Towards an Integrated Protein-protein Interaction Map

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science

by

Ariel Jaimovich

Supervised by

Prof. Hanah Margalit and Prof. Nir Friedman

December 2004

The School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel

Abstract

As protein-protein interaction plays a major role in most cellular processes, identifying the full repertoire of interacting protein pairs in the cell is of great importance. This challenge has been addressed both experimentally and computationally. Large-scale experimental studies provide data sets of interacting proteins. These data sets, while only partial and noisy, allow us to characterize properties of interacting proteins and lead to the development of predictive algorithms. Most algorithms, however, ignore possible dependencies between the various interacting pairs, and predict them independently of one another. In this study, we present a computational approach that overcomes this drawback by predicting all interactions simultaneously. For this, we build an integrated probabilistic model that involves all attributes we wish to predict, as well as the observations we have on these attributes. We use the language of relational Markov random fields to build such an integrated probabilistic model, and to learn its properties efficiently. We then use this to predict all interactions simultaneously. We show that by modeling dependencies between interactions, we achieve a more accurate description of the protein interaction network, and gain new insights into the properties of the interacting pairs.

Acknowledgements

I am grateful to my advisors, Hanah Margalit and Nir Friedman, for coming up with the idea behind this work, and for guiding me patiently through the long way it took to implement it. During the last two years I have learned a lot from them, not only about my specific research areas, but also about how to ask the right questions, and how to find the tools to answer them. I am especially thankful to Gal Elidan, who was in many ways a third advisor to this work, and who devoted a great deal of effort and time into helping me. I am very fortunate to be part of Hanah's lab in Hadassah, and Nir's lab in Givaat Ram. Both labs are a constant source of original and productive scientific discussions, as well as of other conversations about more important things in life. I thank all members of both labs, Iftach Nachman, Yoseph Barash, Matan Ninio, Noa Shefi, Ilan Wapinski and Omri Peleg from Nir's lab, and Einat Sprinzak, Yael Altuvia, Esti Yeger-Lotem, Gila Lithwick, Samuel Sattath, Ruti Hirshberg, Arnon Klein, Galit Lipsitz, Neta Ben-Porat and Naama elephant from Hanah's lab, who made my last two years interesting and fun. Especially I want to thank Tommy Kaplan who is also part of both labs, and helped me a great deal in many fruitful discussions. Last but not least, I thank Inbal, for being beside me all this time, and supporting me when times were rough.

Contents

Chapter 1

Introduction

1.1 The World of Proteins

All living organisms share the basic cellular mechanism. Although there is a huge diversity between the simplest uni-cellular creature and the most complicated mammal, they are all consisted of living cells. Amazingly, all those cells, whether from a bacterium or a human being, share the same basic building blocks. These building blocks, which control and determine the behavior of each cell, are proteins.

The proteins are encoded in DNA and generated by a well known mechanism that is often referred to as the central dogma of biology and is depicted in Figure 1.1. It states that the blueprint of each cell is encoded in its DNA

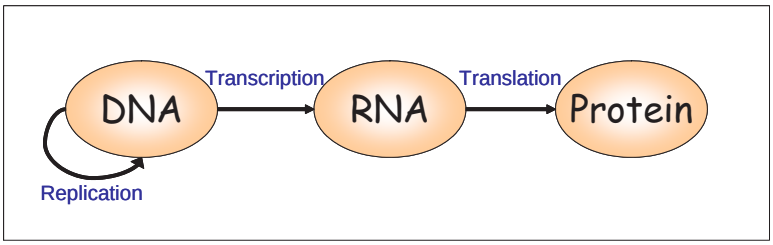


Figure 1.1: The central dogma of biology

sequence. The DNA is replicated (and so the blueprint is passed on to the cell's offsprings), and part of it is transcribed to RNA, from which proteins are translated. After translation of an RNA to a protein there are still many processes the protein has to undergo in order to be functional. First it has to fold into the correct secondary and tertiary structure, then it has to be transported to a specific cellular localization, and often it undergoes specific modifications in order to become active.

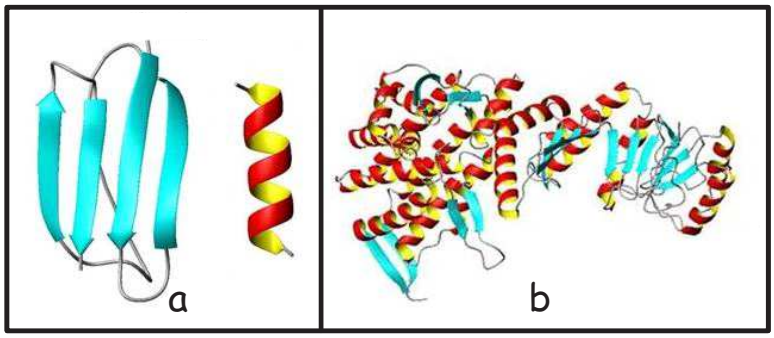


Figure 1.2: The secondary and tertiary structure of proteins

There are many types of proteins in a cell (e.g., 6000 proteins in the budding yeast *Saccharomyces cerevisiae*), and each cell can hold a different number of copies of each protein. The proteins fulfill a wide range of tasks in the cell: they regulate almost all processes in the cell, they act as selective porters on the cell membrane, they accelerate chemical reactions in the cell, and many more.

Conceptually, all proteins can be considered as long strings over an alphabet of twenty letters, each such letter representing an Amino Acid. The length of such a sequence varies from dozens to thousands of amino acids. The specific sequence of amino acids determines the three dimensional structure of the protein. This structure is formed from local secondary structures such as α helices or β sheets (Figure 1.2 a). These local structures join to form the protein configuration in the three dimensional space (termed tertiary structure, see Figure 1.2 b). Folding into a specific structure scheme determines the character and function of the protein, and is vital to its functionality.

The function of proteins in the cell usually involves interactions with other proteins. There are many types of possible interactions. One protein can, for example, transiently interact with another protein in order to modify it. Another common example is a stable interaction between two proteins (or more) that are active only when part of a complex.

1.2 Protein-Protein Interaction Networks

One of the main goals of molecular biology is to reveal the cellular networks underlying the functioning of a living cell. Protein interactions play a central role in these networks. Hence, it is important to know which pairs of proteins interact and characterize each interaction in terms of strength, time

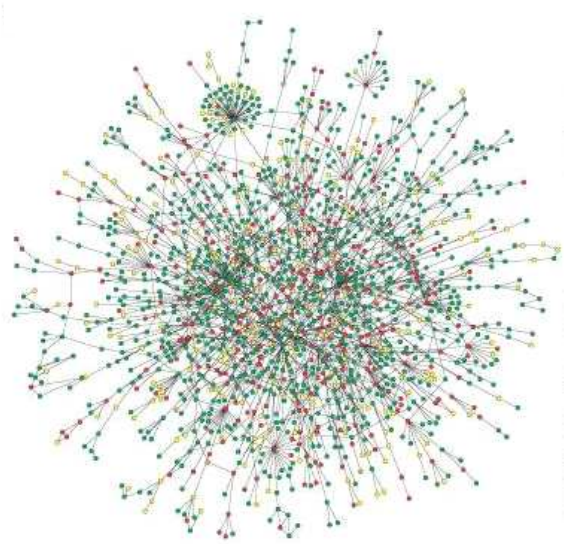


Figure 1.3: Example for an interaction network. Adopted from H Jeong et.al. Nature 411,41 2001

and place. Deciphering this protein interaction network (see for example Figure 1.3) in different conditions and stages is a prerequisite for understanding the complex mechanisms that determine the behavior of the cell.

The challenge of charting protein-protein interactions is complicated by several factors. Foremost is the sheer number of interactions that have to be considered. In the budding yeast, for example, there are approximately 18,000,000 potential interactions between the roughly 6,000 proteins encoded in its genome. Of these, only a relatively small fraction occur in the cell [Sprinzak et al., 2003]. Another complication is due to the large variety of interaction types. These range from stable complexes that are present in most cellular states, to transient interactions that occur only under specific conditions (e.g. phosphorylation in response to an external stimulus).

In the last four years a large effort is invested in developing large-scale screens that will enable us to build such an interaction network. The next sections review some of these methods, as well as describe our own approach for this intriguing problem.

1.3 Methods for Finding Protein-protein Interactions

For the sake of brevity, we divide the methods into categories and give some typical examples for each paradigm. We first roughly divide the methods into three sub-groups : biological methods, computational methods and integrative methods.

1.3.1 Biological Methods

Several biological experimental methods were suggested to approach this problem, we will give here two examples. The first, and maybe the most known one is the *Yeast Two Hybrid* (Y2H) [Uetz et al., 2000, Ito et al., 2001]. In this method the transcription mechanism is used in order to reveal interaction between two proteins. Specifically, one of the proteins is fused to a transcription factor binding domain of a reporter gene, and the other to the transcription factor activation domain. The fusion is done in a way such that if the proteins interact, the interaction will enable the transcription of the reporter gene. This enables testing of many proteins pairs efficiently, namely, for each 'bait' protein, many 'preys' are tested.

Another method, *Tandem Affinity Purification* (TAP) [Rigaut et al., 1999] relies upon recent advances in the technologies that enable detection and identification of small amounts of proteins. Here one protein is fused with a specific tag, and then introduced into a cell, so it can bind to all the proteins it interacts with. Then the cell extract is prepared, and the specific tag is used to capture all protein complexes that include the original fused protein. Finally, all the proteins in the complexes are identified using advanced experimental techniques(e.g., Mass spectrometry). Naturally, this method can only be used to discover interactions within protein complexes or other stable (i.e., not transient) interactions.

1.3.2 Computational Methods

Several computational methods were developed to identify protein-protein interactions. We can divide these into two main approaches. The first approach tries to find functional relations between proteins. It is designed to find protein pairs that are part of the same cellular process but do not necessarily physically interact. The other approach looks only for real physical interactions. There are several examples for the first kind, we will present here two examples.

The first one, *Phylogentic Profile*, was introduced by [Pellegrini et al., 1999]. It suggests that if two proteins are functionally related then the presence of one protein at a certain organism might be useless without the other protein. Hence in most organisms we will see either both proteins or neither one. Formally, we build for each protein i its 'phylogenetic profile' $\vec{v}_i$:

$$v_i[j] = \begin{cases} 1 & \text{if protein } i \text{ appears in organism } j \\ 0 & \text{if protein } i \text{ does not appear in organism } j \end{cases}$$

Now for two proteins k and l we simply have to calculate the correlation between $\vec{v}_k$ and $\vec{v}_l$, and if this correlation exceeds a certain threshold we decide that these proteins are related.

Another strategy for the same 'indirect approach' relies on the assumption that if two proteins interact they should be present at the same times in the cell. [Eisen et al., 1998] suggest to use mRNA level as an evidence for the presence of a protein in a given time in the cell. This means we can try to track down indirect interaction by following the mRNA expression profile of two proteins. Namely, if they tend to co-express at the mRNA level, then they might be functionally related.

The second approach takes a different path. It chooses a set of reliable direct interactions and then tries to capture what differs between interacting pairs and non-interacting pairs. For example, the *Correlated domain signature* method (Sprinzak and Margalit [2001]) looks at the domains that each protein has. These domains are short segments of the protein that often are related to its role or localization. In this method we look for domain pairs that tend to appear in interacting proteins more than expected at random. Specifically, if we look at all pairs of proteins, where protein i has domain A and protein j has domain B and count how many of those pairs interact. These pairs of domain-signatures can be used to support putative interactions between other proteins that contain them.

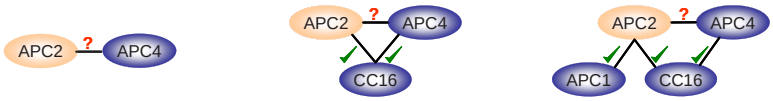
1.3.3 Integrative methods

There are several problems with the methods above. First, most of them are applied only to a limited set of protein pairs and do not cover all possible interactions. Moreover, the overlap between the predictions of all the methods is very small. [Sprinzak et al., 2003] and [von Mering et al., 2002] conducted detailed analyses of the existing methods reliability only to discover that no method alone has reasonable combination of sensitivity and recall (e.g., Y2H has 50% false positives). However, both studies discover that a combination of two (or more) methods can improve significantly the reliability of the identified interactions.

