Methods of Genomic Data Fusion: An Overview

Konstantin Tretyakov (kt@ut.ee)

May 4, 2006

Abstract

The abundance of high-throughput biological data, such as microarray or protein-protein interaction assays has lead to a need for new methods of data analysis, that could infer useful information from large amounts of very noisy and indirect measurements. One solution could be provided by *data fusion*. *Data fusion* is a relatively recent term describing machine learning methods that can integrate disparate datasets and thus reduce the overall noise, increase statistical significance as well as leverage the interactions and correlations between the datasets to obtain more refined and higher-level information. This paper gives a very brief overview of two very general and well-developed approaches to data fusion—Bayesian networks and kernel methods. It may therefore be of interest to a reader not previosly familiar with these terms, willing to grasp the most basic understanding of the underlying ideas.

1 Introduction

High-throughput Methods

High-thoughput methods are playing an increasingly important role in the contemporary biological research. Microarray technologies allow measurements of thousands mRNA transcripts, shotgun-sequencing has resulted in complete genomes of several organisms, ChIP-on-chip assays, mass-spectrometry and yeast-2-hybrid screening provide insights into protein-DNA and protein-protein binding properties, etc. Enormous amounts of data have already been produced using these methods, and a lot of hope is being put in the possibility of discovering the relevant biological information from this data.

All the high-throughput methods, however, share several common traits, that make data analysis rather complicated. Firstly, the measurements are usually *indirect*. For example, mRNA microarrays measure the amount of certain mRNA transcripts for each gene, giving only an indirect indication of the true value of interest—the amount of protein produced for each gene. ChIP measurements can only reveal large regions in the DNA where a given protein would bind, whereas we are interested in the specific positions. Y2H screens can tell whether two proteins can participate in the same complex, while we would like to know whether they can bind to each other, etc. Secondly, the data produced is usually *very noisy*. It is common that two microarray datasets produced by different laboratories would have correlation close to zero—so large is the noise in the data. At last, the overall methodology of first taking some generic measurements and later trying by any means to figure out the underlying patterns is not very well supported by pure statistics and requires the use of much more recent techniques of *machine learning*.

Data Fusion

Data fusion is one technique that is especially useful for the analysis of high-throughput experiment data. The idea is that incorporating more than one genomic dataset in the analysis may be beneficial, by reducing the noise, as well as improving statistical significance and leveraging the interactions and correlations between the datasets to obtain more refined and higher-level information. Considerable work has been devoted to the problem of genomic data fusion. The general approaches can be roughly divided into three classes: *early*, *intermediate* and *late* integration.

The simplest idea is that of late integration: each dataset is treated independently and sepa-

rate inferences are made. The results obtained in this way from each dataset can then be converted to a common form and somehow merged. Such an approach has been used, for example, to validate gene expression and protein-protein interaction data [1, 2, 3], to infer protein function [4, 5], or to analyze the gene regulatory graphs resulting from different datasets [6, 7].

When late integration means combining the *outputs*, *early* integration methods combine heterogeneous *input sources* in a consistent manner, and then rely on a single analysis method. The whole problem is then reduced to finding a convenient common representation for different types of data. For example, if every dataset in question can be represented as a set of vectors, it is often possible to concatenate the vectors from the datasets and feed the combined datapoints into a general-purpose classification or regression procedure. Distance metrics, kernels, probability distributions or graphs are other options for a common data format. Instances of such an approach have been used, among others, for protein function prediction [8, 9, 10, 11, 12], to infer protein-protein interactions [13, 14] and protein complexes [15, 16].

At last, *intermediate* integration corresponds to methods that don't clearly fall under the title of either early or late integration. These usually use a custom statistical procedure for combining specific kinds of data in data-dependent ways. For example, rather common are the approaches that infer regulatory motifs by clustering genes by their expression profiles and searching for overrepresented substrings in the DNA sequences of the genes in a cluster. Another common scenario is functional annotation of gene clusters by analyzing them for overrepresented GO categories. Other examples include methods for locationg transcriptional modules [17] and transcription factor bindings [18].

Of the three named classes, early integration methods seem particularly interesting for they often scale to integrating nearly any number of *nearly arbitrary* kinds of data in a systematic and consistent manner. This article briefly reviews two approaches to early integration: Bayesian networks and kernel methods.

