A Bayesian Integration Model for Improved Gene Functional Inference from Heterogeneous Data Sources

Noor Alaydie  
Dept. of Computer Science  
Wayne State University  
Detroit, MI 48202  
alaydie@wayne.edu

Chandan K. Reddy  
Dept. of Computer Science  
Wayne State University  
Detroit, MI 48202  
reddy@cs.wayne.edu

Farshad Fotouhi  
College of Engineering  
Wayne State University  
Detroit, MI 48202  
fotouhi@wayne.edu

ABSTRACT
Increasing amounts of biological data from various sources are being made available by high-throughput genomic technologies. However, no single biological data source analysis can fully unravel the complexities of the hierarchical gene function prediction. Therefore, the integration of multiple data sources is required to acquire a more precise understanding of the role of genes in the living organisms. In this paper, we develop a Hierarchical Bayesian Integration algorithm, HiBiN, a general framework that uses Bayesian reasoning to integrate heterogeneous data sources for accurate gene function prediction. The system uses posterior probabilities to assign class memberships to samples using multiple data sources while maintaining the hierarchical constraint that governs the annotation of genes. We demonstrate that the integration of the diverse datasets significantly improves the classification quality for hierarchical gene function prediction in terms of several measures, compared to single-source prediction models and fused-flat model, which are the baselines we compared against.

Categories and Subject Descriptors
J.3.1 [Computer Applications]: Life and Medical Science - biology and genetics; I.5.2 [Design Methodology]: [Classifier design and evaluation].

General Terms
Bioinformatics, Machine Learning, Algorithms.

Keywords
Data integration, gene function prediction, hierarchical multi-label classification.

1. INTRODUCTION

Diverse high-throughput genomic data are becoming widely available. Since the functions of a significant number of genes are still unknown, such high throughput data can play a vital role in assigning accurate functional annotations on large-scale through computational prediction. Providing accurate predictions can advance experimental studies by providing specific hypothesis for targeted experimental testing [13]. Learning reliable classification models from a single dataset is often hard due to several complex issues including noise in the data, low relevance of the dataset for some functional classes and an insufficient number of training examples for building accurate classification models [9].

Several issues have contributed to the complexity of the gene functional classification problem. First, a gene may have multiple class labels. The functional classes are related to each other in a tree or a graph structure (Gene Ontology(GO) [4] or MIPS’s FunCat taxonomy [11]). This leads to the second challenge which is known as the hierarchy constraint. In other words, annotating a gene to a given class is automatically transferred to all of its ancestors. Third, as the specificity of the functional classes increases, less number of genes are annotated to the more specific functions, this issue is known as the class imbalance problem. Finally, functionally-similar genes may have huge diversity on their measurements through various datasets. To the best of our knowledge, there is no existing work that handles all of the above challenges in a unified framework.

To address the above challenges and to improve hierarchical gene function prediction from diverse functional data, we propose HiBiN, a probabilistic framework for integrated analysis of multiple heterogeneous biological data. The major contributions of our new scheme can be summarized as follows:

1. To handle the problem of source diversity, the proposed framework integrates multiple heterogeneous data sources to characterize the genes effectively.

2. To predict multi-labels for genes, the HiBiN algorithm allows the prediction of more than one functional class for each gene.

3. To maintain the hierarchy constraint, the parent-child inter-relationships are exploited during the training and the testing phases. More specifically, HiBiN filters out unsuitable genes from percolating to lower levels in the hierarchy.

4. To handle the class imbalance issues, the positive and negative examples for each classifier are chosen based on the hierarchical taxonomy of the classes.
2. RELATED WORK

2.1 Single Source Gene Function Prediction

There are two approaches for gene function classifier training: a) Flat approach, where a classifier for each gene function is learned independently (Figures 1(a)) and (b) Hierarchical approach, where the relationships between gene functions are integrated for gene function classifier training (Figure 1(b)). The flat classification approach may lead to hierarchically inconsistent predictions. On the other hand, the hierarchical approach identifies the relevant positive and negative examples for each class according to the hierarchy. In most of the existing work, a separate binary classifier is learned for each class in the hierarchy.

2.2 Integration of Heterogeneous Data Sources

The value of integrating knowledge obtained from various sources has been illustrated by several studies. Troyanskaya et al. [13] proposed MAGIC framework that considers the outputs of several clustering methods applied on each data source separately and incorporates the knowledge of yeast biology experts in the prior probabilities of the Bayesian framework. Marcotte et al. [8] predicted a number of protein functions for *Saccharomyces cerevisiae* based on a heuristic combination of different data types. However, since the confidence levels of protein-protein links are defined on a case-by-case basis, the data sources are combined in a semi-manual and heuristic fashion. Functional linkage networks [3], kernel fusion [7], vector space integration [10] and ensemble systems are other approaches that have been proposed to deal with the data integration problem.

Limitations of the existing methods: the above discussed approaches focus either on discovering interacting sets of proteins only or on integrating multiple sources of data without taking into account the hierarchical relationships between the functional classes. As a consequence, blindly applying these data integration approaches to the hierarchical gene function prediction domain leads to serious inconsistencies due to the violation of the hierarchy constraint governing the functional annotations of genes in GO and FunCat taxonomies. The proposed method overcomes the above limitations.

