IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. IV (Nov.-Dec. 2016), PP 147-154
www.iosrjournals.org

Kernel methods for gene function prediction

Nguyen Quynh Chi[1], Do Thi Bich Ngoc[1]
[1](*Posts and Telecommunications Institute of Technology, Hanoi, Vietnam)*

Abstract : *This paper proposes an approach using kernel methods to gene function prediction of combining many types of data to get a better performance than the prediction performance of each individual type of data. Some experiments with microarray data and sub-cellular localization data have been implemented. The results show that when we predict some biological process the combination of different types of data using kernels predicts with a better performance than each individual type of data and microarray data with fast fourier transformation gives better prediction performance than original microarray data in some cases.*

Keywords - *kernel methods; gene function prediction; biological process; support vector machine, microarray data, sub-cellular localization data*

I. Introduction

The key task of functional genomics is the determination of biological function for genes and gene products from genomic information. As the amount of genomic data grows steadily and is more and more available, computational functional genomics becomes both possible and necessary. Computational predictions can make experimental determination easier. In machine-learning community, the prediction task belongs to a general problem called classification, which is provided efficient methods to get a high performance. Therefore, machine-learning methods can be considered as one way to do predictions. There have been some works on gene function prediction basing on some machine learning methods like decision trees and Bayes networks. Recently, the existence of kernel methods has resulted in a large number of its application in computational biology in general, and functional gene prediction in particular. This is due to a kernel method, called Support vector machine (SVM). SVM has been known to behave well with noisy and multi-dimensional data, which are the typical properties of biological data. So, kernel methods like SVMs are a promising approach to gene function prediction. Moreover, with the fact that the various kinds of genomic data are available and the fact that kernel methods seems to be a natural way for combining different types of data since they map all data from their own space into a common representation as kernel matrix of real numbers, our approach to this problem is combining many types of data to get a better performance than the prediction performance of each individual type of data. Apart from investigating how to combine types of genomic data, we also try a new kernel for microarray data. We will use kernel methods like SVM for this classification problem and curve method. Some experiments with microarray data and sub-cellular localization data have been implemented and we have got some rather good results. The gene function prediction problem, here, is predicting some biological processes with the annotation of gene ontology. The experiments show that when we predict some biological process by using both individual type of data and combination of different types of data, microarray data with fast fourier transformation gives better prediction performance than original microarray data in some cases. They also illustrate that the combination of different types of data predicts with a little better performance than each individual type of data. The structure of this paper is the following. The part two will present the previous approaches to gene function prediction. Part three will present the computational gene function prediction using kernel methods. Our experiments and results will be described in the part four.

II. related works

In this section, we would like to look at the existing works on functional gene prediction. The first step in function prediction for a new gene is usually to do a sequence comparison against a database of known sequences [1]. Unfortunately, this is sometimes as far as the determination of function goes, and many genes are left annotated as "putative" or "potential" without any indication of where that information came from. If the original sequence was annotated wrongly, then this error may be propagated through the databases though future gene annotation by sequence comparisons. [2] review-ed the process of computational prediction of function from sequence at that time, in all the different stages, from studies of nucleotide frequencies and gene finding, to proteomics and interdependencies of genes. The following list gives some idea of the range of recent works in computational functional genomics.

- Improved sequence similarity search algorithms [3, 4, 5, 6, 7]. Machine learning, intermediate sequences, better indexing schemes and new understanding of the relationship between sequence similarity and function can all be used to improve homology searches.

