\langle \cdot \rangle$ denotes vector inner product, l is the smoothing constant, r is the order of approximation (similar to the degree of a polynomial), s is the length of the binary string specific for each problem, and \mathbf{b}_i and \mathbf{b}_j denote binary strings that are placed in the i^{th} and j^{th} vertices of the hypercube respectively.

4.2 Hypercube

Although the framework of [21] is formulated for Bayesian Network structure search scoring, it is more general and applies to any problem of approximating smooth functions over a hypercube. To use this approximation framework, we need to represent the problem of clustering random variables defined in Section 2 as a hypercube graph.

First we show that the partitioning problem of Section 2 represents a subset of vertices on a hypercube. This will allow us to define smooth functions approximating our objective \mathcal{J} on the vertices of the hypercube and use the approximation only on the subset that represents partitions. For that, we demonstrate that vertices of the hypercube representing partitions lie on a manifold. Together with a distance metric on the manifold, it leads to a setup well suited to global stochastic optimization.

At the risk of being pedantic, we prove the following simple lemma. It serves the purpose making our further development clearer.

Lemma 1. *In the case when m is the upper bound on the number of clusters, and overlapping clusters as well as empty clusters (including all empty simultaneously) are allowed¹, the space of possible assignments forms the Hamming cube.*

¹A relaxation of the partition condition of Section 2

Proof. Let us represent a single assignment of a variable to a cluster by a single bit in a binary vector of length n representing that cluster. Thus for $k \leq m$ clusters and n variables all possible assignments can be represented by a binary number of mn bits, for which there are 2^{mn} possible assignments, These assignments form the vertex set of a hypercube in \mathcal{R}^{mn} . Two vertices from this set are connected by an edge if their binary representations differ in only a single bit. This results in the Hamming cube. \square

If we think of binary numbers that represent the vertices of the hypercube as vectors \mathbf{b} in \mathcal{R}^{mn} , then we can test whether a given vector represents a partition. We do it using the following $mn \times mn$ oblique projection matrix:

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_1 & \dots & \mathbf{I}_m \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \end{bmatrix}, \quad (10)$$

where \mathbf{I}_i is an $n \times n$ identity matrix and the rest of \mathbf{P} is zero. Partitions satisfy the following equation:

$$\mathbf{P}\mathbf{b} = [\mathbf{1}^T \ 0 \ \dots]^T, \quad (11)$$

where $\mathbf{1}$ is a vector of ones of length n .

The set of all vertices of the cube is substantially larger than the set of vertices denoting partitions. If we attempt to minimize \mathcal{J} over the entire Hamming cube, then we will reject many vertices before reaching one that is a valid partition. However, the set of vertices that we examine can be greatly decreased with the help of the following lemma.

Lemma 2. *The vertices of the Hamming cube of Lemma 1 that represent partitions are located on the intersection of a hyperplane and a hypersphere, i.e. a hypersphere of a lower dimension.*

Proof. All binary vectors that satisfy condition (11) belong to the same hyperplane $\mathbf{w}^T \mathbf{b} + c = 0$ in the \mathcal{R}^{mn} space containing the Hamming cube. In order to show this, we first represent \mathbf{b} as a $m \times n$ matrix (this is only for illustration, we still use \mathbf{b} and \mathbf{w} as vectors):

$$\begin{array}{c} n \\ 0000000000 \cdots 0000000000 \\ 0000000000 \cdots 0000000000 \\ \dots \\ m \quad \dots \\ 0000000000 \cdots 0000000000 \end{array} \quad (12)$$

Since a variable in a partition can only be assigned to a single cluster, each column in this matrix has a single non-zero bit.

