35
Integrating Information for Protein Function Prediction

William Stafford Noble and Asa Ben-Hur

1 Introduction

Most of the work on predicting protein function uses a single source of information – the most common being the amino acid sequence of the protein (see Chapter 30). There are, however, a number of sources of data that are predictive of protein function. These include protein–protein interactions (Chapter 31), the genomic context of a gene (Chapter 32), the protein's structure (Chapter 33), information mined from the biological literature (Chapter 34) and data sources indicating coregulation, such as gene expression and transcription factor binding [10]. A classifier that predicts function based upon several sources should provide more accurate predictions than can be achieved using any single source of data. However, the heterogeneous nature of the data sources makes constructing such a unified classifier challenging.

We have divided the various methods for data integration into five categories (Figure 1). First, *vector-space integration* consists of characterizing proteins in various ways by a set of variables, i.e. as vectors. Any standard classification method can then be applied to the resulting vector-space representation. An alternative – *classifier integration* – is to train a classifier on each source of data and then combine the predictions of the various classifiers into a single prediction. *Kernel methods* are a recent advance in the field of machine learning [41]. These methods provide a coherent framework for integrating various sources of data, applicable even when there is no explicit vector-space representation of the data. Several sources of data form a network that is informative of functional relationships. The prime example of such a network is protein–protein interaction data. Proteins that interact often do so because they participate in the same pathway. Therefore, the network of protein–protein interactions in a cell can be informative of protein function (see Chapter 32). The final two approaches model such networks and their relationship to protein function. Graphical models, both directed and nondirected, provide a probabilistic framework for data integration [23]. Modeling is achieved by representing local probabilistic dependencies; the network

Bioinformatics - From Genomes to Therapies Vol. 3. Edited by Thomas Lengauer

ISBN: 978-3-527-31278-8

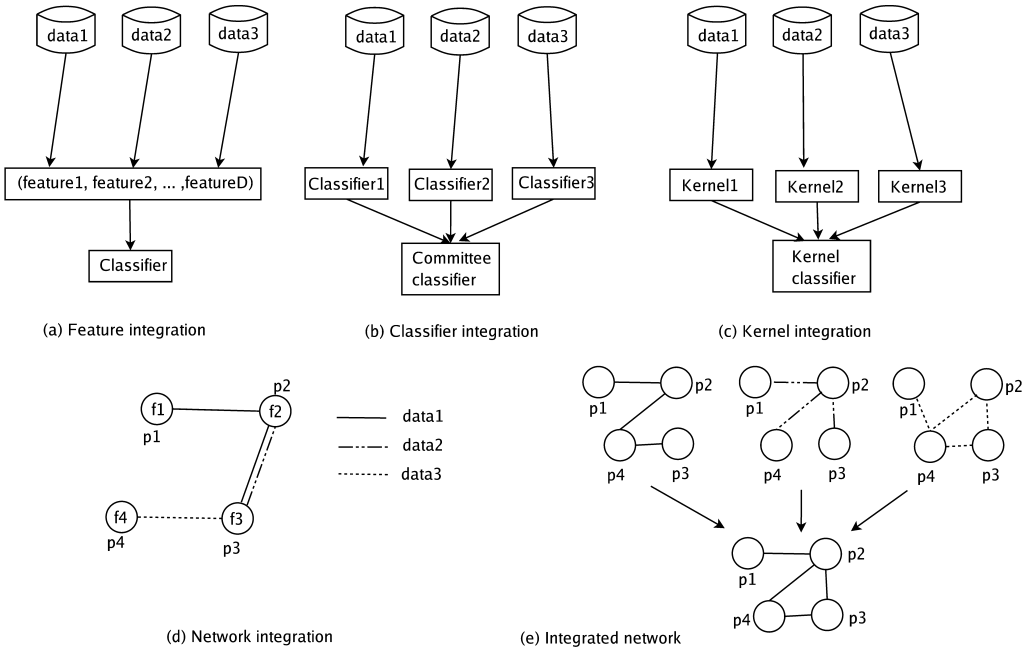

Figure 1 Schematic description of the various methods for integrating genomic information for prediction of protein function. (a) Integration by concatenating data sources into a vector of features. (b) Integrating predictions of several classifiers. (c) Data integration using kernel methods. (d) Integration of network information, typically by Bayesian methods. (e) Integration of several networks of functional relationships into a single network.

structure of these models makes them a natural choice for capturing networks of functional relationships. The last form of integration we discuss does not aim at explicit prediction of protein function, but instead integrates several networks of functional relationships, such as various forms of interaction, coexpression, coregulation, etc., into a single network that unifies all those relationships.

2 Vector-space Integration

Perhaps the simplest form of data integration is to summarize, for each protein, a variety of relevant types of data in a fixed-length vector and feed the resulting collection of vectors into a classification algorithm. This approach has the advantage of simplicity, but treating each type of data identically does not allow us to incorporate much domain knowledge into the design of the

classifier. For example, certain sources of data may benefit from a different measure of similarity than others (see Section 4).