- Identification of motifs, alignments, protein fingerprints, profiles and other sequence based features that can be used to infer function [8, 9, 10]. Conserved regions of amino acids (motifs), multiple alignments, protein fingerprints and profiles can all be used to characterize protein families, which are likely to share common function.
- Sequence comparison to clusters of orthologous proteins can indicate function [11, 12]. Orthologous genes are homologous genes that have diverged from each other over time as a consequence of speciation. Orthologous genes typically have the same function and so comparison to collections of orthologs can indicate function.
- Microarray expression analysis [13, 14, 15, 16, 17, 18, 19] is one of the most popular methods of functional genomics. Analysis of expression data can be used to infer similar functions for genes which show similar expression patterns. Most expression analysis uses unsupervised clustering, but other methods have also been tried. Rough sets have been used to predict gene function from human expression data using the Gene Ontology classification [20] and the Rosetta toolkit. Rosetta generates if-then rules using rough set theory, and has been used in several medical applications [21].
- Computational prediction of protein secondary structure was in [22, 23, 24]. Structure is used as an intermediate step to predicting the function (the 3-dimensional structure and shape of a protein is very pertinent to its function).
- Combined approaches were indicated in [25, 26]. Many approaches to functional genomics can now make use of several sources of data, including protein interaction data and expression data. [27] used C4.5 from the Weka package to predict SWISSPROT "keyword" field for proteins, given data about the taxonomy, INTERPRO classification, and PFAM and PROSITE patterns.
- Naive Bayes, C4.5 and Instance Based Learning were used to predict enzyme classification from sequence [28]
- Several studies have applied machine learning methods to data from biological experiments to infer functional similarities among genes or directly predict function for unknown genes [29, 30, 31].
- Work on ontologies and schemes for defining and representing function [32, 33, 34] is making progress towards standardizing and understanding function. Along with the use of computers comes a need to make terms and definitions rigorous and well defined. If we are to use computing to determine function then the concept of function must be defined first. The reason that machine learning can be used to predict gene functions is that the prediction task belongs to a general problem called classification in machine learning community. Our approach concerns to a class of machine learning algorithms called kernel methods. To predict functions for unknown genes and gene products, we use computational methods basing on kernel methods like support vector machines and basing on diverse large-scale genomic data. In the next part, we will show how kernel methods really work in computational gene function prediction.

III. Kernel Based Approach To Computational Gene Function Prediction

During past 15 years, SVMs [49] have been applied broadly in the field of computational biology, to pattern recognition problems including not only functional classification of promoter regions but also other problems as protein remote homology detection, microarray gene expression analysis, recognition of translation start sites, prediction of protein-protein interaction. There are two reasons causing SVMs have been the state-of-art machine learning methods in computational biology. First, many biological problems correspond to high-dimensional, noisy data, to which SVMs are known to behave well compared to other statistical or machine learning methods. Second, SVM, one kind of kernel methods, can easily handle non-vector inputs, such as variable length sequences or graphs by using knowledge-based kernel functions to map them to a feature space. Also, many kernels for different kinds of genomic data are available. First, there are various kinds of string kernels for sequence data such as: fisher kernel deriving the feature representation for an SVM classifier from a generative model for a protein family; the string kernels basing on spectrum of a sequence for classification of protein sequence data; the string kernels basing on the detection of local alignments to compare biological sequences; mismatch kernels are suitable for matching discrete objects such as strings and trees (the algorithms in [35]); and pairwise kernels comparing two sequences. Second, there is a family of kernel called label sequence kernels for graph data. Third, diffusion kernels are designed for vertices of a graph such as a metabolic gene network. Fourth, there is a kernel for protein structure prediction. In addition, [36] introduced "convolution kernels", a framework for handling discrete data structures by kernel methods in which objects are decomposed into parts and kernels are defined in term of the (sub) kernels between parts. Some motif kernels basing on a set of motifs on the promoter regions of sequences also were proposed for protein classification. The following works will give us some ideas about the recent functional prediction by using kernel methods like SVMs.

- Classification of yeast genes into functional categories [37]. This is the first application of SVMs to microarray data including 79 microarray experiments, each measuring the activity of approximately 6000

genes. The five functional classes were selected from the MISP yeast genome database. The SVMs using either an RBF or third degree polynomial kernel produced the best performance on this task in comparison with a collection of traditional machine learning techniques, including Fisher's linear discriminate, C4.5, Parzen windows and MOC1.