Construct the normal vector \mathbf{w} as follows: for each even bit in each of the rows of the matrix (12) it is -1,

and for each odd bit it is 1. If again \mathbf{w} is represented as a $m \times n$ matrix, this means that elements of each odd column of this matrix are set to 1, and elements of each even column are set to -1 (-1 and 1 are replaced by - and + respectively):

$$\begin{array}{c} n \\ + - + - + - + \cdots - + - + - + - \\ + - + - + - + \cdots - + - + - + - \\ m \quad \dots \\ + - + - + - + \cdots - + - + - + - \end{array} \quad (13)$$

the inner product with \mathbf{w} constructed in such manner turns into zero all binary vectors satisfying (11) and having even n , because for each positive element in the sum $\mathbf{w}^T \mathbf{b}$ there always be a negative one. In the case of the odd n , number of positive components in the sum will always be bigger by 1, hence the bias $c = -1$, otherwise it is 0.

However the hyperplane $\mathbf{w}^T \mathbf{b} + c = 0$ contains more elements than those we are interested in. We can easily reduce their number by observing that all vectors representing valid partitions have equal length of \sqrt{n} (L_2 -norm). This means that the elements are located on the intersection of a mn -dimensional hypersphere of radius \sqrt{n} and the hyperplane. This intersection is itself, by definition, a hypersphere. \square

The important result of Lemma 2 is defining the manifold over which the search for the best clustering is happening. This gives us measures of how far we are from the current estimate of a minimum of \mathcal{J} . This is important for stochastic global search algorithms, e.g. simulated annealing [12], when step size goes beyond just direct neighbors of the current vertex in the hypercube.

5 Stochastic search

In this section we describe a greedy algorithm for stochastic search of a minimum of \mathcal{J} and define all the elements of the simulated annealing algorithm.

For a stochastic search, we require a probability distribution on the domain of the objective function. Using Lemma 2, we choose the von Mises-Fisher (vMF) distribution [1], which is defined on (and restricted to) the hypersphere, allowing us to sample only from the relevant subsets of the hypercube.

A d -variate vMF distribution of a d -dimensional vector \mathbf{x} is defined as:

$$f(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\kappa}) = c_d(\boldsymbol{\kappa}) e^{\boldsymbol{\kappa} \boldsymbol{\mu}^T \mathbf{x}}, \quad (14)$$

where $\|\boldsymbol{\mu}\| = 1$, $\boldsymbol{\kappa} \geq 0$ and $d \geq 2$. The normalization

constant is defined as:

$$c_d(\boldsymbol{\kappa}) = \frac{\boldsymbol{\kappa}^{d/2-1}}{(2\pi)^{d/2} \mathbf{I}_{d/2-1}(\boldsymbol{\kappa})}, \quad (15)$$

where $\mathbf{I}_i(\cdot)$ is the r^{th} order modified Bessel function of the first kind and $\boldsymbol{\kappa}$ is called concentration parameter.

Although the vMF distribution is on the unit sphere, it is easily adapted to our task of drawing samples on the surface of an arbitrary sphere by shifting the center of the sphere and re-normalizing the samples drawn from the distribution.

A greedy search to optimize our objective is described in Algorithm 1. Steps 6 and 8 can be computed using

Algorithm 1 greedy search

- 1: Pre-compute the values of objective function
 - 2: Set the concentration parameter: $\boldsymbol{\kappa}$
 - 3: Randomly select initial node \mathbf{v}_0 on the hypercube
 - 4: Declare it the best estimate $\mathbf{v}_b = \mathbf{v}_0$
 - 5: **repeat**
 - 6: Sample N partitions from the vMF distribution around \mathbf{v}_i .
 - 7: **repeat**
 - 8: Sample a vector from vMF
 - 9: Find the closest binary vector (in L_2 sense)
 - 10: Accept if it is a partition, reject otherwise
 - 11: **until** N points are collected
 - 12: Compute the value of the objective on these partitions using the framework of Section 4.
 - 13: Select the partition with the minimal value of the objective as the current estimate:
 $\mathbf{v}_b = \arg \min(\{\mathcal{J}(\mathbf{v}_i)\}_{i=0}^N, \mathcal{J}(\mathbf{v}_b))$.
 - 14: **until** convergence
-

