BIOINFORMATICS ORIGINAL PAPER Vol. 22 no. 7 2006, pages 830–836
doi:10.1093/bioinformatics/btk048

Systems biology

Hierarchical multi-label prediction of gene function

Zafer Barutcuoglu[1], Robert E. Schapire[1] and Olga G. Troyanskaya[1,2,*]

[1]Department of Computer Science, Princeton University, 35 Olden Street, Princeton, NJ 08544, USA and
[2]Lewis-Sigler Institute for Integrative Genomics, Princeton University, NJ, USA

Received on October 24, 2005; revised on December 21, 2005; accepted on January 6, 2006
Advance Access publication January 12, 2006
Associate Editor: Alvis Brazma

ABSTRACT

Motivation: Assigning functions for unknown genes based on diverse large-scale data is a key task in functional genomics. Previous work on gene function prediction has addressed this problem using independent classifiers for each function. However, such an approach ignores the structure of functional class taxonomies, such as the Gene Ontology (GO). Over a hierarchy of functional classes, a group of independent classifiers where each one predicts gene membership to a particular class can produce a hierarchically inconsistent set of predictions, where for a given gene a specific class may be predicted positive while its inclusive parent class is predicted negative. Taking the hierarchical structure into account resolves such inconsistencies and provides an opportunity for leveraging all classifiers in the hierarchy to achieve higher specificity of predictions.

Results: We developed a Bayesian framework for combining multiple classifiers based on the functional taxonomy constraints. Using a hierarchy of support vector machine (SVM) classifiers trained on multiple data types, we combined predictions in our Bayesian framework to obtain the most probable consistent set of predictions. Experiments show that over a 105-node subhierarchy of the GO, our Bayesian framework improves predictions for 93 nodes. As an additional benefit, our method also provides implicit calibration of SVM margin outputs to probabilities. Using this method, we make function predictions for multiple proteins, and experimentally confirm predictions for proteins involved in mitosis.

Supplementary information: Results for the 105 selected GO classes and predictions for 1059 unknown genes are available at: http://function.princeton.edu/genesite/

Contact: ogt@cs.princeton.edu

1 INTRODUCTION

Discovering biological functions of an organism is a central goal of functional genomics. Assigning functions for every protein with traditional experimental techniques could take decades, but the currently accumulated data from different biological sources make it possible to generate automated predictions that guide laboratory experiments and speed up the annotation process. 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 (e.g. Clare and King, 2003; Karaoz *et al*., 2004; Lanckriet *et al*., 2004a). Recently, integration of different types of biological data in a single model has shown promising improvements, using such learning methods as support vector machines (SVMs) and Bayes nets (e.g. Pavlidis *et al*., 2002; Troyanskaya *et al*., 2003; Chen and Xu, 2004; Lanckriet *et al*., 2004b).

Collections of functional class definitions, such as the Gene Ontology (GO) (The Gene Ontology Consortium, 2000) and CYGD of MIPS (Güldener *et al*., 2005), are organized hierarchically, where general functions include other more specific functions. However, existing prediction approaches typically formulate the problem on a per-function basis, ignoring this structure. A separate independent classifier is constructed to predict membership for each functional class. This turns the problem into a convenient form for common machine learning algorithms, but abstracts away any information in the functional hierarchy, essentially allowing the classifiers as a whole to violate the principle that a gene cannot belong to a child class without also belonging to its parents in the hierarchy. This structure has not been exploited in any previous work to improve classification and enforce consistency. Our approach aims to preserve the hierarchical information to increase predictive accuracy over all classes.

2 SYSTEM AND METHODS

We created a system for exploiting the hierarchical structure of a functional class taxonomy to improve the accuracy of predictions of individual classifiers.

2.1 Algorithm

This type of general setting where an example can belong to any number of classes in a hierarchy is called multi-label hierarchical classification in the machine learning field, though previous work is rather sparse, and available algorithms such as Cesa-Bianchi *et al*. (2004) focus on preventing inconsistent predictions by not consulting child classifiers at all if a parent predicts a gene as negative. Such a scheme does ensure consistency by design, but higher-level classifiers get little or no leverage despite the high impact of their errors on lower-level nodes. Our Bayesian method addresses this issue by allowing all levels of classifiers to be influenced by one another. This is especially important in the biological setting, where most classes of interest lie at the leaf nodes in the hierarchy.

Our approach initially allows inconsistencies to occur, and then exploits them. We start with independently trained classifiers for each class as before and design a Bayesian hierarchical combination scheme to allow collaborative error-correction over all nodes. The possibly inconsistent set of predictions obtained from the independent classifiers is used to find the most probable set of consistent labels. Results from our computational experiments demonstrate the accuracy improvements of hierarchical combination.

We will use y_i to denote the binary label of membership to function class i, and $\hat{y}_i$ to denote the classifier output for that class. Classifier outputs can be

*To whom Correspondence should be addressed.