- Functional classification of promoter regions [38]. The functional roles of proteins can be determined by analyzing DNA sequence in the upstream of the corresponding gene. This region contains the switching mechanism that controls when the gene is turned on or off. The fisher kernel was applied to the problem of classifying genes. This work assumes that genes with similar switching mechanism are likely to have similar functional roles. This method predicted successfully membership in two groups of co-regulated genes in yeast.
- Prediction of protein function from phylogenetic profiles [39]. The protein function can be determined via sequence comparison with other species. A kernel function basing on phylogenetic profiles was used for this problem and performed significantly better than using a simple dot product. The phylogenetic profile is a bit string representation of a protein, in which each bit corresponds to one species for which the complete genome is available. The bit is 1 if the protein has a close homology in that species, and 0 otherwise. So the phylogenetic profiles contain the part of the evolution history of a given protein.
- Recently, integration of different types of genomic data in a single model using learning methods such as SVMs and Bayes nets has showed promising improvements in functional gene prediction [40, 41, 42, 43, 44]. We present here some forms of heterogeneous data combination, using kernel methods like SVMs:
- [40] combined microarray gene expression data and phylogenetic profiles by summing kernel matrices to recognize functional categories of yeast data. They compared three different techniques for combining these data: early integration in which the two kinds of data are concatenated; intermediate integration, in which two kernels of two kinds of data are computed separately, then added; late integration; in which two SVMs are trained separately and their discriminate scores are added. Intermediate integration provides the best results.
- [44] integrated gene expression profiles with prior knowledge of a metabolic network, representing pathways of proteins that operate upon one another in the cell. They hypothesize that gene expression patterns are more likely to be shared by genes that are close to one another in the metabolic network. Two kinds of data are encoded into kernel functions and these functions are combined using canonical correlation analysis. An SVM trained from the combined kernel performs significantly better than an SVM trained only on expression data.
- [43] provides a method to integrate heterogeneous genomic data by summing a collection of kernel matrices, one per each dataset, similar to the work of [40]. However, in this case, each matrix is weighted and the authors demonstrated how to optimize simultaneously the hyperplane selection and the selection of kernel weights. They used four types of data: amino acid sequences, hydropathy profiles, gene expression data and known protein-protein interaction to solve the problem of predicting membrane proteins. An SVM used to train all kinds of data is much better than the SVM trained on any single type of data and better than the existing algorithms for membrane protein prediction.

From the discussion above, kernel methods are really a natural way to combine heterogeneous data since they map each kind of data from its own space to a common feature space so different kinds of data easily combine to solve a problem. In addition, we can see that kernel methods like SVMs are really promising methods for functional gene prediction, especially at the time many different kinds of large-scale genomic data and kernels are available and increasing. Therefore, using kernel methods for computational functional gene prediction with heterogeneous data is a promising and interesting research direction.

IV. experiment and evaluation

4.1. Data for experiments

We used cellular component data or localization data. Each cell contains many compartments, which are also called organelles. In each compartment, a cell maintains different concentrations of relevant molecules. This way, the compartmentalization allows a cell to perform diverse tasks and chemical reactions that require different environments efficiently. Since each type of compartment is devoted to different tasks in the cell, each requires a distinct set of proteins to perform the subtasks. In order to save the resources, proteins are specially delivered to the organelles that require them. Consequently, many proteins contain the signals that specify their destination. These signals can either be entire peptides or characteristic surface patches of a folded protein. There are also default destination when the signals are absent: proteins showing no signal stay in cytosol. The subcellular localization is obviously closely related to the function of the protein. The cellular component ontology describes this localization data, containing locations, at the levels of subcellular structures and macromolecular complexes. Generally, a gene product is located in or is a subcomponent of a particular cellular

component. The cellular component ontology includes multi-subunit enzymes and other protein complexes, but not individual proteins or nucleic acids. Cellular component also does not include multicellular anatomical terms. The cell is defined in Gene Ontology (GO) as all components within and including the plasma membrane and any external encapsulating structures, such as the cell wall and the cell envelope. A cellular component should include more than one gene product; complexes of one gene product with a cofactor. All complexes in the component ontology should be given parentage under the general term protein complex. As GO cellular component terms describe locations where a gene product may act, rather than physical features of proteins or RNAs, the terms integral membrane protein and peripheral membrane protein are present only as non-exact synonyms. GO distinguishes classes of membrane-related location: extrinsic to membrane; intrinsic to membrane. Then each of these terms can have child terms referring to specific membranes. The DAG structure is as follows:

membrane
[p] intrinsic to membrane
---[i] anchored to membrane
[p] extrinsic to membrane