A few studies (e.g., [Jansen et al., 2003, Zhang et al., 2004]) tried to follow this line of thought and combine several attributes into one integrated prediction. These attributes can be either predictions of other computational / experimental methods, or other information sources, such as cellular localization assays. Once you choose a desired set of attributes, it seems intuitively appealing to take a set of reliable interactions, assess the specificity of each method separately, and then combine the prediction of all methods , when each method is weighted according to its reliability. This can be done using machine learning approaches, either in a discriminative way, or using a Bayesian approach. The discriminative methods take the chosen attributes as input and use some kind of classifier (SVM, perceptron ...) to discriminate between interacting and non-interacting pairs (see for example [Bock and Gough, 2003]). In the Bayesian approach we assess the probability of each method prediction given an interaction, and use Bayes rule to calculate the probability of interaction given the predictions of all methods (see for example [Jansen et al., 2003]).

1.4 Simultaneous Prediction of Interactions

These integrated methods examine each pairwise interaction independently, and ignore the dependencies between different interactions. In this study we argue that by examining the protein interaction network as a whole, we can leverage observations on different interactions, as well as from different sources, to get better *joint* predictions of multiple interactions. As a concrete example, consider the budding yeast proteins APC2 and APC4. These proteins were reported as interacting by a large-scale experimental screen [Rigaut et al., 1999]. However, according to Yeast Protein Database (YPD) [Costanzo et al., 2001], APC2 is localized in the cytoplasm while APC4 is localized in the nucleus, i.e., the two proteins are *not* co-localized. Thus, a naive model as in Figure 1.4a, that considers this interaction independently of other interactions, might assign it a low probability. However, we can gain more confidence by looking at related interactions. For example, CC16 is reported to interact with both APC2 and APC4, as illustrated in Figure 1.4b. This observation suggests that the three proteins might form a complex. Moreover, both CC16 and APC1 are nuclear proteins that interact with APC2, Figure 1.4c, implying that APC2 can possibly be nuclear as well, and increasing our belief in the interaction between APC2 and APC4. This example shows two types of inferences. First, certain patterns of interactions (complexes) might be more probable than others. Second, an observation on one interaction can provide information about the attributes of a pro-



(a) (b) (c)

Figure 1.4: Dependencies between interactions can be used to improve predictions. (a) Shows a single interaction of two proteins APC2 and APC4, one of which is localized in the cytoplasm (light orange) and the other in the nucleus (dark blue). The model assigns a low probability to an interaction of proteins that are not co-localized. (b) Introduces a third protein interacting with both proteins of (a). (c) Introduces further evidence that APC2 might also reside in the nucleus, allowing us to hypothesize that this annotation is missing and increase the reliability of the interaction with APC4.

tein (cellular localization in this example), which in turn can influence the likelihood of other interactions.

We present a unified probabilistic model for learning an integrated protein-protein interaction network. We build on the language of relational probabilistic models [Friedman et al., 1999, Taskar et al., 2002] to explicitly define probabilistic dependencies between related protein-protein interactions, protein attributes, and observations regarding these entities. Furthermore, propagation of evidence in our model allows interactions to influence each other in complex ways, facilitating automatic correction of uncertain interacting candidates. Using various proteomic data sources for the yeast *Saccharomyces cerevisiae* we show how our method can build on multiple weak observations to better predict the protein-protein interaction network.

Chapter 2

Probabilistic Graphical Models

We saw in the previous chapter why it is both interesting and hard to investigate protein interactions networks. In this chapter we present the background for the computational tools we are using to explore such networks. We first describe what are probabilistic models, using a small example, and show how they can be mapped into a graphical model. We then show how to generalize this example for any probabilistic model, and introduce some other features we can use to improve our model.

2.1 Probabilistic Models

In this section we describe a small example, formalize it into variables and equations, and finally generalize it. Our model consists of two students (Mark and Jane) who learn together for an exam. Assume we are interested to know how well the students understand what they have learned, and what were their grades in the exam. We formalize this into four discrete random variables :

U_m – Mark's understanding level
U_j – Jane's understanding level
G_m – Mark's exam grade
G_j – Jane's exam grade

Each of these variables can be assigned a few values. We call the values a variable X can be assigned *the domain* of X and denote them by $dom(\mathbf{X})$.

For example, the domains of the random variables of our example are

$$U_i = \begin{cases} \text{HIGH} & \text{If the student i understands the learned material very well} \\ \text{MEDIUM} & \text{If the student i understands it OK} \\ \text{LOW} & \text{If the student i doesn't understand it} \end{cases}$$

$$G_i = \begin{cases} \text{GOOD} & \text{If the student's grade was above 90} \\ \text{AVERAGE} & \text{If the student's grade was between 75 and 90} \\ \text{BAD} & \text{If the student's grade was below 75} \end{cases}$$

In this example we denote by capital letters the random variables (e.g., U_m), and by small letters the values each take (e.g., u_m). For the general case, we will denote by bold capital letters sets of random variables, e.g., $\mathbf{X} = X_1 \dots X_n$ and by $\mathbf{x}$ an assignment for all n variables. Many times instead of $p(\mathbf{X} = \mathbf{x})$ we will use the abbreviation $\mathrm{p}(\mathbf{x})$.

For each assignment of random variables we assign a probability, e.g., we can say that

$$p(U_j = \text{HIGH}, G_j = \text{GOOD}) = 0.5$$

while

$$p(U_j = \text{HIGH}, G_j = \text{BAD}) = 0.1$$

This means that if Jane understands the material well, she is more likely to get a good grade than to get a bad grade. There are three rules a probability function must follow:

1.
$$p \ : \ dom(\mathbf{X}) \ \mapsto \ [0, 1] \tag{2.1}$$

2.
$$\forall \, S \ \subset \ T \quad \sum_{\mathbf{x}_{T \setminus S}} p(\mathbf{x}_T) = p(\mathbf{x}_S) \tag{2.2}$$

3.
$$\forall \, S \quad \sum_{\mathbf{x}_s} p(\mathbf{x}_s) = 1 \tag{2.3}$$

Where S and T are sets of variables, and $\mathbf{x}_S, \mathbf{x}_T$ the assignments for the variables in S and T. We refer the interested reader to [DeGroot, 1989] to learn more about these rules.

Sometimes, the assignments of some variables are correlated with the assignments of other variables. For example, knowing the student's grades clearly tells us something about their level of understanding. We can translate this intuition about correlation into a mathematic formulation that can tell us whether two random variables are correlated:

Definition 2.1.1: Given two random variables X and Y, if $p(x|y) = p(x)$ then we say that X is *independent* of Y. If this equation does not hold we say that they X is *dependent* on Y ∎

These relations are not always direct, for example if Jane and Mark learned together for the exam, then G_j is dependent on G_m. However, assuming Jane and Mark did not cheat in the exam, G_j and G_m do not directly affect each other, and the real connection is between U_j and U_m. We call this property *conditional independence* and its formulation is a natural extension of Definition 2.1.1:

Definition 2.1.2: Given three random variables X,Y and Z, If $p(x|y,z) = p(x|z)$ then we say that X is *conditionally independent* of Y given Z. ∎

In other words, knowledge about Mark's grade, provides us information about the grade Jane got. But, if we would have known exactly how well both students understand the material, Mark's grade provides us with *no* new information about Jane's grade. Hence Mark's grade and Jane's grade are *dependent* on each other if nothing else is known, but they are *conditionally independent* given direct knowledge about their understanding. Given some sets of variables **X**, **Y** and **Z** we denote:

$(\mathbf{X} \perp \mathbf{Y})$	if	**X** and **Y** are independent of each other ($p(x\|y) = p(x)$)
$(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$	if	**X** and **Y** are conditionally independent of each other given **Z** ($p(x\|y,z) = p(x\|y)$)

Now that we have represented our world, we show how can we use this representation in order to ask questions about it . We could be interested to know what is the probability of an assignment for all random variables. This is often referred to as the *likelihood* of the assignment, e.g.,

$$p(G_j = \text{AVERAGE}, U_j = \text{HIGH}, U_m = \text{MEDIUM}, G_m = \text{BAD}) \tag{2.4}$$

Or we could ask about the marginal probability for a set of variables getting a certain assignment, given a (possibly empty) set of assignments for other variables, e.g.,

$$p(G_j = \text{BAD}) \tag{2.5}$$

Sometimes we are interested in the getting the Maximal A-posterior Probability (MAP) of an assignment for a set of variables, given a (possibly empty) set of assignments for other variables.

$$argmax_{u_j} p(U_j = u_j | G_j = \text{MEDIUM}) \tag{2.6}$$

Naively, in order to answer these questions, we can hold a table which contains the probabilities for all possible assignments such as Eq. (2.4)[1]. Now, answering Eq. (2.4) is easy, as we simply need to take the right value from the table. However, calculation of Eq. (2.5) and Eq. (2.6) is a bit more complicated, since we need to sum over all assignments for the other three variables, i.e., :

$$p(G_j = \text{BAD}) = \sum_{u_j} \sum_{u_m} \sum_{g_m} p(G_j = \text{BAD}, U_j = u_j, U_m = u_m, G_m = g_m) \tag{2.7}$$

Answering these questions is called *inference* since we infer from our model what is the probability of a certain state. There are two practical issues we have to address, storage and complexity of computation. If we have four random variables and each one can be assigned three values, we need to store a table of $3^4 = 81$ values. In our world, that is consisted of two students this is feasible, but unfortunately in case of a class with 40 students and 40 grades, we would have 3^{80} possible configurations, and it is not possible to store that amount of entries in a computer. As for the complexity, computing Eq. (2.4) is $O(1)$ but the formula in Eq. (2.7) requires us to sum over all 27 possible configurations of $\{u_j, u_m, g_m\}$, which is again practical for an example of four variables, but infeasible for any real life problem. Graphical Models come to solve both the problem of representation in reasonable space, as well as computing answers to Eq. (2.4) and Eq. (2.5) without having to perform an exponential number of summations. The main idea is to exploit conditional independence properties in order to efficiently represent probabilistic models. When mapping a probabilistic model into a graph, each random variable is represented by a node and the conditional independence is translated into graph structure. Specifically, given three sets of random variables if two sets are independent given the third one then they are mapped into the graph such that the nodes of the third set *separate* the corresponding two sets of nodes. In the next section we formalize this connection between conditional independence and separation in graphs.

2.2 Markov Random Fields

We are using an undirected graphical model called Markov Random Field (MRF)[Hammersley and Clifford, 1971, Pearl, 1988]. In this section we first show how to build an MRF for our small example and then generalize it into

[1]We assume, for now, that we are told what values to fill in this table, but we will later refer to the case where these values are unknown

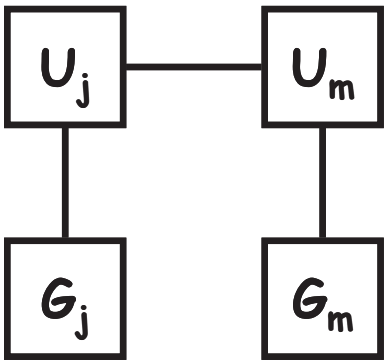


Figure 2.1: The Markov Random Field corresponding to the model in Section 2.1

a formal description of an MRF. But first we have to explain the notion of a factor.

Definition 2.2.1: Let C be a group of random variables, and let $dom(\mathbf{C})$ be their domain, a *factor* ψ with scope C is a mapping from $dom(\mathbf{C})$ to $\Re$ ∎

This notion is very similar to that of a probability function, in the sense that each assignment is mapped to a score correlated with our belief of its likelihood. The difference is that in this case we are not limited into the range $[0, 1]$. In MRFs, factors are assigned to *maximal cliques*.

Definition 2.2.2: Let G be an undirected graph, we define a *clique* in G, as a fully connected group of nodes. A *maximal clique* is a clique which is not a subset of any other clique. ∎

The key in mapping a model into a graph is to build the maximal cliques such that two conditions hold. On one hand we want all the variables that are dependent on each other to appear in the same maximal clique. On the other hand we want variables that are conditionally independent of each other to appear in different cliques. To do so we need to identify the *direct* dependencies within the largest groups of random variables and map them into maximal cliques. In our case the maximal cliques are $\{U_j, U_m\}, \{U_j, G_j\}$ and $\{U_m, G_m\}$. $\{G_j, G_m\}$ is not a clique since $(G_m \perp G_j \mid U_j, U_m)$. Hence for our model we get the graph depicted in Figure 2.1. Notice that in the graph we have built, conditional independence is mapped into structure, i.e., G_j and G_m are separated from each other by U_j and U_m.