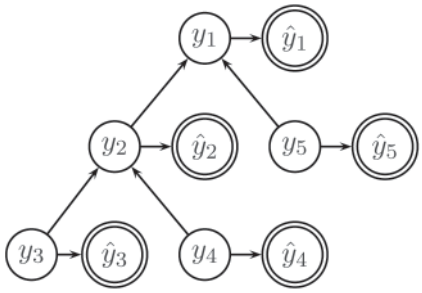


Fig. 1. Bayesian hierarchical combining. The y nodes are the binary-valued hidden nodes representing actual membership to the class, and the corresponding $\hat{y}$ nodes are the observed classifier outputs.

as simple as binary predictions, or real-valued confidence measures including (but not limited to) probabilities. Thresholding of these outputs are handled automatically.

After presenting an unknown example to all classifiers, we obtain a set of (possibly inconsistent) $\hat{y}$ values, and we wish to find the most probable set of (consistent) y values that may have led to them, i.e. for N nodes we need to find the $y_1, \ldots, y_N$ that maximizes

$$P(y_1, \ldots, y_N \,|\, \hat{y}_1, \ldots, \hat{y}_N) = \frac{P(\hat{y}_1, \ldots, \hat{y}_N \,|\, y_1, \ldots, y_N) P(y_1, \ldots, y_N)}{Z}, \quad (1)$$

where Z is a constant normalization factor.

We propose a Bayesian network structure for this problem, of which Figure 1 shows an example. The y nodes are conditioned on their child classes[1], and the $\hat{y}$ nodes are conditioned on the label y of the class. The edges among the y nodes serve to impose the hierarchical consistency among the labels. By setting the appropriate entry in its conditional probability table in the Bayes net, we can trivially ensure that a label is 1 when any one of its children is 1. The edges from y to $\hat{y}$, on the other hand, reflect a simplifying observation: for a given example, a classifier output $\hat{y}_i$ is a random variable independent of all other classifier outputs $\hat{y}_j$ and labels y_j $(i \neq j)$ given its true label y_i. This allows us to simplify Equation (1):

$$P(\hat{y}_1, \ldots, \hat{y}_N, \,|\, y_1, \ldots, y_N) = \prod_{i=1}^{N} P(\hat{y}_i \,|\, y_i), \quad (2)$$

and from the y edges,

$$P(y_1, \ldots, y_N) = \prod_{i=1}^{N} P(y_i \,|\, ch(y_i)), \quad (3)$$

where $P(y_i \,|\, ch(y_i))$, the label probabilities conditioned on child labels, can be inferred from the training labels by counting.

$P(\hat{y}_i \,|\, y_i)$ can also be estimated during training by validation, modeling the distributions of $\hat{y}_i$ outputs over positive and negative validation examples. For real-valued classifier outputs, one way to do this is to assume a parametric model (e.g. Gaussian) for each of the distributions $P(\hat{y}_i \,|\, y_i = 1)$ and $P(\hat{y}_i \,|\, y_i = 0)$, and estimate model parameters (e.g. mean and variance) on held-out examples. For binary classifier outputs, these distributions would be binomial, trivially estimated by the confusion matrices from validation.

Now we can use any standard exact inference or simulation algorithm to find the most likely configuration of consistent hidden y labels for the given $\hat{y}$ outputs. Or, if we wish to retain individual membership probabilities instead of one most likely discrete assignment, we can calculate $P(y_i \,|\, \hat{y}_1, \ldots, \hat{y}_N)$ for every node separately.

[1]Reversing these arcs and conditioning y nodes on their parents also results in a Bayes net, but conditioning on children makes more causal sense to us, and in our experiments we found the latter to produce slightly better results.

2.2 Training individual classifiers

Our training process starts by training the individual classifiers. In this phase, independent classifiers are conventionally trained for each class, with no regard to the hierarchy. Each classifier is responsible for predicting membership to a particular class by means of a binary or real-valued output. For the classifiers in our experiments, we used SVMs without thresholding their outputs, as will be described in more detail.

Since validation results will be needed for the Bayes net, some training examples have to be held out. However, for most classes in our data positive examples are too rare to hold out completely, so we use bootstrapping (random sampling with replacement) to create 10 bootstrap samples from the full training data for a class, each of which excludes a ratio of ∼0.368 of all examples, and may contain multiple copies of the rest (Efron and Tibshirani, 1993). Each bootstrap sample is used to train a separate classifier that is evaluated on the examples that were excluded from its training sample. This yields 10 classifiers for a class. For a previously unseen example, each classifier of the class will be evaluated, and the median of their outputs will be taken as the combined output. This corresponds to taking a majority vote with a given threshold and is known as bagging (bootstrap aggregation) (Breiman, 1996). For evaluating training examples for validation, only those classifiers which were not trained on that example will be included in the aggregation.

A widely used alternative to bootstrapping is k-fold cross-validation. In this setting, we prefer bootstrapping because to achieve a comparable held-out example ratio to bootstrapping, cross-validation would be limited to $k = 3$, notably reducing the benefit of aggregation.

We use aggregation for the final classifier instead of training a new one on all training data, because in the latter case the final classifier could have an unexpectedly high error that would be impossible to detect. On the other hand, an aggregate classifier is expected to improve the accuracy of individual classifiers, which have known accuracies from validation.