We also used expression data to perform the experiments we exploited the classic microarray data from [45], which included 4 different experiments measuring cell-cycle expression levels in the S. cerevisiae genome: alpha-factor based synchronization, cdc15-based synchronization, cdc28-based synchronization, elutrition-based synchronization. Gene expression data usually come in the form of a matrix of expression levels for a number of genes in a range of cell samples. The expression matrix from [45] is available on line at http://cellcycle-www.stanford.edu/. We took only two different experiments cdc15 and cdc28 for experiments and forms them into two different datasets. There are two essential facts to the proper analysis of microarray data. First, the background noise is present due to the properties of measurements. If the mean background is estimated and substracted, the resulting expression levels may become negative for some genes. Although true expression levels cannot be negative, statistical work seems to suggest that such values should not be censored by setting them to zero or a small positive value; instead, variance-stabilizing transformations may be used. Second, due to varying amounts of mRNA per cell, the results obtained with the different microarrays or for different samples are not likely to be on the same scale. Normalization should be therefore applied. With the microarray data from [45], the data set is quite large and a lot of the information corresponds to genes that do not show any interesting changes during the experiment. To make it easier to find the interesting genes, the first thing to do is to reduce the size of the data set by removing genes with expression profiles that do not show anything of interest. We use a number of techniques to reduce the original expression profiles to some subset that contains the most significant genes with two following steps. The first step is inferring the missing values in expression matrix where the expression level is marked as unknown (NaN). Our approach is using an interpolation method but only if a NaN is surrounded by not NaNs values and deleting the rows with more than two NaNs maintaining the consistency of the dataset. We tried several methods, for each of them the accuracy is tested by removing one known value, doing interpolation with this method and then comparing the new value to the original one. The linear interpolation method got the best result, so we chose the linear interpolation for this step. The second step is smoothing the cdc15 and cdc28 dataset. We doubled the number of points in each row of expression values in each dataset. Similar to the first step, we need to identify the parameters for smoothing method. They were selected by testing the correlation between the expression matrix of cdc15 and that of cdc28 datasets. Now, the datasets are ready for experiments. In addition, with the motivation of seeing how similar the genes are basing on their expression data to cluster, we also do fast fourier transformation (FFT) analysis on this kind of dataset. We can do FFT on this kind of data since this data is cell-cycle expression levels, as time series data, which has a periodical pattern. In FFT analysis, each signal pattern is composed by sinusoid signals, each at a different frequency. After FFT, a signal related to expression data of a gene is transformed to a number of frequencies with their magnitude (amplitude) in a frequency spectrum that related to the sinusoid component of the original signal. We make an assumption that if two genes have a similar pattern in spectrum, we consider them having similar expression pattern, so they may have the same function. The FFT extends the definition of similarity of expression pattern to the similarity of frequency spectrum pattern of two genes. We do the FFT analysis to cdc15 and cdc28 datasets during the process (discussed above) of finding the subset of significant genes as the following. After the interpolation step, data is transformed using FFT, then the data is smoothed using the same smoothing technique as we described with original expression values. The two new datasets are called fft15 and fft28. To compare the strength of correlation between the two new data with the original data, we need to do inverse FFT with the new data to get the compatible data to the original one. We tested the correlation between cdc15 and cdc28 data by applying some kernel functions as linear, RBF and polynomials. The results show that with FFT analysis, two datasets are more correlated than the original ones and the Gaussian distribution (i.e. RBF kernel function) give the best correlation. Therefore, we will use both the datasets with FFT and the original ones to do our experiments and compare the results.

4.2 Kernels for datasets

With the expression data, we will apply some simple kernels: polynomial with some degree, linear, and RBF kernels for the prediction task. With the data of cellular components annotation, we use the kernel that measures the similarity of the GO annotation of a pair of proteins [46]. The feature space for the GO kernel is a vector space with one component for each node in the directed acyclic graph in which GO annotations are represented. Let the annotations (nodes in the GO graph) that are assigned to protein p be denoted by L_p. Note that, in GO, a single protein can be assigned several annotations. A component of the vector corresponding to node a is nonzero if *a* or a parent of *a* is in L_p. We consider two ways to define the dot product in this space. When the non-zero components are set equal to 1, then when each protein has a single annotation, and the annotations are on a tree, the dot product between two proteins is the height of the lowest common ancestor of the two nodes. An alternative approach assigns annotation *a* a score of $-\log p(a)$, where $p(a)$ is the fraction of proteins that have annotation *a*. We then score the similarity of annotations *a*, *a'* as $\max_{a" \in (ancestor\ (a) \cap ancestor\ (a'))} -\log(p(a"))$. In a tree topology, this score is the similarity between the deepest common ancestor of *a* and *a'*, because the node frequencies are decreasing along a path from the root to any node. The score is a dot product with respect to the infinity norm on the annotation vector space. This also holds when the proteins have more than one annotation and the similarity between their annotations is defined as the maximum similarity between any pair of annotations. When one of the proteins has an unknown GO annotation, the kernel value is set to 0. Since GO is a DAG, the matrix which is called pseudo-kernel we get is not positive semidefinite. Therefore, to get a valid kernel, an empirical kernel map is needed to perform. That means, the rows of the square pseudo-kernel are considered as feature vectors, and then we simply compute a scalar product kernel on these vectors. The result kernel is the GO kernel we need for experiments.