In graphical models we often wish to *factorize* $p(g_j, u_j, u_m, g_m)$ into a multiplication over small local factors. e.g.,

$$p(g_j, u_j, u_m, g_m) = \frac{1}{Z} e^{\psi_j(g_j, u_j)} e^{\psi_m(g_m, u_m)} e^{\psi_u(u_j, u_m)} \tag{2.8}$$

Where each ψ is a factor over a maximal clique ($\psi_c(\mathbf{x}_c)$ is the score given to $\mathbf{x}_c$) and Z is a normalization factor, which is termed *Partition Function*. It is used so the probability will comply to the rules in Eq. (2.1), Eq. (2.2) and can be calculated by

$$Z = \sum_{g_j,u_j,u_m,g_m} e^{\psi_j(g_j,u_j)} e^{\psi_m(g_m,u_m)} e^{\psi_u(u_j,u_m)} \tag{2.9}$$

This enables the storage of our parameters in only 3 tables of 4 entries (12 cells instead of 81). We will present in the next sections some efficient algorithms for computing Eq. (2.4) and Eq. (2.5) as well as for computing the partition function.

We are now ready to generalize our description into a formulation of an MRF.

Definition 2.2.3: A Markov Random Field (or Markov Network) over a set of variables $\mathbf{X}$ is defined as a pair $\langle G, \Psi \rangle$ where each variable X_i is mapped into a node in G, and for each maximal clique C in the graph G there is a factor ψ_c such that

$$p(\mathbf{x}) = \frac{1}{Z} \prod_c (e^{\psi_c(\mathbf{x}_c)}) \tag{2.10}$$

∎

Before we finalize our formulation, we need to map the notion of conditional independence into separation in graphs.

Definition 2.2.4: Let G be an undirected graph whose nodes map the random variables $\mathbf{X} = \{X_1 \dots X_n\}$. We say that $\mathbf{X}$ is *locally Markov* with respect to G if

$$\forall\, S \quad p(S|\{\mathbf{X} \setminus S\}) = p(S|\mathcal{N}_S)$$

Where S denotes a group of nodes in G, and $\mathcal{N}_S$ denotes the group of all the neighbors of S in the graph. ∎

This means that each group of variables is conditionally independent of all the other variables in the graph given its neighbors. All that is left now, is to connect this notion of separation with the factorization of the probability function.

Theorem 2.2.5 [Hammersley Clifford]: Let G be a graph over the variables $\mathbf{X} = \{X_1 \dots X_n\}$, if $\forall\, \mathbf{x} \quad p(\mathbf{x}) > 0$ and $\mathbf{X}$ is *locally markov* with respect to G, then $p(\mathbf{x})$ factorizes with respect to G, i.e.,

$$p(\mathbf{x}) = \frac{1}{Z} \prod_j (e^{\psi_c(c)})$$

Remark 2.2.6: Also the reverse direction is correct, if $p(\mathbf{x})$ factorizes with respect to G, then $\mathbf{X}$ is *locally markov* with respect to G. ∎

2.2.1 Template Models

Our aim is to construct a Markov random field over protein-protein interaction networks. This poses two significant problems. First, these models will involve a large number of random variables and potential functions. Robustly estimating a unique parameter for each value in each potential function is impractical. Second, we want the same "rules" to apply to different interaction maps, whether we use all 6,000 proteins of the yeast, or focus on a smaller subset. Clearly, in each case we construct a model over a different set of variables. We address these problems by using *template models*. These models are related to relational probabilistic models [Friedman et al., 1999, Taskar et al., 2002] in that they specify a recipe with which a concrete Markov random field can be constructed for a specific set of proteins and observations. This Markov random field reuses template potentials specified by the model.

As a simple example, we go back to our understanding and grade model. We assume that the correlation between the understanding of the material and the exam grade is the same for all students. As a result we can hold one template potential $\psi_{ug}(u_i, g_i)$ for any student i. Note that this representation solves both problems that we have just introduced. First, the number of parameters in the model does not depend on the number of students (or proteins in our real model). Second, we apply the same "rule" for all the potentials between student and grade. Following the same logic, we create a $\psi_{uu}(u_i, u_j)$ potential for each pair of students that learned together for the exam. Thus, for some k students our induced Markov random field has k potentials between students and grades, all of which replicate the same parameters of the template potential, and in addition a ψ_{uu} potential between each two students that learned together for the exam.

Note that the ψ_{uu} potential needs to be ignorant of the order of its arguments (as we can "present" each pair of students in any order). Thus, the actual number of parameters for ψ_{uu} is six – three parameters when the understanding of the two student is the same (High,Medium or Low),and another three for each possible combination (u_i = High and u_j = Low etc...)

In summary, a template model consists of several template potentials and rules that specify how to replicate each of these template potentials when inducing a specific Markov random field over a specific set of entities (students in our example and proteins in our real model).

2.2.2 Directed potentials

The models we discussed so far make use of undirected dependencies between variables. In many cases, however, a clear directional cause and effect relationship is known. For example, it is natural to view the student's grade (G_m) as a noisy sensor of his understanding (U_m). In this case, we can use a *conditional distribution* $P(G_m = g_m \mid U_m = u_m)$ that captures the probability of the observation (in our case the grade) given the underlying state of the system (in our case the understanding). Conditional probabilities have several benefits. First, due to local normalization constraints, the number of free parameters of a conditional distribution is smaller (six instead of nine in this example). Second, since $\sum_{g_j} (p(G_j = g_j \mid U_j = u_j)) = 1$, such potentials do not contribute to the global partition function. Probabilistic graphical models that combine directed and undirected interactions are called *Chain Graphs* [Buntine, 1995]. In this study we examine a simplified version of Chain Graphs where a dependent variable associated with a conditional distribution (i.e., G_j) is not involved with other potentials or conditional distributions. Denoting by $\mathcal{X}$ the set of all variables that participate in undirected potentials and by $\mathcal{Y}$ the set of all variables that only interact with other variables via conditional probabilities, we can rewrite Eq. (2.10) as

$$P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) = p(\mathbf{X} = \mathbf{x}) p(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) = \left[\frac{1}{Z_x} \prod_c e^{\psi_c(\mathbf{x}_c)} \right] \left[\prod_j p(y_j | \mathbf{u_j}) \right] \tag{2.11}$$

where $\mathbf{u_j} \subseteq \mathbf{X}$ are the directed *parents* of the variable Y_j in the model. Note that Z_x is defined by sum over all assignments to $\mathcal{X}$ (as in Eq. (2.9)).

Chapter 3

The Protein-Protein Interaction Model

Our goal is to build a unified probabilistic model that can capture the integrative properties of the protein-protein interaction network. We want our model to be able to capture the type of reasoning that appears in the example of Figure 1.4. Thus, we aim for simultaneous prediction of all interactions. In addition, the model should take into account different attributes of the protein such as its cellular localization. Furthermore, we would like to explicitly account for measurement noise of different assays. To this end we build a probabilistic model that encompasses the relevant phenomena. This model is composed of classes of random variables and template potentials that represent interactions among these variables (Figure 3.1). We now describe the model in a piecewise fashion.

3.1 Building the Model Skeleton

Given a set of proteins $\mathcal{P} = \{p_1, \ldots, p_k\}$, an interaction network is a set of edges among these proteins. We can describe such a network by a collection of indicator random variables. The first class of random variables we introduce, is the one that describes the "true" interaction network. In this class we have binary random variables that represent whether such an interaction exists . We use the notation $I(p_i, p_j)$ for the random variable that describes an interaction between p_i and p_j:

$$I(p_i, p_j) = \begin{cases} 1 & \text{If there is an interaction between } p_i \text{ and } p_j \\ 0 & \text{Otherwise} \end{cases}$$

Note that this is a symmetric relation, and so $I(p_i, p_j)$ and $I(p_j, p_i)$ designate the same random variable. The simplest template model over such an inter-

action network has a single univariate potential $\psi_1(I(p_i, p_j))$. The induced Markov random field has a potential for each possible edge. Each one of these potentials is identical to the template potential, so that *a priori* all interactions are equally likely.

3.2 Adding Information from Large Scale Assays

Usually, we expect these interaction random variables to be hidden from us. We learn about interactions through experimental assays or computational predictions. For each type of assay a, we have a random variable $IA_a(p_i, p_j)$ that denotes the result of the assay for the interaction between p_i and p_j:

$$IA_a(p_i, p_j) = \begin{cases} 1 & \text{If the interaction between } p_i \text{ and } p_j \text{ was detected by the assay } a \\ 0 & \text{If the interaction between } p_i \text{ and } p_j \text{ was not detected by the assay } a \end{cases}$$

Note that some assays are not symmetric. For example, in the yeast two hybrid assay there is a difference between a bait and a prey. In such a situation $IA_a(p_i, p_j)$ and $IA_a(p_j, p_i)$ are distinct random variables that describe the outcome of two different experiments. These variables are observed (with either positive or negative results) for these cases where the assay has been carried out.

We relate the observed interaction assays with the "true" interactions using a directed conditional distribution $p(IA_a(p_i, p_j) \mid I(p_i, p_j))$ (Figure 3.1(b)). This conditional distribution views the assay as a noisy sensor. Thus, we explicitly model experiment noise and allow the measurement to stochastically differ from the ground truth. It is important to note that $IA_a(p_i, p_j) = 0$ means that the interaction between p_i and p_j was tested in the assay a but not found. In case that this interaction was not tested in the assay a we can ignore the corresponding random variable. This is a direct consequence of the directionality of the potential between $I(p_i, p_j)$ and $IA_a(p_i, p_j)$, since in the case that $IA_a(p_i, p_j)$ is not observed, it has no effect on the rest of the model (i.e., since it is not observed, summing it out results in the identity factor). This is a very important consequence, since many interaction assays are performed only for a small set of proteins, and we can integrate their information without adding too many random variables to our model.

For each type of assay we have a different conditional probability that reflects the particular noise characteristics of that assay. In addition, the basic

model contains a univariate template potential $\psi_1(I(p_i, p_j))$ that is applied to each interaction variable. This potential captures the prior probability of interaction (before we make any additional observations). If we restrict our model to only these components, then note that the model consists of potentials over $I(p_i, p_j)$ and its related assays. Thus, the model specifies a collection of independent *naive Bayes* models for each interaction. In this model we predict interactions based solely on the results of different assays of each interaction, as in Jansen et al. [2003].

3.3 Enriching The Model

We can enrich the model in two different ways. The first is by explicitly describing dependencies between related interactions by potentials over sets of interactions, and the second is by adding information about the properties of each protein.

3.3.1 Adding Dependencies between Interactions

We want to capture local dependencies between different interactions. The *triad model* [Frank and Strauss, 1986] has template potentials over triplets of edges that share common proteins: $\psi_3(I(p_i, p_j), I(p_i, p_k), I(p_j, p_k))$. This potential can capture properties such as preferences for (or against) adjacent edges, as well as transitive closure of adjacent edges. Given $\mathcal{P}$, the induced Markov random field has $\binom{|\mathcal{P}|}{3}$ potentials, all of which replicate the same parameters of the template potential. Note that this requires the potential to be ignorant of the order of its arguments (as we can "present" each triplet of edges in any order). Thus, the actual number of parameters for ψ_3 is four – one when all three edges are present, another one for the case when two are present, and so on. Thus, we allow the evidence about some interactions to propagate and influence our belief on related interactions. Once we introduce such potentials, the interaction variables are not independent of one another, and we can consider simultaneous prediction of all interactions together.