An early example of such an approach is described in Ref. [14]. This work presents a limited form of data integration: many different types of protein features are used, but most of these features are derived from the protein's amino acid sequence. Such features include protein length, molecular weight, charge, amino acid composition (i.e. residue frequencies) and isoelectric point. For a subset of the data for which three-dimensional (3-D) structures are available, the authors also include several features based upon secondary structure features; however, the experiments suggest (somewhat surprisingly) that these features are not very informative. The authors apply three different out-of-the-box machine learning algorithms to their data and compare the resulting performance with that of the BLAST sequence comparison algorithm [1] at predicting whether a sequence is an enzyme as well predicting the first two digits of the protein's Enzyme Commission (EC) number. Among the three machine learning algorithms – the C4.5 decision tree, naive Bayes and *k*-nearest-neighbor algorithms – the best-performing algorithm is *k*-nearest-neighbor, which predicts the class of a query protein by finding the most similar protein in the training set and outputting the corresponding label (see Chapters 24 and 27 for more detailed descriptions of this algorithm). This simple approach works as well as BLAST at discriminating between enzymes and nonenzymes, but less well when the task is more specific. The latter result is not surprising, since many of the enzyme classes are characterized by highly specific sequence features [4].

A closely related set of experiments was described 5 years later in Ref. [24]. Like in the previous work, the authors summarize each protein using a fixed-length vector of features derived from the amino acid sequence. After considering 25 such features, the authors settle on 14, which include straightforward features such as average hydrophobicity and number of negatively charged residues, as well as outputs from nine different previously described prediction methods. These predictions include subcellular location, various types of post-translational modifications, low-complexity regions, transmembrane helices, etc. The resulting 14-element feature vectors are given to a feed-forward neural network, which can subsequently predict EC numbers and "cellular role" Gene Ontology (GO) terms with good accuracy.

A larger version of this type of experiment was described in Ref. [46]. In this work, the authors build classifiers for all EC protein families that include 50 or more members (299 families), as well as 233 Pfam families [45]. Each protein is represented using an extremely rich collection of 453 features. These features include statistics derived from the protein sequence, including amino acid frequencies, predicted secondary structure content, molecular weight, average hydrophobicity, isoelectric point, etc. In addition, the authors

extract information about sub-cellular location, tissue specificity, etc., from the Swiss-Prot database [3] and encode this information in nine of the 453 features. The paper demonstrates the utility of probabilistic decision trees on this task. This algorithm is essentially an improved version of the C4.5 decision tree classifier, and does a better job of handling unbalanced data sets (when negative examples far outnumber positive examples) and missing data, and which exhibits more stable learning behavior.

Protein structure is often conserved when no significant sequence conservation can be detected. Instead of making predictions using direct structure comparison, one can represent structural features of the protein, analogously to the methods presented earlier, that represent features of the protein sequence [15]. This paper characterizes a protein using several structural features: the total surface area attributable to each residue type, the fractal dimension of the protein surface (which quantifies how "crinkly" the surface of the protein is), surface area to volume ratio, secondary structure content, and the presence of cofactors and metals. Then a support vector machine (SVM) classifier is used to predict the the first digit of the EC number of enzymes whose structure is known. The prediction task prepared by Dobson and Doig [15] is a very difficult one: in each of the six enzyme classes no two structures belong to the same SCOP superfamily. Therefore, sequence-based methods will provide very poor results. The authors have not compared their approach to a sequence-based approach that uses the same set of sequences, so it is unclear whether these structure-based features provide added value. In general, the advantage of using sequence-based classifiers is that many more protein sequences are available than protein structures.

One of the challenges in vector-space integration is determining the contribution of each feature to the accuracy of the classifier and finding small subsets of features that maintain or improve classifier accuracy. This task is known as *feature selection* and is an active area of research in machine learning. The interested reader can find a wealth of information about the state-of-the-art of the field in Refs. [18,19]. The simplest approach to feature selection is the so-called *filter* method whereby one computes for each feature a statistic that reflects how predictive the feature is. Statistics that achieve this goal include the area under the receiver operating characteristic (ROC) curve, the Pearson correlation coefficient, the Fisher criterion score, etc. [18,19]. Independently scoring each feature does not take into account the redundancy that often exists in high-dimensional data such as gene expression and also ignores the classifier with which the data will be ultimately classified. These issues are handled by *wrapper* or *embedded* methods (see Refs. [18,19]). Wrapper methods use a classifier to evaluate the merit of subsets of features and, as such, can be combined with any classifier. In embedded methods, on the other hand, the classifier is part of the selection process and uses the properties of the classifier

to select relevant features. An example of a simple embedded method is the recursive feature elimination (RFE) method [20], that for a linear classifier iteratively removes features for which the magnitude of the corresponding component of the classifer's weight vector is the smallest.

The drawback of vector-space integration is modeling all features the same way. One way to address this issue is to train different classifiers for each source of data and then to combine the predictions of the different classifiers. We call this integration method *classifier integration*, which is described in the next section. *Kernel methods*, which are presented in Section 4, train a single classifier, but allow more flexibility in combining data sources than the vector-space integration methods, by allowing the user to define a different similarity measure for each data source and thereby incorporating more domain knowledge into the design of the classifier. Moreover, kernel methods are applicable in modeling data sources such as protein sequences where no obvious vector-space representation is available.

3 Classifier Integration

The second approach to building a unified protein classification algorithm trains several classifiers and then combines their predictions. Gene finding is a well-known bioinformatics problem for which combining the predictions of several classification methods can provide more accurate predictions [40]. Conceptually, there are several classes of methods for combining the output of different classifiers:

(i) Integration of different classification methods, each trained on the same data.