4.3 Yeast biological process prediction by combining heterogeneous data

4.3.1 Creating biological process labels for yeast genes

Our problem is to predict the biological process for each yeast gene, so the first work is labeling each ORF basing on the biological process GO terms. After labeling, we have a matrix that each row is an ORF and each column is a GO term related to the corresponding biological process. The number of rows is the number of ORFs, i.e, more than six thousands, and the number of columns is the number of biological processes, i.e more than nine thousands processes. The entries of this matrix is the labels of ORFs, belong to {-1, 0, 1} for negative, unknown and positive. The entries are defined as the followings. For each GO term T, we partition the list of yeast genes into three sets. First, all genes that are annotated with T are labeled as "positive". As mentioned above, the process terms are represented as nodes in a DAG so T is a node in the DAG. Next, we traverse from T along all paths to the root of the Gene Ontology graph. At each GO term along this path, we look for genes that are assigned to that node and not to any of that node's children. We consider that such genes might be properly assigned to T, and so we label those genes as ``uncertain'' or "unknown". Note that, according to this rule, completely unannotated genes receive a label of ``unknown'' because they are assigned only to the root of the GO graph. Finally, all remaining genes are labeled as ``negative.''

4.3.2 Experiments and results

We perform the experiments on some biological processes. The datasets used for these experiments are cdc15 and cdc28 microarray data and the localization data. We do experiments on binary classification so we choose some certain biological processes among more than 10000 processes: reproduction (GO: 0000003), unknown biological process (GO: 0000004), organic acid metabolism (GO: 0006082), nucleobase, nucleoside, nucleotide, and nucleic acid metabolism (GO: 0006139). For the prediction of each process A, we create corresponding datasets including cdc28 microarray data, cdc28 with FFT (denoted as fft28), cdc15 microarray data, cdc15 with FFT (denoted as fft15), and localization data.

Training each individual dataset: First, each individual dataset is trained with a soft-margin SVM using some kernels. When the dataset is a set of vectors, it is often effective linearly scale each attribute to zero mean and unit variance, and then apply the Gaussian RBF kernel or polynomial kernel in [47]. The main advantage of normalization to avoid attributes in large numeric ranges dominating those in smaller ranges. To use a basic SVM for binary classification, two kinds of parameters have to be determined: the regularization parameter C of the SVM and the kernel and its parameters. In our experiments, the parameter setting for microarray data is performed by training SVM using 3-fold cross validation method with 3 kinds of kernels RBF, linear and polynomial. For localization data, the kernel is fixed and the parameter setting for C use 3-fold cross validation. The performance of SVM classifiers is measured by Area under Curve in [48]. After setting all parameters, microarray datasets are normalized, then cdc28 is trained with polynomial kernel degree 3 and the parameter C=500, fft28 with RBF sigma=5 and C=500. Localization data is trained with the kernel described in section 3.3 and C=500.

Combination strategy: We have tried to combine two types of data: microarray data (denoted as M) and localization data (denoted as L) using a linear combination. The formula for this combination uses a parameter λ, which represents the percentage of each type of data in the combined kernel. $K_{combination}=\lambda*K_M + (1- \lambda)*K_L$ with the value of λ in a range [0,1] where K_M is the kernel matrix of microarray data and K_L is the kernel matrix of localization data, $K_{combination}$ is the combination kernel matrix. If λ=0 the combination kernel matrix is equal to the K_L, if λ=1 the combination kernel matrix is equal to the K_M. So, with λ=0 or λ=1, the combination becomes the case of individual dataset.