the algorithm from [1]. Step 9 is done by thresholding the elements of sampled vector to 0 if the magnitude is less than 0.5 or to one otherwise. The test in step 10 can in general, be performed by condition (11), which can be computed in time linear in the number of elements in \mathbf{b} . However for the case when all vectors are on the described hypersphere, the running time can be improved by observing that for partitions the following equality holds:

$$\left| \bigoplus_{i=1}^k \mathbf{b}_i \right|_1 = n, \quad (16)$$

where \mathbf{b}_i are rows of the matrix representation of \mathbf{b} as in (12). For the vertices located on the hypersphere (i.e. exactly n bits are active) this condition holds only for partitions (i.e. only single bit is active per column of matrix (12)). Due to single tick hardware XOR and POPCNT (count number of active bits in a word) instructions, the test by (16) can be computed

about 32 or 64 times faster, depending on the length of machine word.

With everything defined so far we have all the required pieces to construct a global stochastic search algorithm: simulated annealing. The energy function is our objective from (3). The state space is the hypersphere containing a subset of the vertices of the Hamming cube. And with the vMF distribution the candidate generator procedure is also defined.

In summary, our objective \mathcal{J} can be efficiently approximated using the complete Hamming cube. However, our greedy search of \mathcal{J} and the required datapoints to estimate \mathcal{J} need to be computed only on the intersection hypersphere. This provides computational advantages both for selecting a partition for the next step and thereafter, computing the approximation for the next step.

6 Improving efficiency

6.1 Rejections

Unfortunately, there is a caveat in step 10 of Algorithm 1. The number of vertices that needs to be rejected may be very large, slowing the procedure. In the following we bound the total number of vertices on the hypersphere and compare it with the number of vertices that represent partitions.

The matrix in (12) together with restriction for partitions (only a single bit is set per column) makes it clear that there are m^n valid vertices on the hypersphere. Note that this is the number of total binary representations of valid partitions of n random variables into up to m clusters. There is substantial symmetry in these numbers, unlike the number of total unique partitions of a set, which is expressed by the Stirling numbers of the second kind.

The total number of vertices on the hypercube can be computed based on two following conditions (WLOG, we restrict our attention to the case of even n):

1. Number of nonzero bits in the binary numbers of these vertices is n
2. Vertices belong to the hyperplane described by the normal vector (13).

These conditions limit binary vectors to have particular structure: n set bits must be split in two groups with equal number of elements, one of the groups belongs to even columns of (12), the other to the odd.

Total number of the vertices of the hypercube that

belong to the intersection hypersphere is:

$$\begin{aligned} \left(\frac{k \frac{n}{2}}{\frac{n}{2}}\right)^2 &\leq \left(\frac{(k \frac{n}{2})^{k \frac{n}{2}}}{\left(\frac{n}{2}\right)^{\frac{n}{2}} \left(k \frac{n}{2} - \frac{n}{2}\right)^{(k-1) \frac{n}{2}}}\right)^2 \\ \left(\frac{k \frac{n}{2}}{\frac{n}{2}}\right)^2 &\leq \left(\frac{(k^k)^{\frac{n}{2}} \left(\frac{n}{2}\right)^{(k-1) \frac{n}{2}}}{(k-1)^{(k-1) \frac{n}{2}} \left(\frac{n}{2}\right)^{(k-1) \frac{n}{2}}}\right)^2 \\ \left(\frac{k \frac{n}{2}}{\frac{n}{2}}\right)^2 &\leq \left(\left(\frac{k}{k-1}\right)^{k-1}\right)^n k^n \end{aligned}$$

Observing that $\lim_{k \rightarrow \infty} \left(\frac{k}{k-1}\right)^{k-1} = e$ we arrive at the following bound to the total number of vertices of the hypercube that are located on the intersection hypersphere:

$$\left(\frac{k \frac{n}{2}}{\frac{n}{2}}\right)^2 \leq e^n k^n. \quad (17)$$

This means that total number of possible rejections is about e^n times bigger than the number of vertices representing partitions.