2.3 Constructing the Bayes net

The next step is to construct the Bayes net to combine outputs. First, we assume the aggregate classifier outputs to have Gaussian distributions for positive and negative examples. Using the validation-phase predictions from held-out examples (or rather, held-out bootstrap-classifiers for each example), we estimate means and variances for each class. Figure 2 contains a typical plot of outputs for a GO node, showing that our assumption of Gaussian distributions is not unreasonable. These estimated distributions define the conditional probabilities $P(\hat{y}_i \,|\, y_i = 0)$ and $P(\hat{y}_i \,|\, y_i = 1)$ in our Bayes net.

3 IMPLEMENTATION

3.1 Support vector machines

The SVM classifier is a state-of-the-art machine learning method that separates positive and negative examples with a linear decision boundary (i.e. a hyperplane) in a feature space and aims to achieve better generalization through a principle termed maximizing the margin (Burges, 1998).

For our individual classifiers, we trained the SVMs using the SVM^{light} software (Joachims, 1999).

We experimented with linear SVMs as well as RBF-kernel SVMs using various γ, over a wide range of capacity (C) values. The functional genomics data we used turned out to be linearly separable in the input space, and cross-validated results showed that generally hard-margin ($C = \infty$) linear SVMs performed slightly better than soft-margin linear or RBF-kernel SVMs, the latter severely overfitting at times. This is not very surprising, considering the large number of input features compared with the relatively small number of training examples. To consider the versatility of our hierarchical

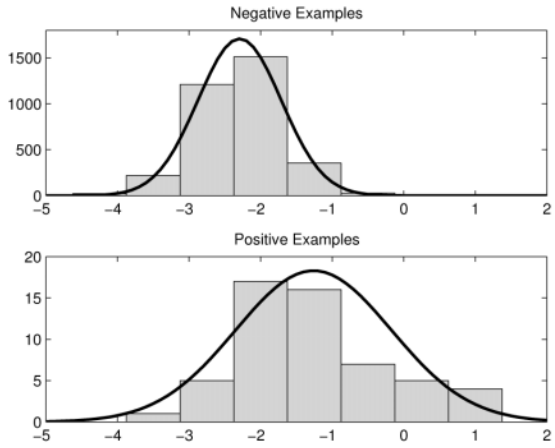


Fig. 2. Modeling classifier outputs (`GO:0007059 'chromosome segregation'`). The histograms indicate the distribution of unthresholded median SVM outputs for positive and negative validation examples. (The *y*-axes have different scales.)

learning method, we also trained Logistic Regression classifiers which lack the maximum-margin properties of SVMs. Because Logistic Regression classifiers show worse generalization than SVMs as expected, we focus on hard-margin linear SVMs as the base classifiers in this study.

For our purposes of Bayesian combination, we use unthresholded SVM outputs, i.e. all further mentions of SVM predictions are unbounded real values. The calibration of these outputs to actual probabilities is performed implicitly by the output distribution modeling procedure explained below.

3.2 Data sources and processing

3.2.1 Training labels Our training labels come from the `biological_process` hierarchy of the GO. A positive example for a class denotes a gene that is annotated to that node or one of its descendant nodes in GO. This upward propagation of annotations is natural because of the hierarchical nature of GO, and it also has the consequence that all training label configurations are hierarchically consistent by definition.

Of the 1000 nodes in the GO which have direct annotations for yeast, in consultation with a yeast geneticist, we chose 105 nodes that have both a reasonable number of annotations (at least 1 direct and 15 total annotations) and were specific enough to have some biological significance if a new gene were predicted to belong there.

Unfortunately, GO annotations are almost exclusively positive[2]. This renders the data unsuitable for typical classifiers as is, as there are few negative examples to separate from the positives. Similarly to previous work, we use an annotated gene as a negative example for the nodes it is not annotated to. There is some justification to this *ad hoc* introduction of negatives—if a gene is already known to be annotated to a node, it is less likely to be annotated to additional nodes. As a slight improvement to this, we also do not include a node's positive annotations as negatives in any of its descendants, reasoning with the same biological intent of making the assumed set of negatives more specific. Our experiments show that this additional filtering of introduced negatives provides a small improvement in accuracy, but more importantly it yields more unused examples to be evaluated later at the child nodes for new discoveries.

[2]The GO does include occasional negative annotations, but these are the very few cases found to be particularly surprising for biologists.

3.2.2 Interaction data Pairwise interaction datasets denote the existence of an interaction between pairs of proteins (gene products) under certain experimental conditions. The biological premise is that if two proteins have a certain interaction, their genes might be more likely to belong to the same functional class.

In our experiments, we use the GRID collection of pairwise interaction datasets (Breitkreutz *et al*., 2003). Our snapshot contains eight types of interactions (Affinity Precipitation, Two Hybrid, Synthetic Lethality, Affinity Chromatography, Synthetic Rescue, Dosage Lethality, Purified Complex and Biochemical Assay). Each of these datasets can be viewed as a square binary matrix whose non-zero elements denote an interaction between the row gene and the column gene. To transform this second-order data type into the form expected by typical machine learning algorithms, we treat each column of a matrix as a feature, treating rows as the examples' feature vectors. For the nine matrices (and their transposes since they are not necessarily symmetric), we concatenate the feature vectors obtained as such, and obtain 88 200 features for 4900 genes.