2 Bayesian networks

Bayesian network is a graphical probabilistic model, that allows to combine disparate evidence with prior knowledge of interrelations within the data to produce a single probabilistic answer. The idea is easy to explain on a series of examples. Consider the question of whether two given proteins interact or not. Suppose we cannot determine this fact directly, but we have microarray expression profiles of the genes at our disposal. It might be reasonable to assume that interacting genes are more likely to have similar expression. We can describe this belief as a set of conditional probabilities:

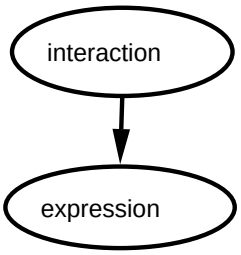
$$P(\text{similar expression} \,|\, \text{interaction}) := P(E \,|\, I) = 0.7$$
$$P(\text{similar expression} \,|\, \text{no interaction}) := P(E \,|\, -I) = 0.4$$

Once we determine that two genes have similar expression, we can calculate, using the Bayes rule, the probability of them interacting:

$$P(I \,|\, E) = \frac{P(E \,|\, I)P(I)}{P(E)} = \frac{P(E \,|\, I)P(I)}{P(E \,|\, I)P(I) + P(E \,|\, -I)P(-I)}$$

where $P(I)$ indicates our *prior belief* in the fact that two proteins interact. Choosing $P(I) = 0.5$ would mean that we are completely uncertain about whether the interaction takes place or not. By plugging this prior in the formula, we obtain $P(I \,|\, E) \approx 0.64$, which indicates some degree of confidence in the fact that proteins indeed interact.

We could graphically display the employed computation scheme in the following way:



The graph indicates the causality structure that, we believe, underlies the phenomenon of interest: *interaction* causes *similar expression*. We can plug-in our knowledge of similar expression and obtain the resulting probability of interaction.

But suppose now that besides the microarray dataset, we also posess data of a Y2H screening experiment. Again, it is reasonable to assume that interacting proteins are highly likely to report Y2H binding:

$$P(\text{Y2H binding} \,|\, \text{interaction}) := P(Y \,|\, I) = 0.9$$
$$P(\text{Y2H binding} \,|\, \text{no interaction}) := P(Y \,|\, -I) = 0.3$$

Suppose the Y2H data reported no binding. Can we combine this evidence with previous microarray data to refine our knowledge of interaction probability?

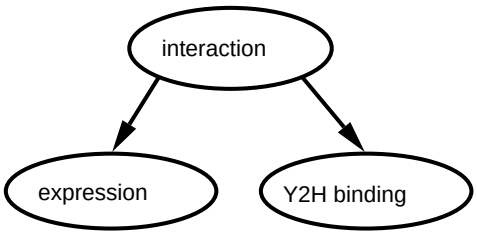
$$P(I \,|\, -Y, E) = \frac{P(-Y, E \,|\, I) P(I)}{P(-Y, E)}$$

As you see, not unless we know the combined conditional distribution $P(Y, E \,|\, I)$. However, it may be reasonable to *assume* that Y2H binding results and microarray results are *conditionally independent*, given the knowledge about their interaction. That is, if we *knew* whether proteins really interacted or not, the Y2H binding result would not provide any additional information about the possible microarray result. In this case,

$$P(-Y, E \,|\, I) = P(-Y \,|\, I) P(E \,|\, I)$$

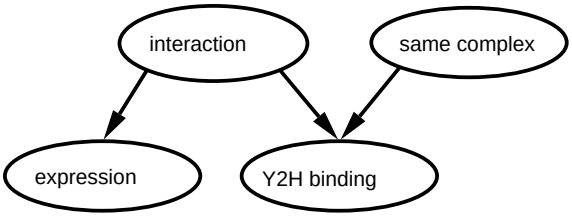
and we can continue the calculation to obtain $P(I \,|\, -Y, E) = 0.2$. As you see, the negative Y2H evidence strongly influenced our posterior belief.

As before, we can visualize the structure of our beliefs as a directed graph:



The graph reads: interaction *causes* certain expression and Y2H results, but these two types of results are *independent*, if their cause (i.e. *interaction*) is known. In order to use the graph, we feed in our knowledge of actual expression and Y2H results, propagate the probabilities and obtain the *posterior* probability of proteins interacting.

Now we might note that Y2H results are actually more influenced by the fact that genes can participate in the same complex, we could therefore add another causal relationship:



In this case the Y2H result depends on both the interaction and the ability to participate in the same complex. To specify such network, we should provide the probabilities $P(Y \,|\, I, C)$ for each pair (I, C) (where C stands for *participation in the same complex*). We could use the network in the same manner as before: by feeding the known inputs Y and E and receiving the outputs I and C. In fact, we can feed any set of known variables as inputs and obtain the posterior distribution of the remaining variables.

Much more complex networks can be constructed in this manner, that would be able to combine data in a variety of sophisticated ways. And although in most common applications a Bayesian network is designed with a help of an expert, algorithm exist that can automatically infer network topology from data. The major drawback of Bayesian networks is their computational cost: evaluation of networks with complicated topologies can take exponential time. However, evaluation of simple tree-like networks is always effective.