The following lemma allows us to decrease the number of rejections to zero.

Lemma 3. *If the maximum number of clusters, m , is a power of two, then all possible partitions (including their permutations) in clusters up to m form a hypercube.*

Proof. When m is power of two, we can always write the total number of partitions (including their permutations) as:

$$m^n = 2^{n \log_2 m}, \quad (18)$$

which gives us the vertex set of the hypercube.

Figure 2 again uses the matrix representation of \mathbf{b} to demonstrate the compression in the number of required bits when encoding a partitions with m a power of two. The shaded area shows that unsurprisingly only $\log_2 m$ bits are needed to express assignment of a random variable to one of m clusters.

		n					
4	$\log_2 4$	1	1	1	...	0	0
		0	1	0	...	1	1
		0	0	0	...	0	0
		0	0	0	...	0	0

Figure 2: Representing cluster assignments for 4 clusters with only $2n$ bits

The topology of the hypercube is again defined by a single bit flips of the compressed encoding forming a Hamming cube. \square

Although, the assumption of upper bound on the number of clusters to be a power of two may seem restrictive, it can be used for search of partitions into any number of clusters $k \leq m$. However, the development of Section 4.2 is more general as it can deal with any m and advantageous for the cases of large number of random variables, when restriction of m to powers of two can significantly increase the size of the search space.

6.2 Kernel

In the smooth function approximation framework [21] a crucial part plays computation of an $(ij)^{th}$ element of the kernel matrix K_1 (9). The authors have proposed a recursive formulation that allows computing the element in feasible time. Here we show that an element of K_1 can be computed very efficiently in closed form.

We define $S_k(\mathbf{b}_i, \mathbf{b}_j) = \sum_{|\mathbf{h}|_1=k} (-1)^{(\mathbf{b}_i \oplus \mathbf{b}_j, \mathbf{h})}$ following [21] to clarify the notation in (9). The following theorem and its corollary bring substantial speed up to the originally introduced procedure of computing S_k .

Theorem 1. *S_k depends only on a single parameter $|\mathbf{b}_i \oplus \mathbf{b}_j|_1 = p$ and can be computed in a closed form:*

$$S_k(p) = \sum_{i=0}^p (-1)^i \binom{p}{i} \binom{s-p}{k-i}$$

Proof. S_k denotes the difference between two quantities: the number of times the inner product in the power is even, and the number of times it is odd. The number of cases when not a single active bit in $\mathbf{b}_i \oplus \mathbf{b}_j$ coincides with an active bit in l contributes to the first quantity. The number of cases when there is only one coincidence contributes to the second quantity (or negatively to S_k). Continuing further, the times there is an even number of coincidences contributes positively to S_k whereas the times when the number of coincidences is odd contributes negatively. Observe, that for a fixed k the number of coincidences does not depend on position of active bits in the binary representation. It only depends on $|\mathbf{b}_i \oplus \mathbf{b}_j|_1 = p$. Hence:

$$S_k(\mathbf{b}_i, \mathbf{b}_j) = S_k(p) \quad (19)$$

Given the above observation, expressions for the number of coincidences from 0 to p , which we denote by

C_i , can be summarized as:

$$\begin{aligned} C_0 &= \binom{p}{0} \binom{s-p}{k-0} \\ C_1 &= \binom{p}{1} \binom{s-p}{k-1} \\ &\vdots \\ C_x &= \binom{p}{x} \binom{s-p}{k-x}. \end{aligned} \quad (20)$$

As we have already observed, when i is even C_i contributes positively to S_k and negatively when i is odd. This can be summarized in:

$$S_k(p) = \sum_{i=0}^p (-1)^i C_i \quad (21)$$

□

Corollary 1. *The RKHS kernel of (9) is a radial basis kernel and can be computed in the closed form for each of its elements.*

Proof. We only sketch the proof here. At first we reorganize equation (9) using Theorem 1:

$$(K_1)_{ij} = \frac{s^l}{2^l} \sum_{k=1}^r \frac{1}{k^l} S_k \quad (22)$$

The only part in the expression that depends on ij is S_k , which is already proved to depend only on the distance between \mathbf{b}_i and \mathbf{b}_j . Follows, the kernel is an RBF kernel.