Comparing the results of training each dataset to those of combination: We did the experiments with four biological processes (BPs) and the prediction performance of three BPs with data combination is better than performance of each individual data. The following tables show the performance of these predictions with the corresponding BPs.

Reproduction prediction C=500 (Pnum=121,Nnum=2994)			**Organic acid metabolism prediction C=500 (Pnum=233, Nnum=2875)**		
λ	$\lambda*K_{Mcdc28} + (1- \lambda) *K_L$	$\lambda*K_{Mfft28} + (1- \lambda)*K_L$	λ	$\lambda*K_{Mcdc28} + (1- \lambda)*K_L$	$\lambda*K_{Mfft28} + (1- \lambda)*K_L$
0	0.8882	0.8882	0	0.7608	0.8608
0.01	**0.8973**	0.8269	0.1	**0.8457**	**0.8398**
0.012	0.8042	0.8339	0.2	0.8446	0.8268
0.013	0.8167	0.8347	0.3	0.8405	0.8197
0.014	0.8166	0.8347	0.4	0.8416	0.812
0.015	0.8242	0.8344	0.5	0.8378	0.8053
0.016	0.8241	0.8342	0.6	0.8404	0.7988
0.017	0.8217	0.8342	0.7	0.8402	0.798
0.02	0.8032	0.828	0.8	0.8344	0.7958
0.9	0.7512	0.7559	0.9	0.8371	0.7925
1	0.6787	0.6668	1	0.7711	0.7491
Unknown biological process prediction C=500 (Pnum=909,Nnum=3114)			**Nucleobase, nucleoside, nucleotide, and nucleic acid metabolism prediction C=500 (Pnum=997, Nnum=2111)**		
λ	$\lambda *K_{Mcdc28} + (1- \lambda) *K_L$	$\lambda *K_{Mfft28} + (1- \lambda)*K_L$	λ	$\lambda *K_{Mcdc28} + (1- \lambda)*K_L$	$\lambda *K_{Mfft28} + (1- \lambda)*K_L$
0	0.8406	0.8406	0	0.9107	0.9107
0.01	0.8420	**0.8584**	0.1	0.8356	0.8549
0.1	0.8433	0.8327	0.2	0.8217	0.8494
0.11	**0.8442**	0.833	0.3	0.8127	0.8487
0.12	0.8441	0.8318	0.4	0.808	0.8452
0.9	0.8264	0.8055	0.5	0.8065	0.8452
1	0.7372	0.7192	0.6	0.8032	0.8426
			0.7	0.7991	0.8356
			0.8	0.7993	0.8307
			0.9	0.7957	0.8227
			1	0.6427	0.6349

The results show that three BPs (reproduction, organic acid metabolism, an unknown biological process) among four biological processes have the prediction performance of combination better than each individual dataset for both two microarray datasets cdc28 and fft28. With the same value of λ, the combination of L with Mfft28 give a better performance than the combination of L with Mcdc28 in the case of reproduction BP and nucleic acid metabolism BP. Mfft28 gives the better overall result than Mcdc28 in case of unknown BP. So it seems that using Mfft28 for prediction is better than Mcdc28 in some cases. These results show that the combination giving the better performance is promising.

V. Conclusion

For future works, we intend to solve any problem related to gene function prediction that may occur with our approach, using kernel methods like SVMs and a trend of combination heterogeneous data. We will consider all types of genomic data as well as all kinds of specific problems belong to gene function prediction such as prediction of specific molecular function.

We can focus on three directions basing on three aspects related to our approach: gene function prediction problem, kernels used for types of genomic data and ways to combine different kernels of heterogeneous data. The first one is to use available kernels for different kinds of data, apply some available ways to combine them to solve various aspects on gene function prediction. The second one is to design a new kernel for a specific type of data, then combine with other types of data to improve the performance of gene function prediction problem, which has been solved with previous approaches. The third one is to focus on a

new strategy to combine different kernels to improve the performance of existing gene function problem. In this direction, we can use available kernels for different types of data. The last direction is any combination of three directions above.

References

[1] Shah, I., & Hunter, L. 1997. Predicting Enzyme Function from Sequence: A Systematic Appraisal. Pages 276–283 of: ISMB 97.