3.3.2 Adding Protein Attributes

A different way to extend the model is by introducing protein attributes that influence the interactions. It is fairly clear that the structure of the interaction network reflects various aspects of the structure, localization, and function of specific proteins. Thus, the likelihood of different patterns of

interactions can depend on these attributes. Here we consider cellular localization as an example of an attribute that may have impact on the interactions. The intuition is clear: if two proteins interact, then we expect them to be co-localized. Since each protein can be present in multiple locations, we model cellular localization by several indicator variables, $L_l(p)$:

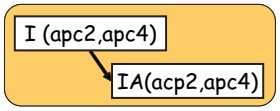
$$L_l(p_1) = \begin{cases} 1 & \text{If } p_1 \text{ is present in cellular localization } l \text{ in some context} \\ 0 & \text{Otherwise} \end{cases}$$

As with interaction variables, localization variables represent the underlying truth and can not be directly observed. Instead, we have access to partial noisy observations of localization through experimental assays (e.g.,Ghaemmaghami et al. [2003]). The outcome of these assays is denoted by localization assay random variables $LA_l(p_i)$, which are observed:

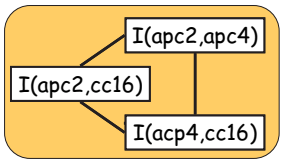
$$LA_l(p_i) = \begin{cases} 1 & \text{If } p_i \text{ was found in cellular localization } l \text{ in some context} \\ 0 & \text{If } p_i \text{ was not found in cellular localization } l \text{ in any context} \end{cases}$$

Again, we distinguish between the case where p_i was searched but not found (in this case $LA_l(p_i) = 0$) and the case where p_i was not looked after (in this case we simply ignore this random variable in our model). We relate each localization assay variable to its corresponding ground truth variable using a conditional probability (Figure 3.1(d)). The parameters of this conditional probability depend on the type of assay and the specific cellular localization. For example, some localizations, such as "bud", are harder to detect as they represent a transient part of the cell cycle. On the other hand other localizations, such as "cytoplasm", are easier to detect since they are present in all stages of the cell's life and many proteins are permanently present in them.

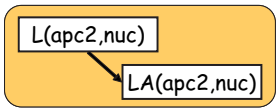
In addition, we also relate the localization variables for a pair of proteins with the corresponding interaction variable between them by introducing for each localization l a potential $\psi_l(L_l(p_i), L_l(p_j), I(p_i, p_j))$ (Figure 3.1(e)). This potential encodes preferences for interactions between co-localized proteins. It also captures the increase (or decrease) in probability of interactions due to their co-localization. As with the template over potentials between interactions, we enforce this potential to be symmetric in the role of p_i and p_j. As discussed above, including such potentials in the model allows for indirect dependencies between interactions. That is, evidence for interaction between p_i and p_j can change the beliefs about the localization of p_i, and consequently change the belief in the existence of the interaction between p_i and another protein p_k.



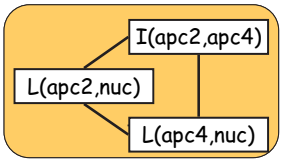
(a) Interaction assay



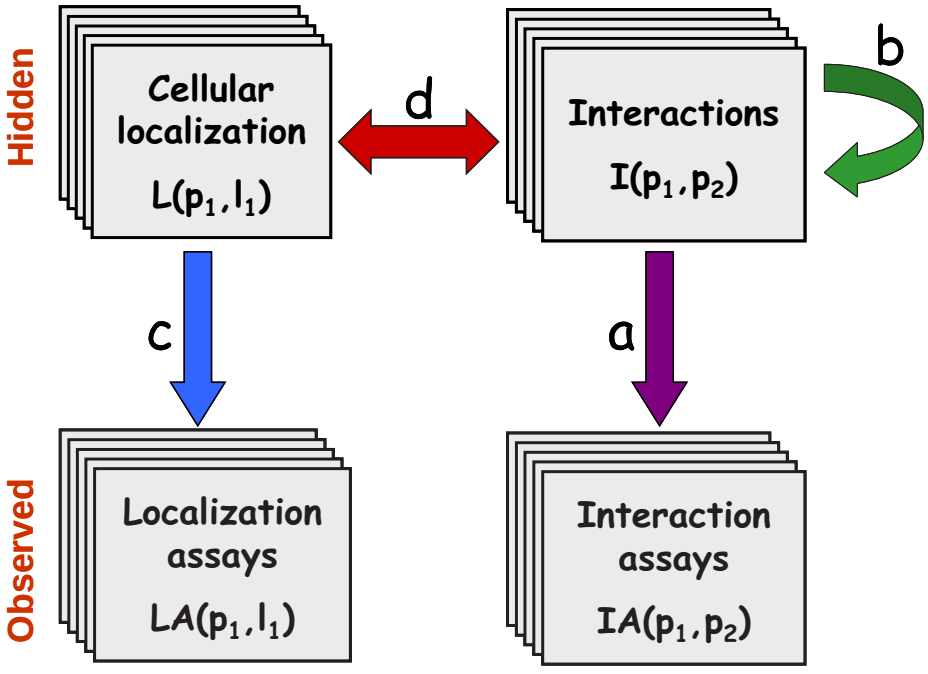
(b) Dependencies among interactions



(c) Localization assay



(d) Localization and interactions



(e) Global model structure

Figure 3.1: The model consisting of four classes of variables. The relations between them are divided to four types of potentials: (a) a potential between an interaction and the corresponding interactions assay; (b) a potential between three related interacting pairs; (c) a potential between a localization and the corresponding localization assay; (d) a potential between an interaction and the localization of the two proteins. The integration of four classes and the global relations between them can be seen in (e).

Chapter 4

Inference in Markov Random Fields

We have already shown how to map probabilistic models onto graphs, and how this provides an efficient representation of the distribution. We have also presented in Section 2.2 a few probabilistic queries in such probabilistic models, and then demonstrated that answering them naively is computationally intractable in real life problems. In this section we show how we can use probabilistic models in order to efficiently compute answers to these queries.

4.1 Exact Inference in Markov Random Field

First, let us return to the example presented in Section 2.1 and the corresponding MRF we have built (Figure 2.1). Assume, for example, we are interested in calculating $p(G_j = \text{BAD})$. As we previously showed, this can be done by Eq. (2.7) but may require an infeasible number of computations. We can use the factorization of the probability function to try and reduce the number of summations. By plugging in Eq. (2.8) we get:

$$p(G_j = \text{BAD}) \quad = \quad \sum_{u_j} \sum_{u_m} \sum_{g_m} \frac{1}{Z} e^{\psi_j(G_j=\text{BAD},u_j)} e^{\psi_m(g_m,u_m)} e^{\psi_u(u_j,u_m)}$$

Since in this equation not all factors depend on all variables, we can change the order of summation:

$$p(G_j = \text{BAD}) = \frac{1}{Z} \sum_{u_j} e^{\psi_j(G_j=\text{BAD},u_j)} \sum_{u_m} e^{\psi_u(u_j,u_m)} \sum_{g_m} e^{\psi_m(g_m,u_m)} \tag{4.1}$$

To sum one variable out we first create an *intermediate factor* that includes all the functions in which the eliminated variable takes part. Then we sum

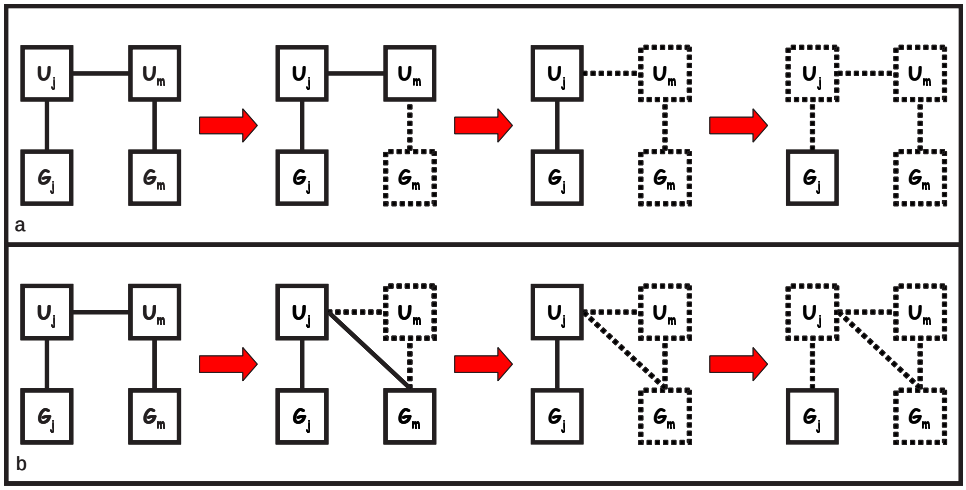


Figure 4.1: A graphic illustration for elimination process (the dashed line represent the parts that are already summed out). (a) shows original elimination order shown in Eq. (4.1) and (b) shows the elimination order in Eq. (4.3) Notice that when summing out a variable whose neighbors are not directly connected, an edge is added since they are now part of the same factor (see for example Eq. (4.4))

out this variable, thus eliminating it from the equation[1]. We denote the resulting marginalized factor with a new notation, for example, if we sum out the variable $\mathbf{x}$ and after marginalization we remain with a function over $\mathbf{y}$ we denote the resulting factor by $f_{\mathbf{x}}(\mathbf{y})$. Thus, we can algebraically describe the process of elimination (it is also illustrated graphically in Figure 4.1):

$$\begin{aligned} p(G_j = \text{BAD}) &= \frac{1}{Z} \sum_{u_j} e^{\psi_j(G_j=\text{BAD},u_j)} \sum_{u_m} e^{\psi_u(u_j,u_m)} f_{g_m}(u_m) \quad (4.2) \\ &= \frac{1}{Z} \sum_{u_j} e^{\psi_j(G_j=\text{BAD},u_j)} f_{g_m,u_m}(u_j) \\ &= \frac{1}{Z} f_{g_m,u_m,u_j}(G_j = \text{BAD}) \end{aligned}$$

The main idea behind these steps is to take advantage of the factorization of the probability function in order to sum one variable at a time. Notice that now, instead of summing over 27 possible configurations for $\{u_j, u_m, g_m\}$ we perform only 9 summations that are done on three iterations. This process is

[1]This process is referred to as *marginalization*, for example in our case we take a table of nine entries and sum each column into one value to end up with a table of three entries, each holding a *marginal* probability.

called *elimination* since at each time we sum out one variable (i.e., eliminate it). The computational complexity of the elimination grows exponentially with the size of the largest *intermediate factor* that is created during the elimination. Note that the order in which we eliminate the variables is very important, if we change the order of elimination in Eq. (4.1) we will get different performance. We will demonstrate this for our example (see also Figure 4.1 b):

$$
\begin{aligned}
p(G_j = \text{BAD}) &= \frac{1}{Z} \sum_{u_j} e^{\psi_j(G_j=\text{BAD},u_j)} \sum_{g_m} \sum_{u_m} e^{\psi_u(u_j,u_m)} e^{\psi_m(g_m,u_m)} && (4.3)\\
&= \frac{1}{Z} \sum_{u_j} e^{\psi_j(G_j=\text{BAD},u_j)} \sum_{g_m} f_{u_m}(g_m, u_j) && (4.4)\\
&= \frac{1}{Z} \sum_{u_j} e^{\psi_j(G_j=\text{BAD},u_j)} f_{u_m,g_m}(u_j)\\
&= \frac{1}{Z} f_{u_m,g_m,u_j}(G_j = \text{BAD})
\end{aligned}
$$

For this elimination order the size of the largest clique is bigger than the one in the original order we presented. Specifically, while the complexity of the original elimination order was $O(3^2)$, the complexity of Eq. (4.3) is $O(3^3)$ since the size of the largest intermediate factor grew from 2 to 3. Unfortunately, choosing an optimal elimination order for a general graph is a hard problem, and although there are some intuitions (e.g., start from the leaves), there is no straightforward solution. Furthermore, for most real-life models, regardless of the elimination order, the size of the largest intermediate factor is exponential with the number of variables in the model.

Though this means that in many cases this exact inference is infeasible, sometimes we might be willing to compromise on an approximate answer. In these cases we resort to approximate inference methods. There are several methods of approximate inference in graphs, in this work we will deal with a special subgroup of these methods, which use an interesting connection between the inference computation in graphical models and free energy minimization in theoretical physics.

4.2 Free Energy Minimization and Inference Approximations

In these two subsections we explain about the surprising connection between free energy minimization and inference. First we show an example for a model where this connection can be intuitively shown and then we give some theoretical foundation for this connection.