(ii) Integration of the same method trained on different subsets of the data or on different subsets of features. This is an active field of research in machine learning called *ensemble methods*, and includes methods such as boosting [16], random forests [9] and various "committee machines" [6].

(iii) Integration of several classifiers, each trained on a different source of data.

In our context we are focusing on the third class of methods. The standard method for integrating the results of several classifiers is by a majority vote. A more sophisticated approach is to use a classifier whose job is to integrate the predictions of the various classifiers [6]. Not much work has been done to apply classifier integration methods to protein function prediction. One example of integrating the predictions of several classifiers is described in Refs. [36, 37] and is compared with kernel-based integration in those papers.

See the next section for details. Classifier integration is most useful in cases in which each classifier is available as a black-box, e.g. as is the case for gene finders.

4 Kernel Methods

Recently, a class of algorithms known as kernel methods have become popular in the machine learning community [41, 43] and this popularity has extended into computational biology [42]. A *kernel* is a function that defines similarities between pairs of data objects and a kernel method is an algorithm whose implementation depends on the data only through the kernel. More specifically, a kernel is a similarity measure that satisfies the condition of being a dot product in some space, i.e. $K(x, y)$ can be expressed as $\langle \Phi(x), \Phi(y) \rangle$, where Φ is some possibly nonlinear mapping. This mapping technique has been known for decades, but has gained popularity recently in the context of a particularly powerful classification algorithm, known as the support vector machine (SVM) [8, 12, 50]. The so-called "kernel trick" – mapping data into a higher-dimensional space by means of a predefined kernel function – often results in a problem with more dimensions than examples. The SVM, it turns out, can cope remarkably well with such cases, effectively reducing the curse of dimensionality. Other kernel methods have subsequently been described for classification, regression, clustering, principal components analysis, etc. [43].

Kernel methods provide a coherent framework for data integration. The kernel function provides a form in which to represent a wide variety of data types, including vectors, matrices, strings, trees and graphs. As a kernel method represents data via the kernel function, any data set of n elements can be summarized as an n-by-n matrix of pairwise kernel values. This kernel matrix is a sufficient representation: once it is computed, the original data can be discarded and the kernel method can still perform its function. Furthermore, kernel matrices from different data sources can be combined in a simple kernel algebra, that includes the operations of addition, multiplication and convolution [22]. The simplest way to combine kernels is by adding them: adding kernels is equivalent to concatenating their feature space representations. When the kernels are linear kernels over an explicit vector-space representation this is the same as the vector-space integration described in Section 2. The feature space for the multiplication of kernels is the product of the feature spaces of the kernels. This approach has been used in the context of predicting protein–protein interactions [5].

It is possible to perform vector-space integration with kernel methods. Since kernel methods are sensitive to the scale of each feature, it is often

useful to normalize the features such that they are on a similar scale, e.g. by standardizing each feature. When performing integration at the kernel level, an alternative is to normalize the kernel itself, rather than its feature space representation by using a cosine-like kernel $K'(x,y) = K(x,y)/\sqrt{K(x,x)K(y,y)}$, which is the same as projecting the feature-space representation to the unit sphere.

In two related papers, Pavlidis and coworkers apply a kernel-based data integration technique to the problem of protein function prediction [36, 37]. The authors use kernels to combine microarray expression data with phylogenetic profiles and use the resulting combined kernel to train an SVM classifier to place yeast genes into MIPS functional categories [33]. This kernel-based approach is compared to a vector-space integration scheme, which simply concatenates the two types of data into a single vector, and a classifier integration scheme, which trains two different SVMs and then sums the resulting discriminants. In this case, the primary difference between the vector-space integration scheme and the kernel approach is the use of a third-degree polynomial kernel on each data set prior to integration. The polynomial kernel maps each data set into a higher-dimensional space whose features are all monomials over the original features with degree less than or equal to 3. By performing this mapping on each data set individually, rather than on the concatenated vectors, the method incorporates the prior knowledge that inter-feature dependencies within one data set are more likely to be relevant than dependencies between two different types of data. This prior knowledge is borne out by the results, which show that the kernel-based integration scheme provides better classification performance than either of the other two schemes. Data integration by kernel summation has been applied in several other bioinformatics applications: prediction of protein–protein interactions [5] and prediction of metabolic networks [51]. Prediction of metabolic networks, i.e. associating enzymes with metabolic pathways, can be considered a form of function prediction; prediction of pairwise relationships, or networks, is discussed in detail in Section 5.

Rather than simply adding kernels, one can consider a linear combination of kernels, which can take into account how informative each kernel is. For example, if we know that data set A is more useful (i.e. more relevant or less noisy) than data set B, then we can combine the corresponding kernels as a weighted sum: $K_{AB} = \lambda K_A + K_B$. The only difficulty, of course, is how best to select the data set weighting factor λ. The value of the weighting factor can be set using cross-validation over several choices for its value. This is feasible when combining two kernels. When using a larger number of kernels this is no longer practical and a different approach for weighting the different kernels is required.