3.2.3 Microarrays Microarray datasets are real-valued matrices measuring gene expression levels under different experimental conditions. We use gene expression microarray data from the Stanford Microarray Database (SMD) containing results from several publications (Chu *et al*., 1998; Spellman *et al*., 1998; Ogawa *et al*., 2000; Sudarsanam *et al*., 2000; Gasch *et al*., 2000, 2001; Yoshimoto *et al*., 2002; Shakoury-Elizeh *et al*., 2004), providing a total of 342 real-valued features for 5737 common genes. A total of 4524 of these genes are among the 4900 genes in GRID, and we fix this set of 4524 genes to be used in our experiments.

Microarray entries typically include missing values due to experimental imperfections. We estimate such entries using the widely accepted `KNNimpute` algorithm (Troyanskaya *et al*., 2001) with $k = 15$.

3.2.4 Colocalization data Colocalization datasets provide information about where the gene products are found in the cell. If two proteins are found in the same locales in the cell, their genes may be more likely to belong to the same class.

We combine two colocalization datasets. One is the curated localization data from the `molecular_complex` hierarchy of the GO, of which we use 148 terms, providing data for a total of 1043 genes. The other is an adaptation of the O'Shea data (Huh *et al*., 2003) used in a previous function study, and contains 2902 genes (Jansen *et al*., 2003).

We treat each locale of each dataset as a binary feature where 1 denotes membership. For genes not included in these datasets, we use zero values.

3.2.5 Transcription factor binding sites Transcription factor binding sites data are another grouping of genes based on a common attribute. If two genes are known to have a common transcription factor binding site, they are more likely to belong to the same biological process, since they are being transcribed together.

We use the PROSPECT dataset (Fujibuchi *et al*., 2001), which has 27 experimentally identified sites for 146 genes. As before, we

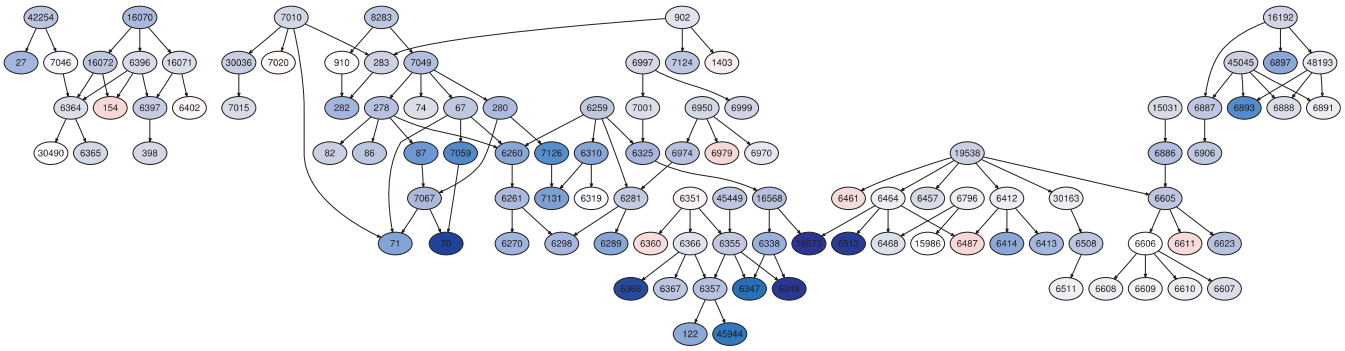


Fig. 3. The GO subhierarchy used in this study colored by AUC changes. Darker shades of blue indicate largest improvement and darker shades of red indicate largest deterioration. White nodes indicate no change in AUC. Nodes are labeled with abbreviated GO IDs; e.g. '7087' stands for `GO:0007087`. Most nodes improve in AUC, with the largest improvement observed at leaf nodes. Among the few deteriorations, most are also at leaf nodes.

treat the presence or absence of each site as a binary feature and use zeros for genes not included.

3.2.6 Processing of input data We integrate all data into a single feature set by concatenating all feature vectors for a gene. Namely, the input features for an example consist of that gene's interaction flags with all genes for all interaction experiments, flags for membership to each colocalization locale and possession of each transcription factor binding site, and the log-ratio values for each slide in each microarray experiment. Real-valued features (currently only microarray data) are Z-score normalized to zero-mean and unit-variance.

Such direct combination of all data into a single dataset was termed early integration in Pavlidis *et al*. (2002), and used with feature scaling, was found to be the best-performing data integration policy for most of their selected classes. Although we do not apply explicit feature scaling in our experiments, our linear SVMs are essentially feature scaling operations themselves, since an inner product with the separating hyperplane normal vector is a particular linear weighting of features.

Most interaction datasets being rather sparse, most of the columns in the resulting dataset of 88 721 dimensions contain less than two non-zero entries. Removing all such uninformative columns leaves 5930 features for the 3465 annotated genes.

Upon training of our linear SVMs, we can assess how much each data type contributes to classification decisions based on the linear coefficients for decision boundaries. We examined the ratio of each data type's absolute-valued weights to the total (including bias). Although the ratios vary widely for different nodes, on average the interaction data contribute 60%, microarrays 36%, colocalization 4% and transcription factor binding sites <1% of the weights. The number of interaction features is disproportionately larger (88 200) than microarrays (342), but the interactions data are very sparse and binary while microarrays are dense and real-valued. Thus, the substantial contribution of microarrays is not surprising, and for several classes (27) microarray data contribute the most to the weights.