4.2.1 The Spin Model

We start with building a graphical model for a new problem. Assume we have a grid of nine particles, each of which can spin either up or down. The direction of the spin is determined by the direction of its neighbors in the grid, as well as by an external force. We are interested in knowing in which direction each spin moves. We use what we have learned in the last sections to build an MRF that can help us solve this problem. First let us determine the random variables for this example

$$\begin{aligned} X_i &= \text{The state of the } i\text{'th spin} \\ Y_i &= \text{The external force operated on the } i\text{'th spin} \end{aligned}$$

And the corresponding domains:

$$\begin{aligned} X_i &= \begin{cases} +1 & \text{If the } i\text{'th spin turns up} \\ -1 & \text{If it turns down} \end{cases} \\ Y_i &= \begin{cases} +1 & \text{If the external force on the } i\text{'th spin is up} \\ -1 & \text{If the external force is down} \end{cases} \end{aligned}$$

To map our model into a graph we have to determine the maximal cliques in it. We have two kinds of direct dependencies, between neighboring spins, and between a spin and the external force operated on it. This results in the graph shown in Figure 4.2 and in the following equation:

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \prod_{\langle i,j \rangle} e^{\psi_{i,j}(x_i, x_j)} \prod_i e^{\psi_i(x_i, y_i)} \tag{4.5}$$

Where $\langle i, j \rangle$ denotes the set of all pairs i, j where spins i and j are neighbors. Now, given an assignment for the external force applied on the spins, we want to find the most probable state of each spin. We can use the elimination process we just described to compute the probability for each assignment of $\mathbf{x}$ and find the most probable among all the assignments. Unfortunately,

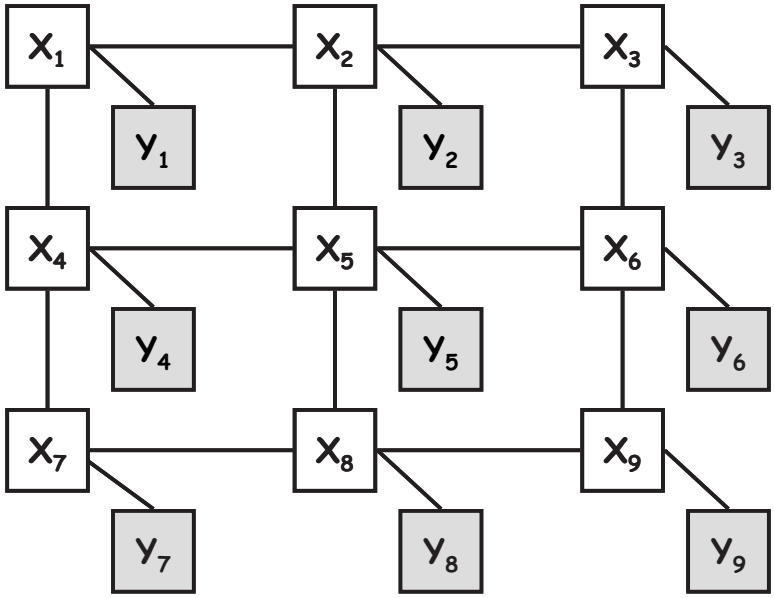


Figure 4.2: The spin model

regardless of the elimination order we choose, it can be shown that the complexity of the elimination order is exponential with the size of the grid.

Another approach to solve this problem comes from a physical perspective, asking what is the state that is energetically preferred, i.e., what is the assignments for $\mathbf{x}$ that bring this system into minimal energy. We show that this two approaches are actually equivalent.

4.2.2 Connecting between Energy and Probability

We now try to reach Eq. (4.5) from another direction. The connection between energy and probability is defined in mechanical statistics by *Boltzman law*:

$$p(\mathbf{x}) = \frac{1}{Z} e^{-\frac{E(\mathbf{x})}{T}} \tag{4.6}$$

The intuition behind it is that the lower the energy of an assignment for $\mathbf{x}$, the higher the probability we have to be in that assignment. This connection depends also on the temperature, since if the temperature is very high, the probability of each state is equal (uniform distribution), and if the temperature is low there will be a low number of preferred states. For example, on our spin model we can define the energy of the spins for some weight function

w as

$$\begin{aligned} E(\mathbf{x},\mathbf{y}) &= \sum_{\langle i,j\rangle} E(x_i,x_j) + \sum_i E(x_i,y_i) \\ &= \sum_{\langle i,j\rangle} (w(x_i,x_j)) + \sum_i (w(x_i,y_i)) \end{aligned}$$

According to the *ising model* w is defined as the constant function -1, which yields:

$$E(\mathbf{x},\mathbf{y}) = -\sum_{\langle i,j\rangle} (x_i x_j) - \sum_i (x_i y_i)$$

Putting this into this Eq. (4.6) and assuming T=1 we get:

$$\begin{aligned} p(\mathbf{x},\mathbf{y}) &= \frac{1}{Z} e^{-\sum_{\langle i,j\rangle}(x_i x_j) - \sum_i (x_i y_i)} \\ p(\mathbf{x},\mathbf{y}) &= \frac{1}{Z} \prod_{\langle i,j\rangle} e^{-x_i x_j} \prod_i e^{-x_i y_i} \end{aligned}$$

We notice now that if we define $\psi_{i,j} = -x_i x_j$ then we obtain Eq. (4.5) from another direction. Hence in this case finding the most probable assignment to $p(\mathbf{x}|\mathbf{y})$ will also minimize $E(\mathbf{x},\mathbf{y})$.

4.2.3 Equivalence between Free Energy Minimization and Inference

Now we are ready to formalize the equivalence between inference and energy minimization for a general model. Assume we have an MRF over a set of random variables $\{X_1 \dots X_n, Y_1 \dots Y_m\}$, where $\mathbf{X}$ are hidden and $\mathbf{Y}$ are observed, and we have some probability function $p(\mathbf{x}|\mathbf{y})$ which we want to compute, where $\mathbf{y}$ are the fixed observed values of $\mathbf{Y}$. As we have seen in section Section 4.1 this calculation might be infeasible. We will start by defining *Gibbs free energy*.

Definition 4.2.1: Let q be a probability function over $\mathbf{x}$, we define the *Gibbs free energy* as:

$$\mathcal{F}(q,\mathbf{x}) = \sum_{\mathbf{x}} (q(\mathbf{x}) E(\mathbf{x},\mathbf{y})) + \sum_{\mathbf{x}} (q(\mathbf{x}) \ln q(\mathbf{x}))$$

Where the first term is usually termed average energy and the second term is simply minus the entropy of q ∎

The heart of the connection between inference in graphical models and minimization of the Gibbs free energy lies in the next proposition.

Proposition 4.2.2:

$$\min_q \left(\mathcal{F}(q, \mathbf{x})\right) = -\ln p(\mathbf{y}) \tag{4.7}$$

$$\text{argmin}_q \left(\mathcal{F}(q, \mathbf{x})\right) = p(\mathbf{x}|\mathbf{y}) \tag{4.8}$$

To prove this proposition we use a well known definition of a distance function between probability functions :

Definition 4.2.3: Given two probability functions p, q over $\mathbf{X}$ we define the *Kulback Leibler distance* between them as:

$$D_{KL}(p\|q) = \sum_{\mathbf{x}} \left(p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x})} \right)$$

∎

We use the two following properties of $D_{KL}(p\|q)$:

$$\forall\, p, q \quad D_{KL}(p\|q) \geq 0 \tag{4.9}$$

$$D_{KL}(p\|q) = 0 \quad \text{iff} \quad \forall\, \mathbf{x} \quad p(\mathbf{x}) = q(\mathbf{x}) \tag{4.10}$$

Now we use these properties to prove Proposition 4.2.2:
Proof: According to Eq. (4.7)

$$\begin{aligned} \mathcal{F}(q, \mathbf{x}) &= \sum_{\mathbf{x}} (q(\mathbf{x}) E(\mathbf{x}, \mathbf{y})) + \sum_{\mathbf{x}} (q(x) \ln q(\mathbf{x})) \\ &= \sum_{\mathbf{x}} (q(\mathbf{x}) E(\mathbf{x}, \mathbf{y})) + \sum_{\mathbf{x}} (q(x) \ln q(\mathbf{x})) + \ln p(\mathbf{y}) - \ln p(\mathbf{y}) \end{aligned}$$

Plugging in Eq. (4.6) and since $\sum_{\mathbf{x}} q(\mathbf{x}) = 1$ we get

$$\begin{aligned} \mathcal{F}(q, \mathbf{x}) &= -\sum_{\mathbf{x}} (q(\mathbf{x}) \ln(p(\mathbf{x}, \mathbf{y})) + \sum_{\mathbf{x}} (q(x) \ln q(\mathbf{x})) + \sum_{\mathbf{x}} q(x) \ln p(\mathbf{y}) - \ln p(\mathbf{y}) \\ &= -\sum_{\mathbf{x}} \left(q(\mathbf{x}) \ln \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \right) + \sum_{\mathbf{x}} (q(x) \ln q(\mathbf{x})) - \ln p(\mathbf{y}) \\ &= \sum_{\mathbf{x}} (q(\mathbf{x}) \ln q(\mathbf{x}) - q(\mathbf{x}) \ln p(\mathbf{x}|\mathbf{y})) - \ln p(\mathbf{y}) \\ &= D_{KL}(q(\mathbf{x}) \| p(\mathbf{x}|\mathbf{y})) - \ln p(\mathbf{y}) \end{aligned}$$

From Eq. (4.10), since D_{KL} is non-negative, we get that the global minimum of $\mathcal{F}(q, \mathbf{x})$ is achieved when the D_{KL} is zero, and this happens iff $p(\mathbf{x}|\mathbf{y}) = q(\mathbf{x})$, proving Proposition 4.2.2. ∎

Several methods use this surprising connection, combined with theoretical approaches from statistical mechanics that compute an approximation for the minimal free energy in order to find approximations to inference problems. We will concentrate on one such approach: Loopy Belief Propagation (LBP).

4.3 Loopy Belief Propagation

In this section we explain how does Loopy Belief Propagation works. We first explain the intuition behind the belief propagation algorithm, and then prove its correctness, using what we have shown in Section 4.2.3.

4.3.1 Belief Propagation

General belief propagation algorithms were proposed by [Pearl, 1988] and have been shown to calculate exact inference in trees. These algorithms use the elimination process we described in Section 4.1 and take advantage of the fact that in many inference questions, there is a repetition of the basic elimination processes. The belief propagation algorithm uses dynamic programming to avoid repeating these calculations. The units of storage are called *messages* and are the result of an elimination process as in Eq. (4.1)

Definition 4.3.1: Belief Propagation Algorithm
We define the algorithm recursively:

$$
\begin{aligned}
b(\mathbf{x}_c) &= \psi_c(\mathbf{x}_c) \prod_{s \in \mathcal{N}_c} m_{s \to c}(\mathbf{x}_c) \\
m_{s \to c}(\mathbf{x}_{s \cap c}) &= \sum_{s \setminus c} \left(\psi_s(\mathbf{x}_s) \prod_{t \in \{\mathcal{N}_s \setminus c\}} m_{t \to s}(\mathbf{x}_s) \right)
\end{aligned}
$$

$b(\mathbf{x}_c)$ is called the *belief* on $\mathbf{x}_c$ and $m_{s \to c}(\mathbf{x}_{s \cap c})$ is called the *message* from s to c, where c and s are cliques and $s \cap c$ denotes the overlapping variables in c and s. ∎

Notice that for a tree this recursion ends in the leaves of the graph, and it can be calculated 'bottom up' in one iteration.

Proposition 4.3.2: *For a tree, given Definition 4.3.1 the following equations hold*

$$\begin{aligned} b(\mathbf{x}_c) &= p(\mathbf{x}_c) = \sum_{x \notin c} p(\mathbf{x}) \\ m_{s \to c}(\mathbf{x}_{s \cap c}) &= f_{s \setminus c}(\mathbf{x}_{c \cap s}) \end{aligned}$$

Where $f_{s \setminus c}(\mathbf{x}_{c \cap s})$ is defined as in Eq. (4.2)

Surprisingly enough, although the original proof holds only for trees, it turns out empirically (e.g., [Murphy et al., 1999]) that the performance of the same algorithm on graphs with loops produces good results. This empirical phenomenon found theoretical basis with works done by [Yedidia et al., 2002] that connects this algorithm with the duality of energy minimization and inference we introduced earlier this section.