Lanckriet and coworkers present a statistical framework for performing kernel-based data integration with weights assigned to each data set [26, 28]. Rather than requiring that the weights be assigned *a priori*, the authors train an SVM and learn the kernel weights simultaneously, using a technique known as semidefinite programming (SDP) [27, 35, 49]. In Ref. [28], this SDP-SVM approach is compared to a previously described Markov random field method for data integration [13] (described in Section 6). Lanckriet and coworkers use the same classification of yeast genes into 13 broad MIPS functional categories and five types of data as [13]: (i) the domain structure of the protein, according to Pfam [45], (ii) known protein–protein interactions, (iii) genetic interactions and (iv) cocomplexed proteins, as identified by the comprehensive yeast genome database, and (v) cell cycle gene expression profiles. Performance is measured using ROC curves [21]. The SDP-SVM approach provides far better performance across all 13 functional classes. A subsequent article [26] applies the same framework to two more yeast classification problems – recognizing membrane proteins and recognizing ribosomal proteins – and provides more details about the SDP-SVM method.

Borgwardt and coworkers propose a kernel method for predicting protein function using protein structure [7]. They represent the structure of a protein as a graph whose nodes are secondary structural elements and whose edges represent proximity in sequence or in 3-D space. The authors propose a kernel that quantifies the similarity between two proteins using the random walk kernel [17], combined with kernels that quantify the similarity between the secondary structural elements of the protein. The proposed kernel thus combines local properties of the protein with the global 3-D structure. They use this kernel with an SVM classifier, as well as with more sophisticated hyper-kernel machinery, to distinguish between enzymes and nonenzymes, and predict the first EC number of an enzyme on a data set used in Ref. [15]. Their more sophisticated approach provides slightly better results than the SVM vector-space integration approach of Dobson and Doig; it is likely that integrating additional structural features into their kernel will provide further improvement.

5 Learning Functional Relationships

Much of the data relevant to predicting the protein function is in the form of a network or can be converted into a network structure. Protein–protein interaction data is an example of such a network: proteins that interact often participate in the same biological process, have a similar localization pattern and, to a lesser extent, have a similar function [5]. Other sources of data that are not directly in the form of a network can be converted into a network

structure. Gene expression data can be represented by a graph whose edges represent comembership in a gene expression cluster or weighted by the correlation between the nodes; sequence data can be similarly converted to a graph by means of sequence similarity scores from algorithms such as Smith–Waterman [44] or PSI-BLAST [2]. Other sources of data for weighting edges include similarity of phylogenetic profiles, gene fusion and cocitation from the literature [29] (see also Chapter 32).

Given several networks of pairwise functional relationships, an important task is to unify those networks into a single network [32]. Marcotte and coworkers demonstrate how to combine pairwise functional relationships from three different sources: correlated evolution using phylogenetic profiles [39], correlated mRNA expression profiles and patterns of domain fusion [31]. The data fusion approach is simple: the authors make a list of all pairs of functionally related proteins in yeast according to each method. This list contains 93 000 pairs of proteins. Functional links that are supported by two out of three of the methods are considered "highly confident" and functional annotations from proteins of known function are then propagated across this high-confidence network. This simple approach yielded functional annotations for more than half of the 2557 yeast proteins that were unannotated at the time.

This simple approach of trusting only predictions that are made by more than one method clearly has drawbacks, especially when some of the contributing methods are more reliable than others or when the methods assign confidence values to their predictions. Lee and coworkers address this issue and propose a framework for unifying the scores associated with different networks [29]. The following function assigns a log-likelihood score to a linkage L between two proteins in the presence of a network E in the context of a particular pathway or annotation:

$$LLS(L|E) = \log \frac{P(L|E)/P(\bar{L}|E)}{P(L)/P(\bar{L})},$$

where $P(L|E)$ are the frequencies of the linkage L observed in the data and $\bar{L}$ are instances where the linkage is not observed.

A similar problem is addressed by the MAGIC system [47]; MAGIC estimates the probability that proteins i and j share a functional relationship. The existence of a functional relationship is modeled using several pairwise relationships between proteins: coexpression, colocalization, physical interaction, genetic interactions and comembership in a complex. The paper proposes a Bayesian network model for estimating the probability of a functional relationship. A Bayesian network is a probabilistic model that represents a probability distribution in a form that makes it amenable to efficient computation by encoding the probabilistic dependencies in the data in the form of a directed

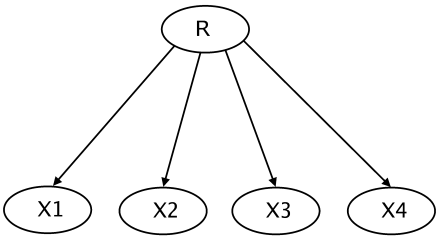

Figure 2 A naive Bayes model: Given that a functional relationship exists between two proteins, the existence of any two relationships between the two proteins (interaction, coexpression, etc.) are independent.

graph (see Refs. [34, 38] for textbooks on Bayesian networks). To illustrate the approach we will consider a very simple model. Let R be the random variable that denotes the existence of a functional relationship and let $X_1, \ldots, X_d$ be the pairwise relationships that serve as evidence for the existence of the relationship. We are interested in the probability $P(R|X_1, \ldots, X_d)$. Using Bayes rule, this probability can be expressed as:

$$P(R|X_1, \ldots, X_d) = \frac{P(X_1, \ldots, X_d|R)P(R)}{P(X_1, \ldots, X_d)}.$$