[2] Bork, P., Dandekar, T., Diaz-Lazcoz, Y., Eisenhaber, F., Huynen, M., & Yuan, Y. 1998. Predicting Function: From Genes to Genomes and Back. Journal of Molecular Biology, 283, 707–725.

[3] Park, J., Teichmann, S.A., Hubbard, T., & Chothia, C. 1997. Intermediate sequences increase the detection of homology between sequences. J Mol Biol, 273(1)(Oct), 349–54.

[4] Karwath, A., & King, R.D. 2002. Homology Induction: The use of machine learning to improve sequence similarity searches. BMC Bioinformatics 2002, 3:11.

[5] Pawlowski, K., Jaroszewski, L., Rychlewski, L., & Godzik, A. 2000. Sensitive Sequence Comparison as Protein Function Predictor. Pages 42–53 of: Pacific Symposium on Biocomputing.

[6] Williams, H.E., & Zobel, J. 2002. Indexing and retrieval for genomic databases. IEEE Transactions on Knowledge and Data Engineering, 14(1), 63–78.

[7] Jaakkola, T., Diekhans, M., & Haussler, D. 2000. A discriminative framework for detecting remote protein homologies. Journal of Computational Biology, 7(1,2), 95–114.

[8] Hudak, J., & McClure, M.A. 1999. A Comparative Analysis of Computational Motif-Detection Methods. Pages 138–149 of: Pacific Symposium on Biocomputing.

[9] Higgins, D. G., J.D. Thompson, J. D., & Gibson, T. J. 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. Nucleic Acids Research, 22, 4673–4680.

[10] Attwood, T.K., Blythe, M., Flower, D.R., Gaulton, A., Mabey, J.E., Maudling, N., McGregor, L., Mitchell, A., Moulton, G., Paine, K., & Scordis, P. 2002. PRINTS and PRINTS-S shed light on protein ancestry. Nucleic Acids Research, 30(1), 239–241. 2002

[11] Koonin, E., Tatusov, R., Galperin, M., & Rozanov, M. 1998. Genome analysis using clusters of orthologous groups (COGS). Pages 135–139 of: RECOMB 98.

[12] Tatusov, R.L., Natale et al., 2001. The COG database: new developments in phylogenetic classification of proteins from complete genomes. Nucleic Acids Res, 29(1)(Jan), 22–8.

[13] DeRisi, J., Iyer, V., & Brown, P. 1997. Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale. Science, 278(October), 680–686.

[14] Eisen, M., Spellman, P., Brown, P., & Botstein, D. 1998. Cluster analysis and display of genome-wide expression patterns. Proc. Nat. Acad. Sci. USA, 95(Dec), 14863–14868.

[15] Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., & Levine, A. 1999. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc. Nat. Acad. Sci. USA, 96(12) (Jun), 6745–50.et al., 1999;

[16] Heyer, L. J., Kruglyak, S., & Yooseph, S. 1999. Exploring expression data: identification and analysis of coexpressed genes. Genome Research, 9(11)(Nov), 1106–15.

[17] Toronen, P., Kolehmainen, M., Wong, G., & Castr´en, E. 1999. Analysis of gene expression data using self-organizing maps. FEBS Lett., 451(2)(May), 142–6.

[18] Tavazoie, S., Hughes, J., Campbell, M., Cho, R., & Church, G. 1999. Systematic determination of genetic network architecture. Nature Genetics, 22(July), 281–285.

[19] Butte, A., & Kohane, I. 2000. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements, PSB 2000.

[20] Hvidsten, T.R., Komorowski, J., Sandvik, A.K., & Lægreid, A. 2001. Predicting Gene Function from Gene Expressions and Ontologies. Pages 299–310 of: Pacific Symposium on Biocomputing.

[21] Komorowski, J., & Øhrn, A. 1999. Diagnosing Acute Appendicitis with Very Simple Classification Rules. Page 462467 of: Proc. Third European Symposium on Principles and Practice of Knowledge Discovery in Databases.

[22] CASP. 2001. Fourth Meeting on the Critical Assessment of Techniques for Protein Structure Prediction. Supplement in Proteins: Structure, Function and Genetics 45 (S5).

[23] Wilson, C., Kreychman, J., & Gerstein, M. 2000. Assessing annotation transfer for genomics: Quantifying the relations between protein sequence, structure and function through traditional and probablaistic scores. JMB, 297, 233–249.