4.3.2 Theoretical Support for Correctness of LBP on Graphs with Loops

Let us recall that we defined the Gibbs free energy as

$$\begin{aligned} \mathcal{F}(q, \mathbf{x}) &= \sum_{\mathbf{x}} (q(\mathbf{x}) E(\mathbf{x}, \mathbf{y})) - \sum_{\mathbf{x}} (q(x) \ln q(\mathbf{x})) \\ &= \mathcal{U}(q(\mathbf{x})) - \mathcal{H}(q(\mathbf{x})) \end{aligned}$$

Where $\mathcal{U}(q(\mathbf{x}))$ is the average energy of $q(\mathbf{x})$, and $\mathcal{H}(q(\mathbf{x}))$ is the entropy of $q(\mathbf{x})$. The theoretical support for the correctness of the belief propagation algorithm comes from an approximation for the free energy which was found by [Bethe, 1935] and generalized later by [Kikuchi, 1951]. Namely, if the belief propagation algorithm converges in graphs with loops, then its fixed points correspond to local minima of Kikuchi approximations for the free energy. We first approximate the average energy, using Boltzman law,

$$\begin{aligned} \mathcal{U}(q(\mathbf{x})) &= \sum_{\mathbf{x}} (q(\mathbf{x}) E(\mathbf{x}, \mathbf{y})) \\ &= \sum_{\mathbf{x}} (q(\mathbf{x}) (-\ln p(\mathbf{x}|\mathbf{y}) - \ln Z)) \end{aligned}$$

If $\mathbf{X}$ is *locally markov* with respect to G we can rewrite this as

$$\begin{aligned} \mathcal{U}(q(\mathbf{x})) &= \sum_{\mathbf{x}} \left(q(\mathbf{x}) \left(-\ln \prod_c \psi_c(\mathbf{x}_c) + \ln Z - \ln Z \right) \right) \\ &= -\sum_{\mathbf{x}} \left(q(\mathbf{x}) \left(\ln \prod_c \psi_c(\mathbf{x}_c) \right) \right) \end{aligned}$$

And finally, if we divide the summation over $\mathbf{X}$ to two different sums, one over the variables in c and the other on the rest of the variables, we get:

$$\mathcal{U}(q(\mathbf{x})) \approx -\sum_{c}\sum_{\mathbf{x}_c}(b(\mathbf{x}_c)\ln(\psi_c(\mathbf{x}_c))) \tag{4.11}$$

Where $b(\mathbf{x}_c)$ are our approximations for $q(x)$ marginal probabilities. Note, that if $\mathbf{X}$ is *locally Markov* with respect to G, this is an exact equality and not an approximation.

We now turn to the entropy term. Here we will approximate $q(\mathbf{x})$ by multiplying all factors, and dividing by the messages in order to count each variable once:

$$q(\mathbf{x}) \approx \frac{\prod_c b(\mathbf{x}_c)}{\prod_{c,s} b(\mathbf{x}_{m_{c\to s}})} \tag{4.12}$$

Again we use $b(\mathbf{x}_c)$ to denote our approximation for the marginal probabilities of $q(\mathbf{x})$. Notice that if G is a tree then this is actually not an approximation, but an exact equality. Now we will use Eq. (4.12) to get an approximation of $\mathcal{H}(q(\mathbf{x}))$:

$$\begin{aligned}\mathcal{H}(q(\mathbf{x})) &= \sum_{\mathbf{x}}(q(\mathbf{x})\ln q(\mathbf{x})) \\ &\approx \sum_{\mathbf{x}}\left(q(\mathbf{x})\ln\frac{\prod_c(b(\mathbf{x}_c))}{\prod_{c,s} b(\mathbf{x}_{m_{c\to s}})}\right)\end{aligned}$$

Again, we divide each summation into two sums as in Eq. (4.11):

$$\mathcal{H}(q(\mathbf{x})) \approx \sum_{c}\sum_{\mathbf{x}_c}(b(\mathbf{x}_c)\ln b(\mathbf{x}_c)) - \sum_{c,s}\sum_{\mathbf{x}_{c,s}}(b(\mathbf{x}_{m_{c\to s}})\ln b(\mathbf{x}_{m_{c\to s}})) \tag{4.13}$$

We sum up Eq. (4.11) with Eq. (4.13) to get the Kikuchi approximation for free energy:

$$\begin{aligned}\mathcal{F}_{Kikuchi}(q,\mathbf{x}) = &-\sum_{c}\sum_{\mathbf{x}_c}(b(\mathbf{x}_c)\ln(\psi_c(\mathbf{x}_c)) + \\ &+\sum_{c}\sum_{\mathbf{x}_c}(b(\mathbf{x}_c)\ln b_c(\mathbf{x}_c)) - \sum_{c,s}\sum_{\mathbf{x}_{c,s}}(b(\mathbf{x}_{m_{c\to s}})\ln b(\mathbf{x}_{m_{c\to s}}))\end{aligned}$$

A minimization for $\mathcal{F}_{Kikuchi}(q,\mathbf{x})$ can be found according to KKT conditions, by adding lagrange multipliers that assure we comply to Eq. (2.1) and

Eq. (2.2), deriving the resulting 'Lagrangian' and comparing to zero. The resulting fixed points form exactly the equations in the belief propagation algorithm as in Definition 4.3.1.

This proves that if the loopy belief propagation converges, we are guaranteed that this fixed point is a local minima of the Kikuchi approximation of Gibbs free energy. However, it is not promised that the belief propagation algorithm will indeed converge, and we can not say anything about the quality of the approximation. Notice that for trees, the belief propagation always converges, and the Kikuchi approximation to the gibbs free energy is exact. Hence, this result give another theoretical proof for the correction of Proposition 4.3.2, as well as a theoretical explanation to the surprising performance of the belief propagation algorithm on graphs with cycles.

Chapter 5

Learning the Model Parameters from Data

Until now we assumed the parameters of each local potential are given to us in advance. Obviously, this is not the case in many real-life scenarios (such as in the protein interaction network) where we do not want to rely on any 'expert' assumption as to the amount and character of correlation between random variables. We would like to estimate the parameters of the model from observations using the *maximum likelihood* approach. Given a training set of M samples $(\mathbf{x}[1] \dots \mathbf{x}[M], \mathbf{y}[1] \dots \mathbf{y}[M])$, we would like to find a parametrization of the potentials Ψ, Θ such that the log-likelihood of these samples is maximized.

Given the set of samples, and two indicator functions f_i and f_k[1], we compute the likelihood that these samples were instantiated from these parameters. In order to make the calculations simpler we compute the log likelihood. We start with using Eq. (2.11) to write the average log likelihood of all the instances:

$$\begin{aligned}
\ell(\Psi, \Theta) &= \frac{1}{M} \sum_m \log p(x[m], y[m]) \\
&= \frac{1}{M} \sum_m \log p(x[m]) + \frac{1}{M} \sum_m \log p(y[m]|x[m]) \\
&= \frac{1}{M} \sum_m \left(\sum_i \theta_i f_i(x[m]) - \log Z_x \right) + \frac{1}{M} \sum_m \sum_k (\psi_k f_k(y_j|\mathbf{u_j}))
\end{aligned}$$

[1]Each i and k are indices over the parameter vectors Ψ and Θ such that for each sample they return 1 if this sample satisfies the assignment of this parameter and zero otherwise

Because of the linearity of expectation we can change the order of summation:

$$\ell(\Psi,\Theta) = \sum_i \left(\theta_i \hat{E}[f_i(x)]\right) - \log Z_x + \sum_k \left(\psi_k \hat{E}[f_k(y_j|\mathbf{u_j}]\right) \quad (5.1)$$

where $\hat{E}$ is the empirical expectation ($\hat{E}[f_i(x)] = \frac{\sum_m f_i(x[m])}{M}$ and $\hat{E}[f_k(y_j|\mathbf{u_j}] = \frac{\sum_m f_k(y_j[m]|\mathbf{u_j}[m])}{M}$).

In order to find the most likely set of parameters we need to search the parameter space. To do so efficiently, many methods take a greedy hill climbing strategy. Starting from an initial guess of parameters, these methods iteratively improve the likelihood till convergence. In each iteration the gradient of the current set of parameters is calculated and the next parameters set is achieved by moving the parameters in the direction of the gradient. We now show how we can calculate the gradient of a given set of parameters, thus enabling the use of such hill climbing methods.

We first show how to derive the likelihood according to the undirected parameters, next we show how one can derive according to the directed parameters. We also show how to use these computations in a relational model. Finally, we explain how we deal with the fact that our observations are only partial, and how we learn the parameters of our model.

5.1 Deriving the Undirected Term

Obviously when deriving the likelihood according to the undirected parameters we need to look only at the left term of Eq. (5.1), and if we derive it according to θ_i we get:

$$\begin{aligned}
\frac{\partial \ell(\Theta,\Psi)}{\partial \theta_i} &= \hat{E}[f_i(x)] - \frac{1}{Z_x}\frac{\partial Z_x}{\partial \theta_i} \\
&= \hat{E}[f_i(x)] - \frac{1}{Z_x}\sum_x f_i(x)\exp\left(\sum_j \theta_j f_j(x)\right) \\
&= \hat{E}[f_i(x)] - \sum_x f_i(x)\frac{1}{Z_x}\exp\left(\sum_j (\theta_j f_j(x))\right) \\
&= \hat{E}[f_i(x)] - \sum_x f_i(x) p_\theta(x) \\
&= \hat{E}[f_i(x)] - E[f_i(x)|\vec{\theta}]
\end{aligned}$$

We can see that the result of the derivation is very intuitive, i.e., it is the difference between the empirical counts of each parameter and the a-priori

estimated counts according to the model. If the a-priori estimation is lower than the empirical count, we have to increase this parameter and vice-versa. [Della Pietra et al., 1997].

5.2 Deriving the Directed Term

For the conditional part we would expect, following the same reasoning as in the previous section, that the next equation will hold :

$$\frac{\partial \ell}{\partial \psi_k} = \hat{E}[f_k(y_j|\mathbf{u_j})]$$

Unfortunately, the derivation of the directed parameters is a little more complicated, since we have to make sure that at each stage and for each j we have a legal CPD, i.e

$$\sum_{k:\psi_k \in p(y_j|u_j)} \psi_k = 1$$

To do so, we can assume that our first parameter set is a legal CPD, and just make sure that for each j :

$$\sum_{k:\psi_k \in p(y_j|\mathbf{u_j})} \frac{\partial \ell}{\partial \psi_k} = 0$$

This means we have a constrained optimization problem. We handle this problem by projecting each derivative vector to the 'legal' hyperplane.

First, let us assume for simplicity that we only require $\sum_k \frac{\partial \ell}{\partial \psi_k} = 0$. Then we just need to project each vector of parameters to the 'legal' hyperplane, or in other words, we are given a point $\mathbf{x}^0$ and a hyperplane $\sum_i x_i = 0$ and we are looking for the closest point to $\mathbf{x}^0$ inside the hyperplane, or formally :

$$\begin{array}{c} \min \left(\frac{1}{2}\|\mathbf{x} - \mathbf{x}^0\|^2\right) \\ s.t \ \sum_i x_i = 0 \end{array}$$

To find such a point we use the KKT conditions and write the Lagrangian :

$$\mathcal{L} = \frac{1}{2}\left(\sum_i \left(x_i - x_i^0\right)^2\right) + \lambda\left(\sum_i x_i\right)$$

Now we require for each i that $\frac{\partial \mathcal{L}}{\partial x_i} = 0$ and we get:

$$\begin{aligned} x_i - x_i^0 + \lambda &= 0 \\ x_i &= x_i^0 - \lambda \end{aligned}$$

We assign this inside our constraint and get:

$$\begin{aligned} \sum_i (x_i^0) - \lambda n &= 0 \\ \lambda &= \frac{\sum_i x_i^0}{n} \end{aligned}$$

So we the result is:

$$x_i = x_i^0 - \frac{\sum_i x_i^0}{n}$$

Now, all we need to do in order to apply this to our case, is to change the restriction for each CPD instead of for all parameters together. Then the corrected term will be

$$\left(\frac{\partial \ell}{\partial \psi_k}\right)^* = \hat{E}[f_k(y_j|\mathbf{u_j}] - \frac{\sum_{k:\psi_k \in p(y_j|\mathbf{u_j})} \hat{E}[f_k(y_j|\mathbf{u_j})]}{\sum_{k:\psi_k \in p(y_j|\mathbf{u_j})} 1}$$

5.3 Dealing with Relational Models

Recall that in our model many potentials and conditional probabilities share the same parameters. Thus, we need to compute gradients with respect to the shared parameters. Using the chain rule of partial derivatives, it is easy to see that if $\psi_c(\mathbf{x}_c) = \theta_k$ for all $c \in \mathcal{C}$, then we have a group of parameters that map to the same parameter.

$$\frac{\partial \ell(\Theta, \Psi)}{\partial \theta_k} = \sum_{\text{i is mapped to k}} \frac{\partial \ell(\Theta, \Psi)}{\partial \theta_i}$$

Thus, the derivatives with respect to the template parameters are aggregates of the derivatives of the corresponding entries in the potentials and conditional probabilities of the model.