4 RESULTS

In this work, we develop a Bayesian framework which incorporates the class hierarchy to improve prediction accuracy of independent classifiers, also solving the problem of hierarchically inconsistent predictions. Using SVMs trained independently on heterogeneous *Saccharomyces cerevisiae* data, we evaluate our method on held-out annotated genes and make predictions for unknown genes.

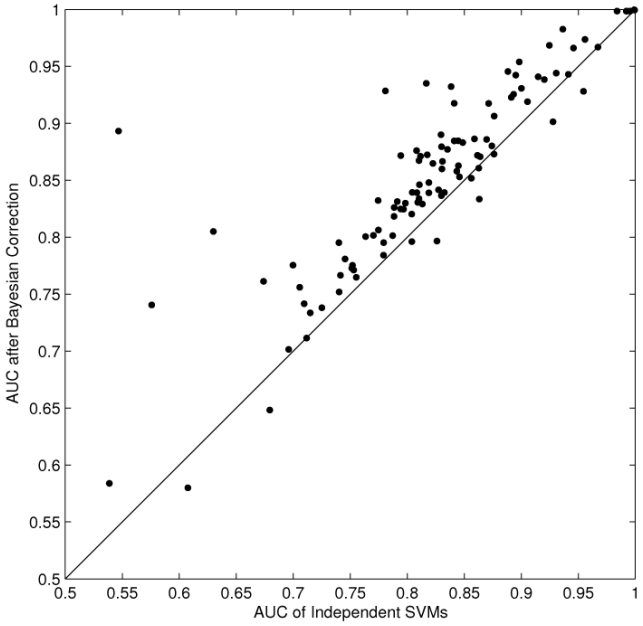
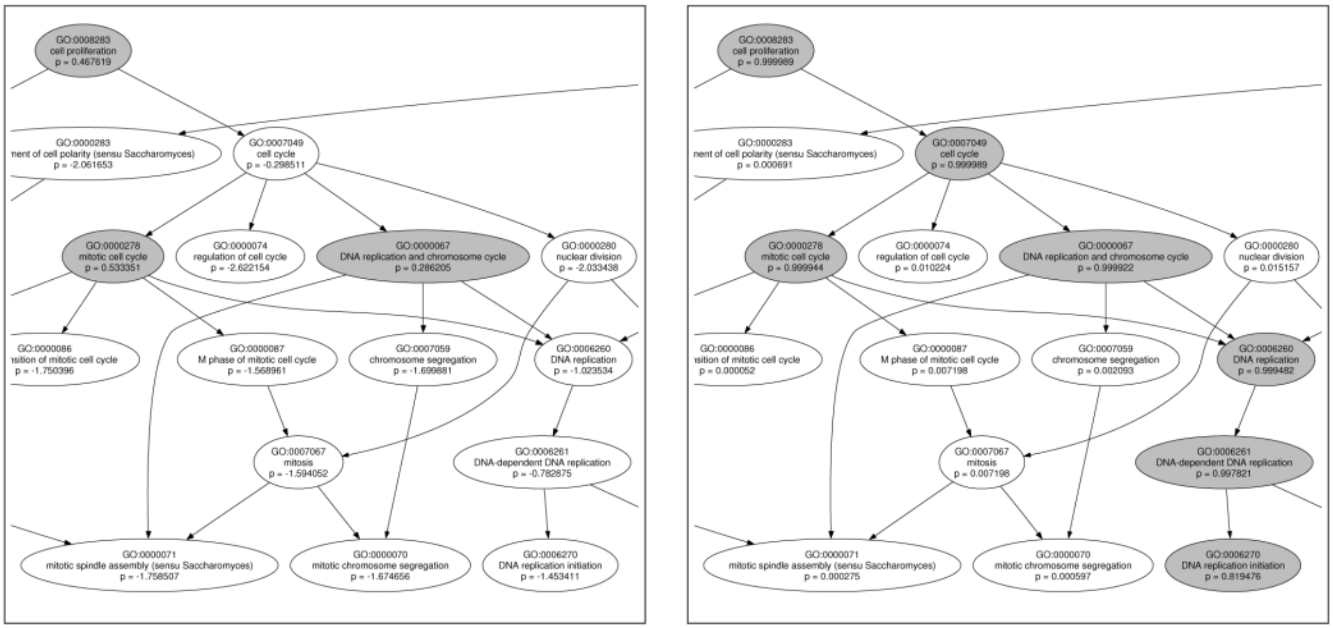


Fig. 4. Scatter-plot of AUCs after versus before Bayesian combination. Points above the diagonal correspond to accuracy improvements by our method.

4.1 Prediction accuracy on held-out data

We use the *AUC score* (area under the ROC curve) as a measure of the ranking accuracy of the classifiers' outputs. This measure is invariant to changes in the actual calibration of outputs as long as their ordering stays the same, so it eliminates the issue of unbounded SVM outputs and Bayes net probabilities having different range and calibration. For each node, we compare the AUC scores of the aggregate SVM output and the marginal probability from the Bayes net.