The topic of Bayesian networks is very wide: extensive literature exists covering the (rather nontrivial) questions of effective network evaluation, training and statistical inference. A simple overview can be obtained in [19, 20, 21] as well as in special textbooks on the topic.

Bayesian Networks for Data Fusion

It should be clear now that the Bayesian network formalism is very convenient for integrating disparate genomic datasets. And, indeed, some work has been done in this area. For example, Jansen et al. [13] designed a Bayesian network for prediction of protein-protein interactions, that combines experimental interaction data, mRNA expression

and GO annotations into predictions, whose reliability is superior to those made on the basis of any single dataset alone.

In another similar work by Troyanskaya et al. [12], a tree-like Bayesian network was designed for purposes of gene function prediction. The network combined different experimental measurements of coexpression, colocalization, physical and genetic associations to predict functional associations of genes. Again, it turned out that results of the combined analysis allowed to produce much more confident functional gene annotations.

3 Kernel Methods

Several machine learning methods, such as linear regression or linear classification, although initially defined as methods for classifying vectors from $\mathbb{R}^m$, are actually applicable to a much wide range of datatypes. Take, for example, ordinary linear regression. The problem is often stated in a way similar to the following:

Given a set of labeled vectors

$$\{(\mathbf{x}_n, y_n), \mathbf{x}_n \in \mathbb{R}^m, y \in \mathbb{R}, n \in 1 \dots N\},$$

find a linear discriminator function

$$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle,$$

that minimizes the sum of error squares

$$\sum_n (f_{\mathbf{w}}(\mathbf{x}_n) - y_n)^2.$$

It is possible to show that the solution vector $\mathbf{w}$ can be expressed as a linear combination of initial datapoints:

$$\mathbf{w} = \sum_n \alpha_n \mathbf{x}_n \,,$$

and the resulting function can thus be expressed as

$$f_{\boldsymbol{\alpha}}(\mathbf{x}) = \left\langle \sum_n \alpha_n \mathbf{x}_n, \mathbf{x} \right\rangle = \sum_n \alpha_n \langle \mathbf{x}_n, \mathbf{x} \rangle,$$

which allows to rephrase the original problem in terms of values α_n:

> Given a set of labeled vectors, find such $\boldsymbol{\alpha}$ for which the function $f_{\boldsymbol{\alpha}}$ has the least possible sum of error squares.

It is notable, that in this new formulation, the fact that $\mathbf{x}_n$ are actually *vectors* is only needed to make sense of the inner product operation $\langle \cdot, \cdot \rangle$. That is, the formulation now actually admits *any* datatype, for which we can define a notion of *inner product*, which in this context is often referred to as a *kernel function*. The case of linear regression is not exceptional—pretty much any *linear* algorithm can be *kernelized*, i.e. represented in such a way that the only operation performed with the datapoints is evaluation of the inner product.

Surprisingly, several different kernel functions have been defined even for such complex datatypes as strings or graphs. One simple example of an inner product defined on strings is given by the *k-mer kernel*: we can transform each string to a vector where each element counts the number of occurences of a certain k-mer in the string and then calculate inner products of such vectors. A kernel must not necessarily be defined as an explicit inner product of some vectorized form of the data. It is possible to show that *any positive semidefinite function* $K(\mathbf{x}, \mathbf{y})$ corresponds to a certain inner product and is thus usable as a kernel.

Therefore, *kernel* is a convenient common representation for many datatypes. Moreover, different kernels can be *combined* in a variety of ways. It is possible to show that for any two valid kernels $K_1(\mathbf{x}, \mathbf{y})$ and $K_2(\mathbf{x}, \mathbf{y})$, their sum and product is also a valid kernel.

The general topic of kernel methods is wide and deep, and in no way is it possible to cover even a considerable bit of it in a report like this. For further information the reader is referred to excellent textbooks by Schoelkopf and Smola [22], Shawe-Taylor and Cristianini [23, 24].

Kernel Methods for Data Fusion

The properties of kernels named above lead to a natural way of using kernel methods for data fusion: express the different views of the data as kernel functions K_1, K_2, …, K_k, combine them in a single kernel by taking, say, a weighted sum, and use this kernel in a suitable classification or regression algorithm. This strategy was employed in a work by Pavlidis et al. [11] for inferring gene functional classifications using the support vector machine (SVM) linear classification algorithm. For each gene, the authors had two disparate datasets

at their disposal—microarray expression profile and phylogenetic conservation data. A separate kernel was defined for each type of data, the kernels were added together and used together with SVM for predictions. The strategy of combining kernels was reported to perform considerably better than simple late integration of results.

Even more advanced method is used in the work by Lanckriet et al. [10], where an expression profile kernel, 4 sequence-based and 2 protein-interaction-based kernels were used to predict membrane proteins. The kernels were combined by forming a linear combination of the form $\sum_k \lambda_k K_k$, and it is notable that the authors developed a significant modification of the SVM training algorithm, that automatically determines the best values for the weights λ_k. Naturally, the performance of the combined kernel significantly surpassed that of any individual kernel.