Computing (22) directly is not very efficient because some parts of the calculation are repeated several times. Regrouping its elements results in expression:

$$(K_1)_{ij} = \frac{s^l}{2^l} \sum_{i=0}^r \binom{s-p}{i} \sum_{j=0}^{r-i} \frac{1}{(j+i)^l} (-1)^j \binom{p}{j}.$$

The expression holds setting $1/(j+i)^m$ to zero when $j+i=0$. □

All bounds checking in expressions of Theorem 1 and Corollary 1 is done automatically when binomial coefficient $\binom{s}{k}$ is set to zero for $k < 0$ or $k > s$ (the standard definition). Note, this can be used for efficient implementation in order to avoid unnecessary computation. The rightmost binomial coefficients in the expression of Corollary 1 are evaluated only once and cached for further use. Further improvement in the running time is achieved by using the binomial coefficients from the previous step to compute those of

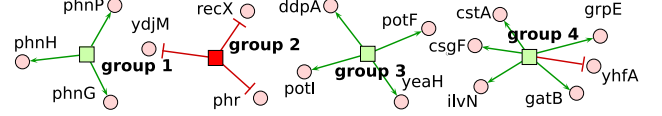


Figure 3: Simulation graph from which the data was generated. Circles are the random variables. Names of genes are also shown.

the next one. However, this may even not be necessary since we can precompute values of the kernel for all possible cases, which is only $n \log_2 m$ when m is a power of two, or mn in the general case. In experiments we perform this computation simultaneously for all possible kernel values, which only takes $O(r^2 mn)$ time, where r is typically less than 10.

7 Experimental evaluation

In order to test the introduced objective and evaluate the search algorithm we use synthetic datasets with known ground truth. The simulated datasets were generated by a gene regulatory network simulator using differential equations for describing gene expression dynamics [8, 14]. The simulator models different regulatory relationships among regulator genes and target genes to generate expression data resembling those from real-world networks. Source graph is shown in Figure 3. The regulator genes are the hidden nodes which determine the groupings of the target genes. These target genes are represented as the observed random variables samples. We generated 1000 samples for each random variable.

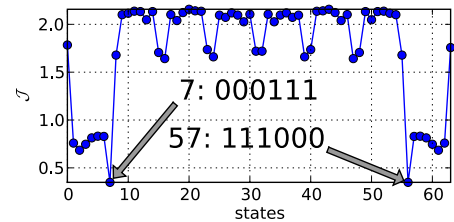


Figure 4: Values of the objective function \mathcal{J} at all states in the simulated gene regulation system with 6 genes and 2 clusters.

Figure 4 shows values of \mathcal{J} from (3) computed at all possible states of a system comprised of group 1 and 2. k in the k -nn estimator is set to the half of the number of samples as in [16]. There are 64 states representing valid partitions in the system. Each can be coded by a binary string of length 6 (Lemma 3). Since \mathcal{J} is undefined at the states assigning all random variables to a single cluster, we have approximated its values at 0 and 63 using order of approximation $r = 5$. As

expected, \mathcal{J} is minimal at the states that split the genes into clusters correctly and there are two of these as seen from Figure 4.