The naive Bayes model is the assumption that each data source X_i is conditionally independent of the other data sources, i.e. $P(X_i|R, X_j) = P(X_i|R)$ for $j \neq i$ [34, 38]. This assumption enables us to write:

$$\begin{aligned} P(X_1, \ldots, X_d|R) &= P(X_1|R)P(X_2, \ldots, X_d|X_1, R) = P(X_1|R)P(X_2, \ldots, X_d|R) \\ &= P(X_1|R)P(X_2|R)P(X_3, \ldots, X_d|X_2, R) \\ &= P(X_1|R)P(X_2|R)P(X_3, \ldots, X_d|R) \\ \ldots &= \prod_{i=1}^{d} P(X_i|R)\ . \end{aligned}$$

Finally, we have:

$$P(R|X_1, \ldots, X_d) = \frac{\prod_i P(X_i|R)P(R)}{\prod_i P(X_i)}. \tag{1}$$

The classifier resulting from this independence assumption is known as *naive Bayes*. The independence assumption underlying the naive Bayes classifier can be expressed as the directed graph shown in Figure 2. The interpretation of the network structure is that a particular pairwise relationship between two genes, e.g. physical interactions, is a consequence of the existence of a functional relationship between the genes. The Bayesian network suggested by Troyanskaya and coworkers introduces some dependencies between the

various sources of data, but the general dependency structure is similar to the one presented here. The conditional probability distributions at the nodes of the network were determined by surveying a panel of experts in yeast molecular biology. Thus, there is no learning involved in the construction of the system. The pairwise relationships used in MAGIC are matrices whose i, j element is the degree to which protein i and j share a particular relationship. For some data sources this is a binary score, e.g. the proteins coded physically interact or their genes belong to the same gene expression cluster. For other data sources, the score is continuous, e.g. when measuring expression correlation.

Learning of metabolic networks is an example of learning of functional relationships. In this problem, one learns a network whose nodes are enzymes and whose edges indicate that the two enzymes catalyze successive reactions in a pathway. Yamanishi and coworkers [51] integrate many sources of data in the context of kernel methods and consider two approaches to this problem. (i) A "direct" approach: a classifier is trained on positive examples – pairs of enzymes that are known to belong to the metabolic network versus pairs of enzymes that are not part of the network. (ii) "Learning the feature space": before training a classifier, a low-dimensional feature space is computed. This space captures the proximity between enzymes that belong to the metabolic network. Yamanishi and coworkers find that learning the feature space significantly improves the results, and, further, that integration of several kernels based on expression data, phylogenetic profiles, localization and chemical compatibility gives better results than any individual kernel.

6 Learning Function from Networks of Pairwise Relationships

When pairwise relationships between proteins are known, the function of unknown proteins can be inferred using the "guilt by association" rule by looking at the annotations of its neighbors in the network. This rule assigns a functional annotation using a majority vote among the annotations of the neighboring nodes. This method of assigning function is clearly an oversimplification of the problem since it ignores the larger context in which a node appears. An alternative approach is to integrate information across the network, rather than relying only upon local information. In this section, we describe several approaches that consider the network as a whole when making predictions. All the methods predict a single function of interest using a network in which each protein is a node. Binary node labels indicate whether the protein has the function of interest and a third label value can be added to represent proteins with unknown function. The nodes are connected with edges whose weights reflect the degree to which the two proteins are

related. Multiple networks can be used either by merging the networks or by having several networks sharing the annotation variables.

Deng and coworkers proposed a Markov random field (MRF) model to take into account multiple networks of relationships between genes [13]. Relationships such as protein–protein interaction and coexpression are symmetric: no directionality can be assigned to such relationships. Therefore, Bayesian networks that rely on a directed graph to model the dependencies between variables are not readily applicable. MRFs, which represent probabilistic dependencies using undirected graphical models, are therefore a more appropriate modeling choice. The reader is referred to Ref. [38], for example, for an in-depth discussion of MRFs. Deng and coworkers estimate an MRF model for each function of interest, and each protein is assigned a variable X_i with a state of either 1 or 0, depending on whether or not the protein has that function. The joint probability of X, the vector of variables, is written as: $\exp(-U(x))/Z(\theta)$, where x is a value of X, $Z(\theta)$ is a normalization factor that depends on the parameters of the model and:

$$\begin{aligned} U(x) &= -\alpha \sum_{i=1}^{N} x_i - \beta \sum_{(i,j)\in S} \left[(1-x_i)x_j + x_i(1-x_j)\right] \\ &\quad - \gamma \sum_{(i,j)\in S} x_i x_j - \kappa \sum_{(i,j)\in S} (1-x_i)(1-x_j)\,, \end{aligned} \tag{2}$$

where S is the set of edges of the graph, $\alpha = \log(\frac{\pi}{1-\pi})$ and π is the prior probability for observing the function of interest. The first term in Eq. (2) represents the prior probability of observing the configuration x. The rest of the terms represent interactions between neighbors in the network: the first counting the number of neighbors that do not agree on the assignment of function, the second counting the neighbors that share the function of interest and the third counting neighbors that are negative examples.

The training data is a subset of the proteins whose function is known. Using this data, the probability distribution of the function of the rest of the proteins is estimated by conditioning on the state of the known proteins. The probability of an unknown protein having the function of interest can then be obtained by summing over the possible configurations of the rest of the unknown proteins. The authors propose a Gibbs sampling scheme for estimating these probability distributions.