[24] Ouali, M., & King, R.D. 2000. Cascaded multiple classifiers for secondary structure prediction. Protein Science, 9(6)(Jun), 1162–76.

[25] Marcotte, E., Pellegrini, M., Thompson, M., Yeates, T., & Eisenberg, D. 1999. A combined algorithm for genome-wide prediction of protein function. Nature, 402(Nov), 83–86.Mewes et al., 1999

[26] Hanisch, D., Zien, A., Zimmer, R., & Lengauer, T. 2002. Co-clustering of biological networks and gene expression data. Bioinformatics, 18, S145–S154.

[27] Kretschmann, E., Fleischmann, W., & Apweiler, R. 2001. Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on SWISS-PROT. Bioinformatics, 17, 920–926.

[28] Des Jardins, M., Karp, P., Krummenacker, M., Lee, T., & Ouzounis, C. 1997. Prediction of Enzyme Classification from Protein Sequence without the use of Sequence Similarity. In: ISMB '97.

[29] U. Karaoz et al., 2004. Whole-genome annotation by using evidence integration in functional-linkage networks. Proc. Natl. Acad. Sci. USA, 101(9):2888-93.

[30] Clare, A., & King, R. D. 2003. Machine learning of functional class from phenotype data. Bioinformatics, 18(1), 160–166.

[31] G. R.G. Lanckriet, M. Deng, N. Cristianini, et al. A statistical framework for genomic data fusion, Bioinformatics 20:2626-35, 2004a

[32] Gene Ontology Consortium, 2000. Nat Genet. 2000 May;25(1):25-9

[33] Riley, M. 1998. Systems for categorizing functions of gene products. Current Opinion in Structural Biology, 8, 388–392.

[34] Rubin, D. L., Shafa, F., Oliver, D. E., Hewett, M., , & Altman, R. B. 2002. Representing genetic sequence data for pharmacogenomics: an evolutionary approach using ontological and relational models. Bioinformatics, 18, S207– S215.

[35] Gusfield, D., Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, New York, 1997.

[36] D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CLR-99-10, Department of Computer Science, UCSC, 1999
[37] M.P.S Brown, W.N. Grundy, D. Lin, N.Cristianini, C.Sugnet, T.S.Furey, Jr.M.Ares and D.Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. Proceedings of the National Academy of Sciences of the USA, 97(1):262-267, 2000
[38] P. Pavlidis et al., Promoter region-based classification of genes. Proceedings of the Pacific Symposium on Biocomputing, pages 151-163, River Edge, NJ, World Scientific, 2001a.
[39] J.-P. Vert, A tree kernel to analyze phylogenetic profiles. Bioinformatics, 18:S276-S284, 2002b.
[40] P. Pavlidis, J. Weston, J. Cai and W. S. Noble. Learning gene functional classifications from multiple data types. Journal of Computational Biology, 9 (2):401-411, 2002.
[41] Troyanskaya et al., A Bayesian framework for combining heterogeneous data sources for gene function prediction, Proc. Natl. Acd. Sci. USA, 100(14):8348-8353, 2003.
[42] Y. Chen and D. Yu. Global protein function annotation through mining genome scale data in yeast Saccharomyces cerevisiae, Nucleic Acids Res., Dec 2004; 32(21):6414-24.
[43] 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. In Proceedings of the Pacific Symposium on Biocomputing (PSB), Big Island of Hawaii, Hawaii, Jan 2004b
[44] J.-P. Vert and M. Kanehisa. Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. Advances in Neural Information Processing Systems, volume 15, pages 1425-1432. Cambridge, MA, MIT Press, 2003b.
[45] Spellman, P, Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., & Futcher, B. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Molecular Biology of the Cell, 9(December), 3273–3297.
[46] Ben-Hur A, Noble WS: Kernel methods for predicting protein-protein interactions. Bioinformatics 2005, 21 suppl 1:i38–i46.
[47] Schoelkopf, B., Tsuda, K. and Vert, J.-P. (eds). Kernel methods in Computational Biology. MIT press, Cambridge, MA, 2004.
[48] Hanley & McNeil, 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, Vol 143, 29-36.
[49] Cristianini, N., & Shawe-Taylor, J. 2000. An Introduction to Support Vector Machines. Cambridge University Press, Cambridge, UK.