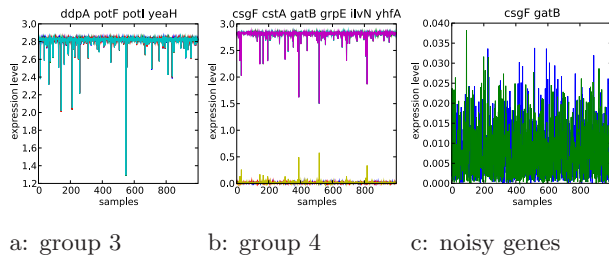


Figure 5: Simulated data for genes from groups 3 and 4 with their gene names listed as plot titles. Data for two noisy genes from group 4 are shown in Figure 5c

For random variables of groups 3 and 4 the resulting solution placed 2 genes belonging to group 4 into group 3. However, the value of \mathcal{J} was zero both at the correct partition and the partition returned by the algorithm. The data used in this experiment is shown in Figure 5. The misplaced genes have expression signal close to random noise (Figure 5c), which explains the invariance to their assignment to clusters.

8 Conclusions

We have described the coregulation analysis framework for capturing arbitrarily complex, multi-way interactions among random variables. We have developed an objective function for solving this problem and presented a way to compute it using an MI estimator. We address the computational overhead of optimizing this function, by using an approximation framework that requires a small number of precomputed values. Additionally, we improved the speed of the approximation framework. Our overall framework is modular and the search algorithm and the objective function can be replaced according to the problem domain. Although the data we are interesting in is continuous, with an appropriate similarity criterion \mathcal{D} it is also valid for discrete and mixed data types. Future work includes further empirical evaluation and extending the model to the case of overlapping clusters.

References

[1] I. Dhillon and S. Sra. Modeling data using directional distributions, 2003.
 [2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.
 [3] G. Elidan and N. Friedman. Learning hidden variable networks: The information bottleneck approach. *J. Mach. Learn. Res.*, 6:81–127, 2005.

[4] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(23):131–163, nov 1997.
 [5] N. Friedman. The Bayesian structural EM algorithm. In *UAI*, pages 129–138, 1998.
 [6] N. Friedman, I. Nachman, and D. Peer. Learning bayesian network structure from massive datasets: The ”sparse candidate” algorithm. In *UAI*, pages 206–215, 1999.
 [7] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schlkopf, and A. J. Smola. A kernel method for the two-sample-problem. In B. Schlkopf, J. C. Platt, and T. Hoffman, editors, *NIPS*, pages 513–520. MIT Press, 2006.
 [8] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. Copasi - a complex pathway simulator. *Bioinformatics*, 22:3067–3074, 2006.
 [9] M. I. Jordan. *Learning in Graphical Models*. MIT Press, 1999. Edited Volume.
 [10] A. Kraskov, H. Stögbauer, and P. Grassberger. Estimating mutual information. *Phys. Rev. E*, 69(6):066138, Jun 2004.
 [11] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Stat.*, 22:79–86, 1951.
 [12] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, October 2002.
 [13] A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 Suppl 1, 2006.
 [14] P. Mendes, W. Sha, and K. Ye. Artificial gene networks for objective comparison of analysis algorithms. *Bioinformatics*, 19:122–129, 2003.
 [15] I. Nemenman, F. Shafee, and W. Bialek. Entropy and inference, revisited. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *NIPS*, Cambridge, MA, 2002. MIT Press.
 [16] F. Pérez-Cruz. Estimation of information theoretic measures for continuous random variables. In *NIPS*, 2008. To appear.
 [17] N. Slonim. *The Information Bottleneck: Theory and Applications*. PhD thesis, The Hebrew University, 2002.
 [18] M. Studený and J. Vejnarová. The multiinformation function as a tool for measuring stochastic dependence. In *Learning in Graphical Models*, pages 261–297, Norwell, MA, USA, 1998. Kluwer Academic Publishers.
 [19] G. Wahba. *Spline Models for Observational Data*, volume 59. Society for Industrial and Applied Mathematics (SIAM), 1990.
 [20] E. Xing, R. Sharan, and M. I. Jordan. Bayesian haplotype inference via the Dirichlet process. In *ICML*, New York, NY, USA, 2004. ACM.
 [21] B. Yackley, E. Corona, and T. Lane. Function approximation using a metagraph kernel. In *NIPS*, 2008. To appear.