So far we presented the MRF model that uses a single network. When several networks are available, the probability distribution is a product of terms, each of which is of the form (2), sharing the same values of X. The authors also add a component that takes into account the domain composition of the given proteins (see Ref. [13] for the details). The prior probabilities for a protein to be assigned the function of interest is determined using data on protein complexes, according to the fraction of members of the complex that

have that function. When that information is not available, a global prior based on the frequency of the annotation is used. Given a set of possible annotations, one can estimate the probabilities of each annotation. Multiple annotations can then be assigned on the basis of the assigned probabilities. The correlations between different annotations are not taken into account.

The authors apply their method to classify yeast proteins using MIPS annotations combining networks from several sources:

- Physical interactions taken from the MIPS database, including 2448 interactions between 1877 proteins.
- MIPS genetic interactions.
- Comembership in a complex obtained using TAP data, including 232 complexes involving 1088 proteins with known function.
- Cell cycle gene expression data. A network is formed by forming edges between proteins whose expression is above some threshold (0.8 in the paper).

The authors find that combining multiple sources of data improves their performance relative to learning from any single data source.

Karaoz and coworkers propose a general framework for integrating and propagating evidence in functional linkage networks [25]. This approach is a generalization of the "guilt by association" rule which, in essence, repeatedly applies the rule until the network reaches a state that is maximally consistent with the observed data. They begin with a functional linkage network in which the edges are defined by protein–protein interactions and the edge weights are defined by correlating the corresponding mRNA expression profiles. A separate network is defined for each functional annotation (GO term) and each node in the network is assigned a label based upon whether the protein is assigned the current GO term (1), a different GO term in the same GO hierarchy (-1) or no GO term at all (0). The optimization procedure attempts to assign labels (1 or -1) to the zero-labeled nodes so as to maximize an "energy" function. Their energy function is similar to the one used in Ref. [13], but the approach is limited to a single network and, rather than assigning function according to the distribution of a variable, they use a local minimum of the energy function. As described, the method integrates two types of data: one used in the definition of the network topology and the other defines the edge weights. A larger number of sources of data can be integrated by performing a preprocessing step of network integration by one of the methods described in the previous section so that the network or the weighting are computed by more than a single source of data. A similar approach is described in Ref. [11]; to address the situation that a node has no

annotated neighbors and "guilt by association" cannot be applied at the node, their model has a state space that indicates whether "guilt by association" is ready to be applied at a node.

Rather than trying to estimate a probabilistic network model, one can take a more direct approach of making predictions on the protein graph. An example of this approach is found in Ref. [48]. This paper follows the standard machine learning paradigm of optimizing a two-part loss (or fitness) function. This function is composed of an error term that measures how well the predicted function follows the training data (existing annotations) and a regularization term that ensures the "smoothness" (regularization, in machine learning terms) of the predicted biological function. In the context of learning on a graph, smoothness means that adjacent nodes have similar predicted function. As in the previous approaches, the authors define a graph whose nodes are proteins labeled as "$+1$" or "-1", depending on whether the protein is annotated with the function of interest. Let n be the number of proteins, and assume that the function of the first p proteins is known and is given by a vector y with elements equal to ± 1. Proteins with unknown function have $y_i = 0$. They define a variable f_i which is the predicted annotation of node i. The value of f is estimated by minimizing the function:

$$\sum_{i=1}^{p}(f_i - y_i)^2 + \mu \sum_{p+1}^{n} f_i^2 + \sum_{i,j} w_{ij}(f_i - f_j)^2 \,,$$

where w_{ij} is the weight on the edge connecting nodes i and j. The first term is the error term; the rest are regularization terms: the second term ensures that the value of f for unlabeled nodes is bounded and the third term ensures that adjacent nodes have a similar annotation. Setting $\mu = 1$, this expression can be written as:

$$\sum_{i=1}^{n}(f_i - y_i)^2 + \sum_{i,j} w_{ij}(f_i - f_j)^2$$

where we take into account that the last $n - p$ entries of y are zero. In order to facilitate extending this formulation to multiple networks this is written as:

$$\min_{f,\gamma} \sum_{i=1}^{n}(f_i - y_i)^2 + c\gamma, \;\; f^{\mathrm{T}} L f \leq \gamma \,,$$

where L is the Laplacian matrix $L = D - W$, where $D = \mathrm{diag}(d_i)$, $d_i = \sum_j w_{ij}$. Multiple networks are incorporated as:

$$\min_{f,\gamma} \sum_{i=1}^{n}(f_i - y_i)^2 + c\gamma, \;\; f^{\mathrm{T}} L_k f \leq \gamma,$$

where L_k is the Laplacian for network k. Sparsity and a dual formulation of the problem yield efficient algorithms that enable solving the problem

even for large networks. The authors apply their method to predict MIPS categories in yeast (the same data used in Refs. [13, 28]). They use networks based on Pfam domain composition similarity, coparticipation in a protein complex, MIPS physical interactions, genetic interactions and cell cycle gene expression similarity. They obtain similar performance to that of the SDP-SVM method [28] and better than the MRF method [13].

7 Discussion

The methods described in this chapter illustrate that integrating several sources of data provides improved accuracy in protein function prediction. We described several approaches for data integration. Selecting among these various approaches is difficult, because a large-scale experimental comparison of different integration techniques has not been performed. In the end, researchers tend to select the modeling technique that they are most comfortable with.