5.4 Our Parameter Learning Approach

Note that both sections above require that the instances we have cover all the variables in the model. In cases were the evidence is partial, computing the empirical counts is not a trivial task at all. Our original solution to this problem was to use inference in order to fill the instances. This means, that for each parameter set we want to evaluate (this evaluation can happen many times during each line search, for example) we need to perform

inference in order to complete the counts. Unfortunately, this inference calculation is computationally expensive, and caused the learning procedure to be very slow. In order to reduce the number of invocations of the inference procedure we perform the next EM-like learning algorithm. At the E-step we perform inference with the current parameters to complete the missing evidence. Then at the M-step we estimate the maximum likelihood parameters for the full evidence we obtain in the E-step. Notice that the fact that we have full evidence allows us to learn separately the directed and undirected parameters. To learn the undirected parameters we use a greedy hill climbing approach with a standard line search maximization (the gradient is computed as described in Section 5.1). Having complete instances also makes it easier to learn the directed parameters. That is, when given full instances, the directed parameters are independent of each other, and we can optimize the likelihood of each of them separately in closed form, without resorting to hill climbing methods. Thus the estimate for each directed parameter ψ_k, is simply the fraction of the count for the specific assignment for y_j out of the total number of times we saw the corresponding assignments for his parents $\mathbf{u_j}$. Now we can describe our learning algorithm:
Repeat the following two steps till convergence

1. *Infer (using Loopy Belief Propagation) the marginal probabilities of the random variables that are missing in the evidence. Use these marginal probabilities together with the rest of the evidence to compute the empirical counts for each parameter.*

2. (a) *Estimate the undirected parameters using a line search optimization.*

   (b) *Estimate the directed parameters parameters as:*

$$\psi_k = \frac{\sharp(y_j = y_j^k, \mathbf{u_j})}{\sharp(\mathbf{u_j})}$$

   Where we denote by y_j^k the assignment of y_j corresponding to ψ_k.

Since it is guaranteed that in each iteration the likelihood increases ([Neal and Hinton, 1998]), we repeat these steps till convergence. It is important to note that as in every EM-like algorithm, our learning procedure is sensitive to the initial parameter set, and often converges to local minima.

Chapter 6

Application of the Integrated Model to Experimental Data of Protein-protein Interactions

In Chapter 3 we schematically described how to construct a protein-protein interaction network model. In this chapter, we evaluate the utility of the integrative model by concretely instantiating such a model for proteins of the budding yeast *S. cerevisiae*, and evaluating its performance. For this purpose we choose to use four data sources, each with different characteristics. The first two are experimental large-scale assays for identifying interacting proteins: data on protein complexes from the MIPS database [Mewes et al., 1998], and data on interacting protein pairs identified by the yeast two hybrid method [Uetz et al., 2000, Ito et al., 2001]. The third data type is composed of correlated domain signatures learned previously from experimentally determined interacting pairs [Sprinzak and Margalit, 2001]. The fourth one regards experimental data on protein cellular localization [Ghaemmaghami et al., 2003]. For the latter we regarded four cellular localizations (nucleus, cytoplasm, mitochondria and bud). In our model, as described in Chapter 3, we have a random variable for each possible interaction and a random variable for each assay measuring such interaction. In addition we have a random variable for each of the four possible localizations of each protein, and yet again another variable corresponding to each localization assay. Currently, we cannot cope with a model for all 6000 proteins in the budding yeast as this requires a model with close to $20,000,000$ random variables. Thus, we limit ourselves to a subset of the plausible interactions, retaining both positive and negative examples. We constructed this subset from the study of von Mering et al. [2002] that ranked $80,000$ protein interactions according to their reliability based on multiple sources of evidence (including some

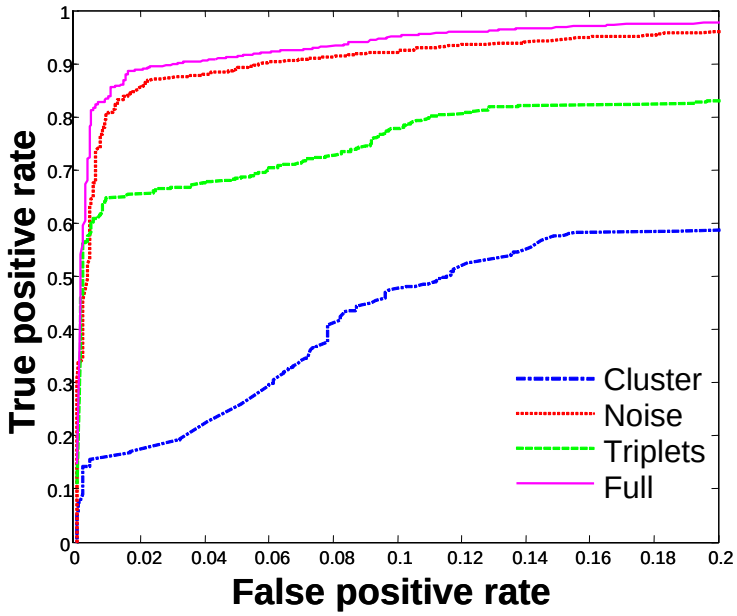


Figure 6.1: Comparison of test performance of a 4-fold cross validation experiment. Shown is the true positive vs. the false positive rates tradeoff for four models: **Cluster** with just interaction, interaction assays, and localization variables; **Noise** that adds the localization assay variables; **Triplets** that adds a potential over three interactions to **Cluster**; **Full** that combines both extensions.

that we do not examine here). From this ranking, we consider the 2000 highest ranked protein pairs as "true" interactions and the last 2000 as "true" non-interacting protein pairs. These 4000 interactions involve 1662 distinct proteins. For these entities we span our full model according to Chapter 3, resulting in approximately 23,000 variables, and 38,000 potentials that share 37 parameters.

The main task is to learn the parameters of the model using the methods described in Chapter 5. To get an unbiased estimate of the quality of the predictions with these parameters, we want to test our predictions on interactions that were not used during the learning. We use a standard 4-fold cross validation technique, where in each iteration we learn the parameters using 1500 positive and 1500 negative interactions, and then test on 500 unseen interactions of each type. Cross validation in the relational setting is more subtle than learning with standard i.i.d. instances. In particular, when testing the predictions on the 1000 unseen interactions, we use not only the

parameters we learned from the other 3000 interactions, but also the observations on them. This simulates a real world scenario when we are given observations on some set of interactions, and are interested in predicting the remaining interactions, for which we have no observation.

To evaluate the performance of the different elements of our model we compare four different models. The baseline **Cluster** model is equivalent to a naive Bayes model similar to the one used by Jansen et al. [2003] and includes the interaction and interaction assays variables, as well as the protein localization variables (where we assume that the localization assay is perfect and take it to be the true localization). The **Noise** model relaxes this latter assumption and distinguishes between the localization variable and the localization assay. In **Triplets** we apply an alternative extension to the **Cluster** model by including potentials over three interactions between three proteins. Finally, the **Full** model combines both extensions.

Figure 6.1 compares the test set performance of these four models. The advantage of using a unified model that allows propagation of influence between varied elements is clear as all three variants improve significantly over the baseline model. We hypothesize that this potential allows for complex propagation of beliefs, beyond the local region of the protein in the graph. When both elements are combined, the model reaches quite impressive results: a 90% true positive rate with just a 5% false positive rate. This is in contrast to the baseline model that achieves less then half of the true positive rate with the same amount of false positives.

Figure 6.2 shows the effect of the training data size on the performance of classification. As shown, even with a relatively small number of interactions, the performance of the model is reasonable, although adding more examples significantly improves the prediction. Moreover we can see that the difference in performance between data sizes of 100 and 1000 is much larger than the difference between 1000 and 3000. This gives us hope, that although we learn the parameters from a relatively small number of interactions, we can use these parameters to predict interactions in much larger models.

Figure 6.3 shows the average log likelihood per sample, as a function of the number of iterations in the EM learning procedure described in Section 5.3. We can see that after less then seven iterations, the algorithm converges. We can also see that although the EM iterations continue, no over-fitting occurres, since the test average log likelihood does not decrease.

Recall that in our model we explicitly account for noise in the localization assay. Thus, it is also insightful to compare the localization predictions made by our model with the annotations of Ghaemmaghami et al. [2003]. For example, out of 1662 proteins in our experiment, 590 proteins are annotated by Ghaemmaghami et al. [2003] as localized in the mitochondria. Our model

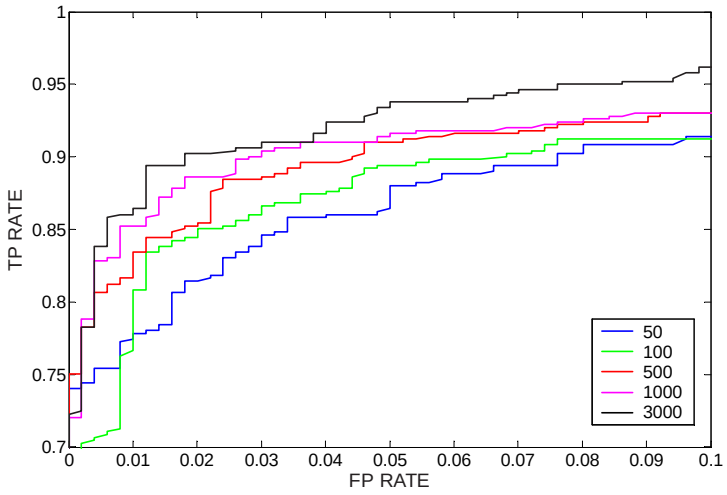


Figure 6.2: Learning Curve: The performance of **Full** on the same test set, using different training data sizes.

predicts that 878 proteins are mitochondrial. These 878 proteins contain the 590 proteins annotated by Ghaemmaghami et al. [2003], 55 additional proteins that are mitochondrial according to YPD, and 59 additional proteins for which there is no known localization. It is reasonable to assume that our current experimental knowledge about the localization of many proteins is still incomplete. Hence, the remaining 174 proteins, which have been annotated by Ghaemmaghami et al. [2003] to different cellular compartments are not necessarily false positive predictions. This example suggests that we are able to correctly predict many localizations and also hypothesize about additional unknown ones.

To get a better sense of the performance of the model, we consider specific examples where the predictions of our model differ from those of the baseline model. We first consider the unobserved interaction between the EBP2 and NUG1 proteins. These proteins are part of a large group of proteins involved in rRNA biogenesis and transport. Localization assays identify NUG1 in the nucleus, but do not report any localization for EBP2. The interaction between these two proteins was not observed in any of the three interaction assays included in our model. Consequently, the baseline model assigns this interaction a very low probability. In contrast, propagation of evidence in the network of the full model effectively integrates information about interactions of both proteins with other rRNA processing proteins. We show a small fragment of this network in Figure 6.4a. In this example, the model is able

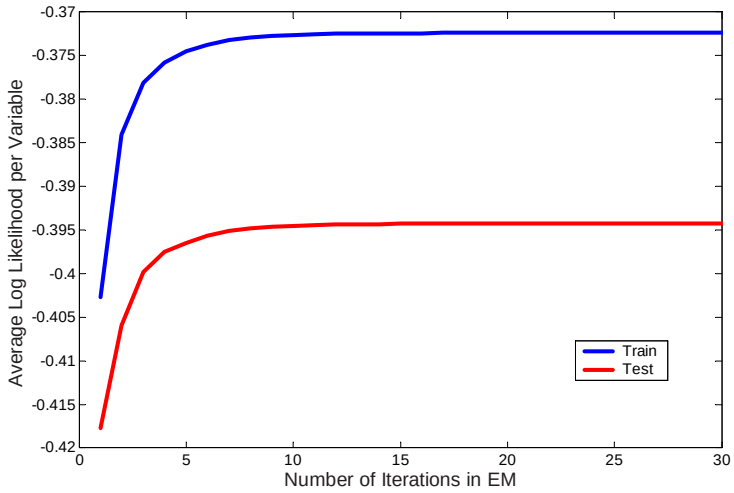
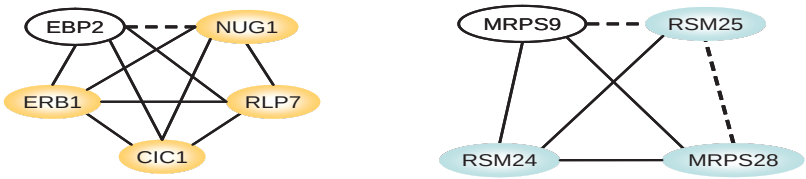


Figure 6.3: The convergence of the EM algorithm: after seven iterations both graphs reach saturation.

to make use of the fact that several nuclear proteins interact with *both* EBP2 and NUG1, and thus predicts that EBP2 is also nuclear, and indeed interacts with NUG1. Importantly, these predictions are consistent with the cellular role of these proteins, and are supported by other experiments reported in the literature [Costanzo et al., 2001, von Mering et al., 2002].