Our approach of hierarchical Bayesian correction increased AUC for 93 of the 105 nodes. Our GO subhierarchy is shown in Figure 3,



(a) Independent SVMs **(b)** Bayesian correction

Fig. 5. **(a)** SVM outputs (gray if above 0) and **(b)** marginal probabilities from Bayes net (gray if above 0.5) for held-out gene YNL261W. SVM outputs have an inconsistency at the GO:0007049 'cell cycle' node which is corrected by the Bayes net. Also, the Bayes net can change predictions for seemingly consistent nodes as well, including leaf nodes. For this gene, all Bayesian predictions thresholded at $P = 0.5$ are correct. YNL261W is a subunit of the origin recognition complex that binds to replication origin and directs DNA replication (Bell, 2002).

colored by changes in AUC score where darker shades of blue indicate larger increases, and darker shades of red indicate larger decreases. The largest improvements were observed at deeper nodes. The Bayesian combination has substantially increased accuracy for the vast majority of functional categories, as evident from the comparison of AUC scores before and after hierarchical improvement (Fig. 4).

Figure 5 compares a set of raw SVM (aggregate) predictions to the corresponding Bayesian marginal predictions for a particular held-out gene. Using the default zero threshold for SVMs and 0.5 for the Bayes net probabilities, the Bayes net corrects the inconsistency in the SVM predictions, and also correctly changes predictions for lower-level nodes as well.

The average change in AUC over all nodes is +0.033 (a ratio of 4% improvement over the old AUC), with a minimum of −0.031 and maximum of +0.346 (63% improvement over the old AUC). Note that this comparison of absolute changes in AUC does not take into account that for fairly accurate classifiers the room for improvement is small. One way to correct this is to observe the proportion of decrease in the area over the ROC curve (1−AUC), which is literally the room for improvement. On average, the new area over the ROC curve is 21% less than the old one, reflecting substantial improvements for the previously weakly performing nodes.

The AUC deterioration in some nodes after Bayesian combination might be due to poor output modeling by Gaussians, or correlated errors (mutual misleading) among nodes. In any case, such nodes are identified in these cross-validation results before actual use, so for those nodes SVM predictions can simply be used unmodified. In that case, we get an average AUC change of +0.035.

4.2 Predictions for new annotations

To maintain consistency of our experiments over time, we used a snapshot of the GO annotations taken at the beginning of this work in April 2004. Since then, as of July 2005, 88 previously unannotated yeast genes for which we have input data have been annotated in our selected hierarchy of 105 nodes (or their descendants), yielding 346 new gene-to-node pairs. For these 88 genes, we examined predictions using our old snapshot data before and after Bayesian correction.

Independent SVM classifiers for these genes achieve 32% precision [TP/(TP + FP)] and 7% sensitivity [TP/(TP+FN)]. Using Bayesian correction improves the sensitivity three times (21%) at comparable precision (31%). Furthermore, at the same sensitivity (7%), the Bayes net produces 51% precision. As the Bayes net gives a confidence-based output, by thresholding at the desired level our method can be used to leverage sensitivity or precision or both.

Although some SVM predictions are incorrectly changed from positive to negative due to noisy data, many more predictions are correctly modified, and even with a high-precision threshold of 0.99 our Bayesian scheme yields more true positives than independent SVMs. Incidentally, independent SVMs cannot find any of the direct annotations (which are of particular interest for being the most specific) for the 88 genes, while the Bayes-corrected system is able to correctly predict some, even for the higher thresholds.

4.3 Predictions of novel proteins involved in mitosis

Our system makes predictions of function for multiple unannotated genes. Such predictions can be used to guide experimental verification of these functions. To assess the system's ability to make