A related problem for which data integration yields improved classification accuracy is prediction of protein–protein interactions and the related problem of prediction of comembership in a complex. In this domain, examples of vector-space integration include the works described in Refs. [30, 52]. These papers use a collection of features to predict comembership in a complex using probabilistic decision trees and random forests. In these experiments, features include microarray experiment correlations, transcription factor-binding data, localization, phenotype, gene fusion, gene neighborhood and phylogenetic profiles. An example of kernel-based integration is found in Ref. [5] where sequence-based kernels are used in conjunction with kernels based on features such as GO annotations to predict protein–protein interactions. The use of kernel-based classifiers allows the use of a high-dimensional feature-space that cannot be represented explicitly in practice. As in the other examples presented in this chapter, the combined method performs significantly better than methods that use only a single source of data.

References

1 S. F. ALTSCHUL, W. GISH, W. MILLER, E. W. MYERS, AND D. J. LIPMAN.. A basic local alignment search tool. J. Mol. Biol. **215**: 403–10.

2 S. F. ALTSCHUL, T. L. MADDEN, A. A. SCHAFFER, J. ZHANG, Z. ZHANG, W. MILLER, AND D. J. LIPMAN. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. Nucleic Acids Res. **25**: 3389–3402, 1997.

3 A. BAIROCH. The SWISS-PROT protein sequence data bank: Current status. Nucleic Acids Res. **22**(17): 3578–80, 1994.

4 A. BEN-HUR AND D. BRUTLAG. Protein sequence motifs: Highly predictive features of protein function. In:

I. GUYON, S. GUNN, M. NIKRAVESH, AND L. ZADEH, (eds.), *Feature extraction, foundations and applications*. Springer, Berlin, in press.

5 A. BEN-HUR AND W. S. NOBLE. Kernel methods for predicting protein-protein interactions. Bioinformatics **21** (Suppl. 1): i38–46, 2005.

6 C. BISHOP. Pattern Recognition and Machine Learning. Springer, 2006.

7 K. M. BORGWARDT, C. S. ONG, S. SCHOENAUER, S. V. N. VISHWANATHAN, A. SMOLA, AND H.-P. KRIEGEL. Protein function prediction via graph kernels. Bioinformatics **21**(Suppl. 1): i47–56, 2005.

8 B. E. BOSER, I. M. GUYON, AND V. N. VAPNIK. A training algorithm for optimal margin classifiers. In: Proc. 5th Ann. ACM Workshop on COLT, 144–152, Pittsburgh, PA, 1992.

9 L. BREIMAN. Random forests. Machine Learning **45**(1): 5–32, 2001.

10 M. P. S. BROWN, W. N. GRUNDY, D. LIN, N. CRISTIANINI, C. W. SUGNET, T. S. FUREY, JR. M. ARES, AND D. HAUSSLER. Knowledge-based analysis of microarray gene expression data using support vector machines. Proc. Natl Acad. Sci. USA **97**: 262–7, 2000.

11 Y. CHEN AND D. XU. Genome-scale protein function prediction in yeast *Saccharomyces cerevisiae* through integrating multiple sources of high-throughput data. Pac. Symp. Biocomput. **10**: 471–82. 2005.

12 N. CRISTIANINI AND J. SHAWE-TAYLOR. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, 2000.

13 M. DENG, T. CHEN, AND F. SUN. An integrated probabilistic model for functional prediction of proteins. Proc. RECOMB **7**: 95–103, 2003.

14 M. DES JARDINS, P.D. KARP, M. KRUMMENACKER, T.J. LEE, AND C.A. OUZOUNIS. Prediction of enzyme classification from protein sequence without the use of sequence similarity. Proc. ISMB **5**: 92–9, 1997.

15 P. D. DOBSON AND A. J. DOIG. Predicting enzyme class from protein structure without alignments. J. Mol. Biol. **345**: 187–99, 2005.

16 Y. FREUND AND R. E. SCHAPIRE. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55(1)**: 119–39, 1997.

17 T. GÄRTNER, P. FLACH, AND S. WROBEL. On graph kernels: Hardness results and efficient alternatives. In: Proc. 16th Annu. Conf. on Computational Learning Theory and the 7th Annu. Workshop on Kernel Machines, Washington, DC, USA: 129–43. 2003.

18 I. GUYON. Special issue on variable and feature selection. J. Machine Learn. Res. **3** (March) 2003.

19 I. GUYON, S. GUNN, M. NIKRAVESH, AND L. ZADEH (eds.) *Feature Extraction, Foundations and Applications*. Springer, Berlin, 2006.

20 I. GUYON, J. WESTON, S. BARNHILL, AND V. VAPNIK. Gene selection for cancer classification using support vector machines. Machine Learn. **46**: 389–422, 2002.

21 J. A. HANLEY AND B. J. MCNEIL. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology **143**: 29–36, 1982.

22 D. HAUSSLER. Convolution kernels on discrete structures. *Technical Report UCSC-CRL-99-10*, University of California, Santa Cruz, CA, 1999.

23 F. V. JENSEN. *Bayesian Networks and Decision Graphs*. Springer, Berlin, 2001.

24 L. J. JENSEN, R. GUPTA, N. BLOM, ET AL. Prediction of human protein function from post-translational modifications and localization features. J. Mol. Biol. **319**: 1257–65, 2002.