As another, somewhat more complex, example we consider the interactions between RSM25, MRPS9, and MRPS28. While there is no annotation of RSM25's cellular role, the other two proteins are known to be components of the mitochondrial ribosomal complex. Localization assays identify RSM25 and MRPS28 in the mitochondria, but do not report any observations about MRPS9. As in the previous example, neither of these interactions was tested by the assays in our experiment. As expected, the baseline model predicts that both interactions do not occur with high probability. In contrast, by utilizing a fragment of our network shown in Figure 6.4b, our model predicts that MRPS9 is mitochondrial, and that both interactions occur. Importantly, these predictions are verified by the literature [Costanzo et al., 2001, von Mering et al., 2002]. These predictions suggest that RSM25 is related to the ribosomal machinery of the mitochondria. Such an important insight could not be gained without using an integrated model such as the one presented here.



(a) (b)

Figure 6.4: Two examples demonstrating the difference between the predictions by our model and those of the baseline naive clustering model. Solid lines denote observed interactions and a dashed line corresponds to an unknown one. Orange colored nodes represent proteins that are localized in the nucleus and blue colored ones represent proteins that are localized in the mitochondria. Uncolored nodes have no localization evidence. In (a), unlike the naive model, our model correctly predicts that EBP2 is localized in the nucleus and that it interacts with NUG1. Similarly, in (b) we are able to correctly predict that MRPS9 is localized in the mytochondria and it interacts with RSM25, that also interacts with MRPS28.

Chapter 7

Discussion

Predicting the protein-protein interaction network is not a trivial task. The partial and noisy data from interaction assays, the small fraction of true interactions out of the vast amount of potential protein pairs, and the variation between interaction types are the main factors that make this problem so hard. Our framework is different from many existing integrative methods (e.g., [Bock and Gough, 2003]) in the sense that it suggests not only a predictive algorithm, but also provides real insights into the relations between the attributes of the interactions as well as into the relations between these and the attributes of the proteins that take part in the interactions.

We constructed a concrete model that takes into account interactions, interaction assays, localization of proteins in several compartments, and localization assays, as well as the relations between these entities. Our results demonstrate that modeling the dependencies between interactions leads to a significant improvement in predictions. Furthermore, our model accounts for the inherent noise in each of the different assays (both of interaction assays and localization assays), and we show that this noise model allows the evidence to propagate through the model to reach plausible conclusions (see Figure 6.4).

Our main insight is that we should view this problem as a *relational learning problem* where observations about different entities are not independent, and that the real dependency is between these hidden entities (e.g., the interactions and localizations) and not between the observations on them. To capture this, we build on and extend tools from relational probabilistic models to combine multiple types of observations about protein attributes and interactions in a unified model. These models exploit a template level description of the model to induce models for a given set of entities and relations among them [Friedman et al., 1999, Taskar et al., 2002]. In particular, our work is related to applications of these models to *link prediction* [Getoor

et al., 2001, Taskar et al., 2003b]. The main advantage of our approach is that our model learns the parameters from the data while accounting for all evidence, and at the same time taking into account the dependency between hidden entities of our model. By using the relational model, we enable learning of the same parameters for all the copies of a specific template potential. This learning algorithm is different from other works (e.g., [Iossifov et al., 2004]) that use the characteristics of the interaction network in prediction, in the sense that these methods learn the reliability of each interaction assay separately, and then use the structure attributes of the resulting network (e.g., the node degrees) to improve prediction (thus ignoring dependencies between interactions when learning the parameters). The relational models enable incorporation of all the evidence from different data sources when learning the parameters, while taking into account that many potentials in the model are in fact copies of one template (e.g., the relation between a certain interaction assay and the real interaction).

Learning the parameters of a Markov network, even when the training set is composed of complete data, is not an easy task, and a few approaches were suggested to address this issue (e.g., [Taskar et al., 2003b]). However, a major problem that is not addressed in these works, is how to deal with a large number of unobserved random variables (as in our training data). This poses significant challenges for the learning algorithm, since we have to invoke the inference function each time we need to estimate counts (see Section 5.4). Learning the parameters of our model remains the bottleneck of our algorithm, and in our future work, we are looking for better ways to find an optimal set of parameters.

Our probabilistic model over network topology is also related to models devised in the literature of *social networks* (e.g., [Frank and Strauss, 1986]). An important difference from these studies is that we combine these models with potentials that deal with properties of proteins that affect the patterns of interactions of specific proteins. In a recent study, Iossifov et al. [2004] proposed a method to describe properties of an interaction network topology when combining predictions from literature search and yeast two-hybrid data for a data-set of 83 proteins. Their model is similar to our **Triplet** model in that it combines dependencies between interactions and observations about interactions. Their model of dependencies, however, is quite different and deals with the global distribution of node degrees in the network, rather than on local patterns of interactions. Other recent studies employ variants of Markov random fields to analyze protein interaction data. In these studies, however, the authors assumed that the interaction network is given and use it for other tasks, e.g., predicting protein function [Deng et al., 2004] and clustering interacting co-expressed proteins [Segal et al., 2003].

Our emphasis here was on presenting the methodology and evaluating the utility of integrative models. However, it is clear that these models can facilitate incorporation of additional data sources. Our modeling framework allows us to naturally extend the model to other properties of the interactions and the proteins. We intend to add to our model other protein attributes such as cellular processes or expression profiles, as well as different interaction assays. For example, we can look at a set of gene expression experiments under different conditions, and add the correlation of expression profile under each condition as a noisy observation on the interaction between proteins. We can also make our model more accurate by refining our random variables cardinality, e.g., we can look at an interaction not as a binary random variable, and allow it to take more values (not interacting ; transient interaction ; stable interaction)

Another important extension for our model is trying to learn new dependencies. From the computational angle, learning parameters of large undirected networks is a very challenging task, but from the biological angle, many new insights can be gained. For example, the cellular localization of a protein might influence the probability of its interactions being observed by a particular assay (e.g., yeast two-hybrid experiment might be more accurate on nuclear proteins and less accurate on mitochondrial proteins). Such dependencies can be incorporated into our model by adding the suitable template potential. An exciting challenge is to learn which dependencies actually improve predictions. This can be done by methods of *feature induction* [Della Pietra et al., 1997]. Such methods can also allow us to discover high-order dependencies between interactions and protein properties.

It is important to state, that in addition to extending our framework to more elaborate models, we need to build networks that consider a larger number of proteins. This poses several technical challenges. Approximate inference in larger networks is both computationally demanding and less accurate. Generalizations of the basic loopy belief propagation method (e.g., [Yedidia et al., 2002]) and other related alternatives ([Jordan et al., 1998, Wainwright et al., 2002]), may improve both the accuracy and the convergence of the inference algorithm. As we previously mentioned, learning presents additional computational and statistical challenges. In terms of computations, the main bottleneck lies in multiple invocations of the inference procedure. One alternative is to utilize information learned efficiently from few samples to prune the search space when focusing on larger models. That is, we can learn the parameters on a model with a relatively small number of interactions, and use these parameters to predict interactions on a larger model (see Figure 6.2). The statistical challenge when learning is to improve the accuracy of the predictions of the learned model. Recent results suggest that

large margin discriminative training of Markov random fields can lead to a significant boost in prediction accuracy [Taskar et al., 2003a]. These methods, however, apply exclusively to fully observed training data, and we need to find ways to extend them to our partial data scenario.

Our challenge is to find computational solutions to the problems discussed above, in order to enable more complex and interesting models, that will lead to better prediction and new insights. The ultimate goal of this research is to be able to build a model that will capture most important dependencies between the interaction attributes and the protein attributes, while being able to infer the hidden attributes of the model. Such a model, will enable us to provide optimal prediction on many interesting hidden attributes. We hope, that this current work establishes the framework that will enable such developments.

Bibliography

H. A. Bethe. *Proc. Roy. Soc. London*, 150:552, 1935.

J. R. Bock and D. A. Gough. Whole-proteome interaction mining. *Bioinformatics*, 19:125–134, 2003.

W. Buntine. Chain graphs for learning. In *UAI '95*, pages 46–54. 1995.

M.C. Costanzo, M.E. Crawford, J.E. Hirschman, J.E. Kranz, P. Olsen, L.S. Robertson, M.S. Skrzypek, B.R. Braun, K.L. Hopkins, P. Kondu, C. Lengieza, J.E. Lew-Smith, M. Tillberg, and J.I. Garrels. Ypd, pombepd, and wormpd: model organism volumes of the bioknowledge library, an integrated resource for protein information. *Nuc. Acids Res.*, 29:75–9, 2001.

M. H. DeGroot. *Probability and Statistics*. Addison Wesley, Reading, MA, 1989.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19: 380–393, 1997.

M. Deng, T. Chen, and F. Sun. An integrated probabilistic model for functional prediction of proteins. *J Comput Biol*, 11:463–75, 2004.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–8, 1998.

O. Frank and D. Strauss. Markov graphs. *Journal of American Statistics Association*, 81, 1986.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI '99*. 1999.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of relational structure. In *Eighteenth International Conference on Machine Learning (ICML)*. 2001.

S. Ghaemmaghami, WK. Huh, K. Bower, R. W. Howson, A. Belle, N. Dephoure, E. K. O'Shea, and J. S. Weissman. Global analysis of protein expression in yeast. *Nature*, 425:737 – 741, 2003.

J. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.

I. Iossifov, M. Krauthammer, C. Friedman, V. Hatzivassiloglou, J.S. Bader, K.P. White, and A. Rzhetsky. Probabilistic inference of molecular networks from noisy data sources. *Bioinformatics*, 20:1205–13, 2004.

T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc Natl Acad Sci U S A*, 98:4569–4574, 2001.

R. Jansen, H. Yu, D. Greenbaum, Y. Kluger, N. J. Krogan, S. Chung, A. Emili, M. Snyder, J. F. Greenblatt, and M. Gerstein. A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302:449–453, 2003.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational approximations methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

R. Kikuchi. A theory of cooperative phenomena. *Phys. Rev.*, 81:988–1003, 1951.

HW Mewes, J Hani, F Pfeiffer, and D Frishman. MIPS: a database for genomes and protein sequences. *Nucleic Acids Research*, 26:33–37, 1998.

K. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proc. Fifthteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, 1999.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

M. Pellegrini, E. M. Marcotte, and T. O. Yeates. A fast algorithm for genome-wide analysis of proteins with repeated sequences. *Proteins*, 35:440–446, 1999.

G. Rigaut, A. Shevchenko, B. Rutz, M. Wilm, M. Mann, and B. Seraphin. A generic protein purification method for protein complex characterization and proteome exploration. *Nat Biotechnol*, 17:1030–1032, 1999.

E. Segal, H. Wang, and D. Koller. Discovering molecular pathways from protein interaction and gene expression data. In *Proc. Eleventh International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 2003.

E. Sprinzak and H. Margalit. Correlated sequence-signatures as markers of protein-protein interaction. *J Mol Biol*, 311:681–692, 2001.

E. Sprinzak, S. Sattath, and H. Margalit. How reliable are experimental protein-protein interaction data? *J Mol Biol*, 327:919–923, 2003.

B. Taskar, A. Pieter Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI '02)*, pages 485–492, 2002.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 16*, Cambridge, Mass., 2003a. MIT Press.

B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Advances in Neural Information Processing Systems 16*, Cambridge, Mass., 2003b. MIT Press.

P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J. M. Rothberg. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature*, 403:623–627, 2000.

C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.

M. J. Wainwright, T. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. In *P roc. Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI '02)*, 2002.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR-2002-35, Mitsubishi Electric Research Labaratories, 2002.

L. V. Zhang, S. L. Wong, O. D. King, and F. P. Roth. Predicting co-complexed protein pairs using genomic and proteomic data integration. *BMC Bioinformatics*, 5:38, 2004.