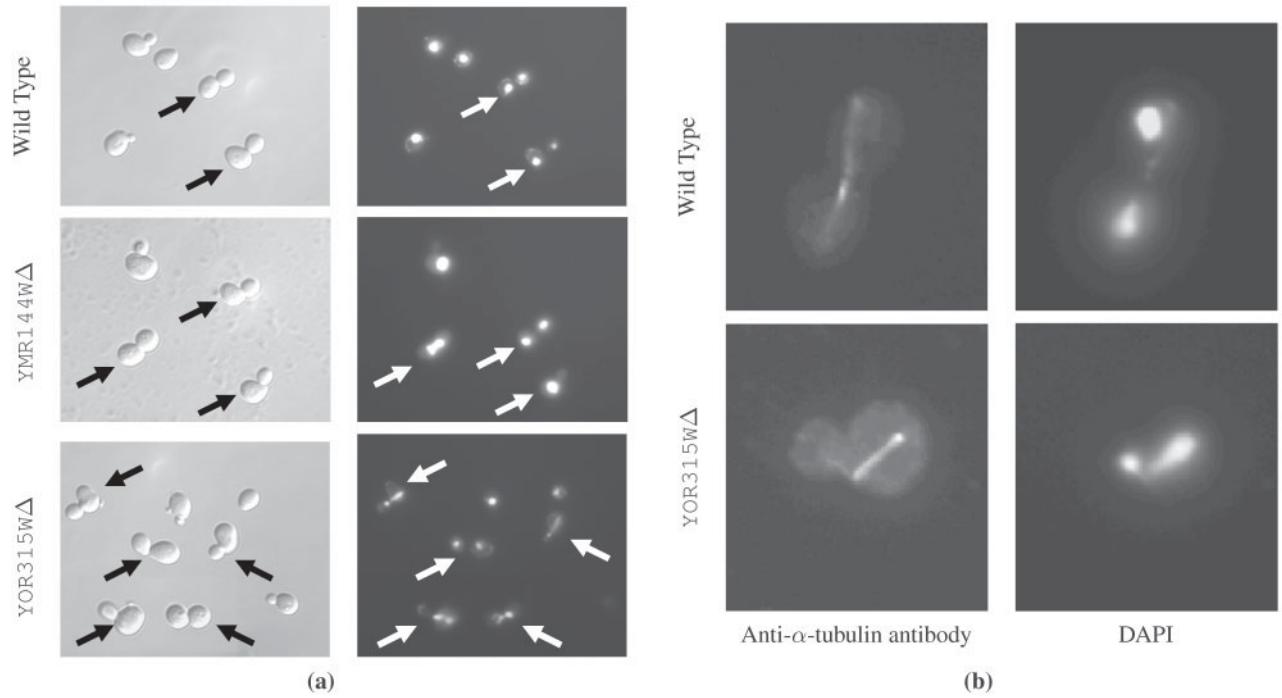


Fig. 6. Experimental validation of predictions. **(a)** Yeast cells were stained, and photographed using differential inference contrast imaging or DAPI staining. Populations of cells lacking either YMR144W or YOR315W have a significantly higher number of large-budded cells with nuclear defects. **(b)** Cells were fixed and their spindles were visualized with an anti-α-tubulin antibody. Large-budded cells lacking YOR315W exhibit frequent misaligned spindles and nuclear defects.

biologically relevant predictions that can be readily tested experimentally, we have examined in detail a small group of unknown genes with function predictions related to mitosis. All of these predictions were introduced by our hierarchical system, i.e. they were not predicted as positive by the individual classifiers prior to hierarchical combination.

YMR144W is an unknown protein that our system predicted to be involved in mitotic chromosome segregation. Indeed, when we examine a *S.cerevisiae* strain lacking this protein, the cells show significant increase in chromosome segregation defects as compared wild-type cells (F-score $= 6.6 \times 10^{-12}$), with multiple large budded cells with nuclei in the bud neck. This phenotype is consistent with that of ctf4Δ mutant—a strain lacking a known chromosome segregation gene (Miles and Formosa, 1992).

For the YOR315W gene, we make a novel prediction of mitotic spindle assembly. Yeast cells lacking this gene cannot properly separate their DNA during cell division—DNA is localized in elongated clumps along the spindle, mostly in the mother cells. These nuclear defects in large budded cells are significant with F-score of 2.8×10^{-12}. Furthermore, using anti-α-tubulin antibody, we demonstrate that YOR315WΔ cells have misaligned spindles during cell division, supporting our specific predictions of YOR315W function (Fig. 6).

YMR299C was predicted by our system to be involved in mitotic cell cycle. A recent study by Lee *et al*. (2005) has shown that this gene encodes a protein that is part of a dynein pathway that plays a critical role in mitosis, supporting our prediction.

While these verifications do not prove that all novel predictions made by our system are accurate, they demonstrate that these predictions can be readily tested by directed experiments. Until experimental validation, these are predictions, not true functional assignments. However, availability of such predictions can greatly accelerate the assignment of such annotations by driving experimentation.

DISCUSSION

Our proposed method of correcting inconsistent predictions in a multi-label class hierarchy was shown to improve performance for the majority of functions. In addition to eliminating inconsistencies, our Bayesian scheme also implicitly transforms unbounded real-valued classifier outputs into marginal probabilities and provides better calibration. Although our base classifiers of choice were SVMs, our system is a generic hierarchical ensemble method for leveraging accuracy that can work with any type of classifier without modification. Thus, given existing trained classifiers (e.g. from previous research on GO) our method can be directly applied to improve performance, needing only cross-validation results for output distribution modeling which are most often already available.

Prediction over the GO is characterized by the virtual lack of negative examples. One obvious path to explore as future work is applying density estimation algorithms, such as *Maximum Entropy*, which work with only positive examples to estimate a probability density function over the input space. Unfortunately, the lack of a real gold standard that includes negatives will still prevent comparing such an algorithm to our current approach on an equal basis.

Our approach in this article is based on combining pre-trained classifiers. Constructing a system that trains the individual base classifiers in cooperation with each other and the combination

mechanism might be able to make better use of the data, providing another prospect for further improvement.

ACKNOWLEDGEMENTS

The authors would like to thank Camelia Chiriac and Rajiv Ayyangar for laboratory work, Kara Dolinski and members of the Functional Genomics Laboratory for their contributions, and the anonymous referees for improvements in this manuscript. O.G.T. is an Alfred P. Sloan Fellow. This research was supported by NSF grant IIS-0513552 and partially supported by NIH grant R01 GM071966 to O.G.T.