4 Summary

The article reviewed two very different approaches to data fusion: Bayes nets and kernel methods. Both approaches, are truly generic, have well-developed background theory, a history of successful applications and a considerable base of available software tools. Not much attention has been given to actual results obtained by these approaches, mainly because the author sees more value in describing the main concepts, than specific applications. Also, the given description of the concepts is as minimal as possible because attempts to present them in better depth would result in significant increase in the length of the text. The article does not aspire to give an exhaustive overview of all available data fusion methods: rather generic approaches such as clustering or decision trees have been left out. The two selected topics just seemed to the author to be both mathematically interesting and practically very promising.

References

[1] H. Ge, Z. Liu, G. M. Church, and M. Vidal. Correlation between transcriptome and interactome mapping data from Saccharomyces cerevisiae. *Nat Genet*, 29(4):482–6, Dec 2001.

[2] A. Grigoriev. A relationship between gene expression and protein interactions on the proteome scale: analysis of the bacteriophage T7 and the yeast Saccharomyces cerevisiae. *Nucleic Acids Res*, 29(17):3513–9, Sep 2001.

[3] Christian von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G Oliver, Stanley Fields, and Peer Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417(6887):399–403, May 2002.

[4] 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(6757):83–6, Nov 1999.

[5] Ronald Jansen, Ning Lan, Jiang Qian, and Mark Gerstein. Integration of genomic datasets to predict protein complexes in yeast. *J Struct Funct Genomics*, 2(2):71–81, 2002.

[6] A. Nakaya, S. Goto, and M. Kanehisa. Extraction of correlated gene clusters by multiple graph comparison. *Genome Inform Ser Workshop Genome Inform*, 12:44–53, 2001.

[7] Amos Tanay, Roded Sharan, and Ron Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18 Suppl 1:S136–44, 2002.

[8] Minghua Deng, Ting Chen, and Fengzhu Sun. An integrated probabilistic model for functional prediction of proteins. *J Comput Biol*, 11(2-3):463–75, 2004.

[9] Jyotsna Kasturi and Raj Acharya. Clustering of diverse genomic data using information fusion. *Bioinformatics*, 21(4):423–9, Feb 2005.

[10] G.R.G. Lanckriet, N. Cristianini, M.I. Jordan, and W.S. Noble. Kernel-based integration of genomic data using semidefinite programming. In K. Tsuda and J.-P. Vert, editors, *Kernel Methods in Computational Biology*. MIT Press, 2004.

[11] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Stafford Noble. Learning gene functional classifications from multiple data types. *J Comput Biol*, 9(2):401–11, 2002.

[12] Olga G Troyanskaya, Kara Dolinski, Art B Owen, Russ B Altman, and David Botstein. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces cerevisiae). *Proc Natl Acad Sci U S A*, 100(14):8348–53, Jul 2003.

[13] Ronald Jansen, Haiyuan Yu, Dov Greenbaum, Yuval Kluger, Nevan J Krogan, Sambath Chung, Andrew Emili, Michael Snyder, Jack F Greenblatt, and Mark Gerstein. A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302(5644):449–53, Oct 2003.

[14] Yanjun Qi, Judith Klein-Seetharaman, and Ziv Bar-Joseph. Random forest similarity for protein-protein interaction prediction from multiple sources. *Pac Symp Biocomput*, pages 531–42, 2005.

[15] Michael A Gilchrist, Laura A Salter, and Andreas Wagner. A statistical framework for combining and interpreting proteomic datasets. *Bioinformatics*, 20(5):689–700, Mar 2004.

[16] Lan V Zhang, Sharyl L Wong, Oliver D King, and Frederick P Roth. Predicting co-complexed protein pairs using genomic and proteomic data integration. *BMC Bioinformatics*, 5:38, Apr 2004.

[17] T. De Bie, P. Monsieurs, K. Engelen, N. Cristianini, B. De Moor, and K. Marchal. Discovering transcriptional modules from motif, chip-chip and microarray data. January 2005.

[18] Konstantin Tretyakov. A linear model of genetic transcription regulation that combines microarray and dna sequence data. Bachelor's thesis, 2005.

[19] Eugene Charniak. Bayesian networks without tears. Available in the Internet.

[20] Kevin P. Murphy. An introduction to graphical models. Available in the Internet, 2001.

[21] Swen Laur. Tõenäosuste leidmine bayesi võrkudes. Andmekaevanduse uurimisseminar MTAT.03.169, 2003.

[22] Schoelkopf and Smola. *Learning with Kernels*. MIT Press, 2002.

[23] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[24] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-base learning methods*. Cambridge University Press, 2000.