25 U. KARAOZ, T. M. MURALI, S. LETOVSKY, Y. ZHENG, C. DING, C. R. CANTOR, AND S. KASIF. Whole-genome annotation by using evidence integration in functional-linkage networks. Proc. Natl Acad. Sci. USA **101**: 2888–93, 2004.

26 G. R. G. LANCKRIET, T. DE BIE, N. CRISTIANINI, M. I. JORDAN, AND W. S. NOBLE. A statistical framework for genomic data fusion. Bioinformatics **20**: 2626–35, 2004.

27 G. R. G. LANCKRIET, N. CRISTIANINI, P. BARTLETT, L. EL GHAOUI, AND M. I. JORDAN. Learning the kernel matrix with semi-definite programming. In: Proc. 19th Int. Conf. on Machine Learning, Sydney: xxx–xxx, 2002.

28 G. R. G. LANCKRIET, M. DENG, N. CRISTIANINI, M. I. JORDAN, AND W. S. NOBLE. Kernel-based data fusion and its application to protein function prediction in yeast. Pac. Symp. Biocomput.: 300–11, 2004.

29 I. LEE, S. V. DATE, A. T. ADAI, AND E. M. MARCOTTE. A probabilistic functional network of yeast genes. Science **306**: 1555–8, 2004.

30 N. LIN, B. WU, R. JANSEN, M. GERSTEIN, AND H. ZHAO. Information assessment on predicting protein-protein interactions. BMC Bioinformatics **5**: 154, 2004.

31 E. M. MARCOTTE, M. PELLEGRINI, H.-L. NG, D. W. RICE, T. O. YEATES, AND D. EISENBERG. Detecting protein function and protein-protein interactions from genome sequences. Science **285**: 751–3, 1999.

32 E. M. MARCOTTE, M. PELLEGRINI, M. J. THOMPSON, T. O. YEATES, AND D. EISENBERG. A combined algorithm for genome-wide prediction of protein function. Nature **402**: 83–6, 1999.

33 H. W. MEWES, D. FRISHMAN, C. GRUBER, ET AL. MIPS: a database for genomes and protein sequences. Nucleic Acids Res. **28**: 37–40, 2000.

34 R. NEAPOLITAN. *Learning Bayesian Networks*. Prentice Hall, Englewood Cliffs, NJ, 2003.

35 Y. NESTEROV AND A. NEMIROVSKY. *Interior-point Polynomial Methods in Convex Programming: Theory and Applications*, volume 13 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, 1994.

36 P. PAVLIDIS, J. WESTON, J. CAI, AND W. N. GRUNDY. Gene functional classification from heterogeneous data. In: Proc. 5th Annu. Int. Conf. on Computational Molecular Biology, Montreal, Canada: 242–8, 2001.

37 P. PAVLIDIS, J. WESTON, J. CAI, AND W. S. NOBLE. Learning gene functional classifications from multiple data types. J. Comput. Biol. **9**: 401–11, 2002.

38 J. PEARL. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1998.

39 M. PELLEGRINI, E. M. MARCOTTE, M. J. THOMPSON, D. EISENBERG, AND T. O. YEATES. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. Proc. Natl Acad. Sci. USA **96**: 4285–8, 1999.

40 S. ROGIC, B. F. OUELLETTE, AND A. K. MACKWORTH. Improving gene recognition accuracy by combining predictions from two gene-finding programs. Bioinformatics **18**: 1034–45, 2002.

41 B. SCHÖLKOPF AND A. SMOLA. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

42 B. SCHÖLKOPF, A. SMOLA, AND K.-R. MÜLLER. Kernel principal component analysis. Proceedings of the 7th International Conference on Artificial Neural Networks **1327**: 583, 1997.

43 J. SHAWE-TAYLOR AND N. CRISTIANINI. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, 2004.

44 H. O. SMITH, T. M. ANNAU, AND S. CHANDRASEGARAN. Finding sequence motifs in groups of functionally related proteins. Proc. Natl Acad. Sci. USA **87**: 826–30, 1990.

45 E. SONNHAMMER, S. EDDY, AND R. DURBIN. Pfam: a comprehensive database of protein domain families based on seed alignments. Proteins **28**: 405–20, 1997.

46 U. SYED AND G. YONA. Using a mixture of probabilistic decision trees for direct prediction of protein function. *Proc. RECOMB*: 289–300, 2003.

47 O. G. TROYANSKAYA, K. DOLINSKI, A. B. OWEN, R. B. ALTMAN, AND D. BOTSTEIN. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *S. cerevisiae*). Proc. Natl Acad. Sci. USA **100**: 8348–53, 2003.

48 K. TSUDA, H. J. SHIN, AND B. SCHÖLKOPF. Fast protein classification

with multiple networks. Bioinformatics **21**: ii59–65, 2005.

49 L. VANDENBERGHE AND S. BOYD. Semidefinite programming. SIAM Rev. **38**: 49–95, 1996.

50 V. N. VAPNIK. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

51 Y. YAMANISHI, J.-P. VERT, AND M. KANEHISA. Supervised enzyme network inference from the integration of genomic data and chemical information. Bioinformatics **21**: i468–77, 2005.

52 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**: 38–53, 2004.