Conflict of Interest. None declared.

REFERENCES

Bell,S.P. (2002) The origin recognition complex: from simple origins to complex functions. *Genes Dev*., **16**, 659–672.

Breiman,L. (1996) Bagging predictors. *Mach. Learn*., **24**, 123–140.

Breitkreutz,B.-J. *et al*. (2003) The GRID: the General Repository for Interaction Datasets. *Genome Biol*., **4**, R23.

Burges,C.J.C. (1998) A tutorial on support vector machines for pattern recognition. *Data Mining Know. Discov*., **2**, 955–974.

Cesa-Bianchi,N., Conconi,A. and Gentile,C. (2004) Regret bounds for hierarchical classification with linear-threshold functions. *The Seventeenth Annual Conference on Learning Theory LNAI 3120,* pp. 93–108. Springer, Berlin.

Chen,Y. and Yu,D. (2004) Global protein function annotation through mining genome-scale data in yeast *Saccharomyces cerevisiae*. *Nucleic Acids Res*., **32**, 6414–6424.

Chu,S. [Erratum (1998) The transcriptional program of sporulation in budding Erratum yeast. *Science*, **282**, 1421.]. 699–705.

Clare,A. and King,R.D. (2003) Predicting gene function in *Saccharomyces cerevisiase*. *Bioinformatics*, **19** (Supp. 2), II42–II49.

Efron,B. and Tibshirani,H. (1993) An Introduction to the Bootstrap. Chapman and Hall, New York.

Fujibuchi,W. *et al*. (2001) PROSPECT improves cis-acting regulatory element prediction by integrating expression profile data with consensus pattern searches. *Nucleic Acids Res*., **29**, 3988–3996.

Gasch,A.P. *et al*. (2000) Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cell*, **11**, 4241–4257.

Gasch,A.P. *et al*. (2001) Genomic expression responses to DNA-damaging agents and the regulatory role of the yeast ATR homolog Mec1p. *Mol. Biol. Cell*, **12**, 2987–3003.

Güldener,U. *et al*. (2005) CYGD: the Comprehensive Yeast Genome Database. *Nucleic Acids Res*., **33**, D364–D368.

Huh,W.K. *et al*. (2003) Global analysis of protein localization in budding yeast. *Nature*, **425**, 686–691.

Jansen,R. (2003) A Bayesian networks approach for predicting protein–protein interactions from genomic data. *Science*, **302**, 449–453.

Joachims,T. *et al*. (1999) Making large-Scale SVM Learning Practical. In Schölkopf,B., Burges,C. and Smola,A. (eds), *Advances in Kernel Methods—Support Vector Learning*. MIT Press, Cambridge, MA.

Karaoz,U. *et al*. (2004) Whole-genome annotation by using evidence integration in functional-linkage networks. *Proc. Natl Acad. Sci. USA*, **101**, 2888–2893.

Lanckriet,G.R. *et al*. (2004a) A statistical framework for genomic data fusion. *Bioinformatics*, **20**, 2626–2635.

Lanckriet,G.R., Deng,M., Cristianini,N. *et al*. (2004b) Kernel-based data fusion and its application to protein function prediction in yeast. *Pac. Symp. Biocomput*.300–311.

Lee,W.L. *et al*. (2005) The offloading model for dynein function: differential function of motor subunits. *J. Cell Biol*., **168**, 201–207.

Miles,J. and Formosa,T. (1992) Evidence that POB1, a *Saccharomyces cerevisiae* protein that binds to DNA polymerase alpha, acts in DNA metabolism *in vivo*. *Mol. Cell. Biol*., **12**, 5724–5735.

Ogawa,N. *et al*. (2000) New components of a system for phosphate accumulation and polyphosphate metabolism in *Saccharomyces cerevisiae* revealed by genomic expression analysis. *Mol. Biol. Cell*. **11**, 4309–4321.

Pavlidis,P. *et al*. (2002) Learning gene functional classifications from multiple data types. *J. Comput. Biol*., **9**, 401–411.

Shakoury-Elizeh,M. (2004) Transcriptional remodeling in response to iron deprivation in *Saccharomyces cerevisiae*. *Mol. Biol. Cell*, **15**, 1233–1243.

Spellman,P.T. *et al*. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell*, **9**, 3273–3297.

Sudarsanam,P. *et al*. (2000) Whole-genome expression analysis of snf/swi mutants of *Saccharomyces cerevisiae*. *Proc. Natl Acad. Sci. USA*, **97**, 3364–3369.

The Gene Ontology Consortium (2000), Gene ontology: tool for the unification of biology. *Nat. Genet*., **25**, 25–29.

Troyanskaya,O.G. *et al*. (2001) Missing value estimation methods for DNA microarrays. *Bioinformatics*, **17**, 520–525.

Troyanskaya,O.G. *et al*. (2003) A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *Saccharomyces cerevisiase*). *Proc. Natl Acad. Sci. USA*, **100**, 8348–8353.

Yoshimoto,H. *et al*. (2002) Genome-wide analysis of gene expression regulated by the calcineurin/Crz1p signaling pathway in *Saccharomyces cerevisiae*. *J. Biol. Chem*., **277**, 31079–31088.