3. THE PROPOSED FRAMEWORK

Let $\mathcal{G} = \mathbb{R}^d$ be the $d$-dimensional input space of genes and $\mathcal{Y} = \{y_1, y_2, ..., y_L\}$ be the set of $L$ labels. The hierarchical relationships are defined as follows: Given $y_i, y_j \in \mathcal{Y}$, $y_i$ is the ancestor of $y_j$, denoted by $(\uparrow y_i) = y_j$, if and only if $y_i$ is a superclass of $y_j$. We denote by $(\downarrow y_i) = y_j$, the set of children classes of class $y_i$. The set of labels $\mathcal{L}$ are structured according to a hierarchical structure $T$ (in our case according to a FunCat tree). A gene $g$ is represented with a vector of $d$ different features.

Let a training set $\mathcal{M} = \{< g_1, y_1 >, ..., < g_N, y_N >\}$, where $g_i \in \mathcal{G}$ is a feature vector for gene $i$ and $y_i \subseteq \mathcal{Y}$ is the set of labels associated with $g_i$, such that $y_i \in \mathcal{Y}_i \Rightarrow y_i' \in \mathcal{Y}_i, \forall (\uparrow y_i') = y_i$. A gene, $g$, is assigned to one or more functional classes. The assignments are represented through a vector $f_g = (y_1, y_2, ..., y_J) \in \{0, 1\}^J$, where $y_i = 1$ if the gene $g$ is annotated with class $y_i$, while $y_i = 0$ otherwise.

3.1 Boosting Classifiers

ADABOOST.MH [12] is a popular multi-class variant of the AdaBoost algorithm. The algorithm works by transferring a multi-class problem into a binary classification problem by replicating positive instances for the given class labels during the boosting process. After finishing all the iterations, the final hypothesis is generated by summing up all of the weak hypotheses, the classifier at each class will only be presented with examples of the given class.

Let $\mathcal{M}$ be the training set and $\mathcal{Y}$ be the set of labels. The AdaBoost algorithm works by transferring a multi-class problem into a binary classification problem by replicating positive instances for the given class labels during the boosting process. After finishing all the iterations, the final hypothesis is generated by summing up all of the weak hypotheses, the classifier at each class will only be presented with examples of the given class.

#### 3.1.1 Boosting Classifiers

The classifier at each class will only be presented with examples of the given class.

Let $\mathcal{M}_i = \{< g_1, y_1 >, ..., < g_Q, y_Q >\}$ be the training set for class $i$. The AdaBoost algorithm works by transferring a multi-class problem into a binary classification problem by replicating positive instances for the given class labels. After finishing all the iterations, the final hypothesis is generated by summing up all of the weak hypotheses, the classifier at each class will only be presented with examples of the given class.

Let $\mathcal{M}_i = \{< g_1, y_1 >, ..., < g_Q, y_Q >\}$ be the training set for class $i$. The AdaBoost algorithm works by transferring a multi-class problem into a binary classification problem by replicating positive instances for the given class labels. After finishing all the iterations, the final hypothesis is generated by summing up all of the weak hypotheses, the classifier at each class will only be presented with examples of the given class.

## 3.1.2 Boosting Classifiers

The classifier at each class will only be presented with examples of the given class.

Let $\mathcal{M}_i = \{< g_1, y_1 >, ..., < g_Q, y_Q >\}$ be the training set for class $i$. The AdaBoost algorithm works by transferring a multi-class problem into a binary classification problem by replicating positive instances for the given class labels. After finishing all the iterations, the final hypothesis is generated by summing up all of the weak hypotheses, the classifier at each class will only be presented with examples of the given class.

# 3.2 Hierarchical Training and Testing Schemes

The classifier at each class will only be presented with examples of the given class. Hence, the reached examples are positive examples to the current class and/or to the siblings of that class. The training for each classifier is performed by feeding as negative training examples, the positive examples at the parent
of the current that are not positive examples at the current class. It should be noted that the selected negative training examples are the most informative negative examples for training. The testing phase follows the same procedure.

3.3 Bayesian Integration and Classification

For our experiments, the probabilities obtained from boosting classifiers are the likelihood of observing gene \( i \) \( (g_i) \) associated with dataset \( j \) given a specific class. However, the posterior probability is the one of the primary interest:

\[
P(y_k|g_i,M_j) = \frac{P(g_i,M_j|y_k)P(y_k)}{\sum_{k=1}^{m} P(g_i,M_j|y_k)P(y_k)}
\]

(3)

Thus, for each sample, a set of probabilities are obtained which sum to one.

Bayes formula can be used to integrate and compute posterior probabilities for multiple datasets. With the assumption that each dataset is independent but describe the same genes, the integrated likelihood is computed as the product of the individual likelihoods:

\[
P(g_i,M_1,...,M_Q|y_k) = P(g_i,M_1|y_k) \times \ldots \times P(g_i,M_Q|y_k)
\]

(4)

Given that the datasets share common genes, the datasets can have some correlation. The probability of a particular gene is given by

\[
P(y_k|g_i,M_1,...,M_Q) = \frac{\prod_{j=1}^{Q} [P(g_i,M_j|y_k)P(y_k)]}{\sum_{k=1}^{m} \prod_{j=1}^{Q} [P(g_i,M_j|y_k)P(y_k)]}
\]

(5)

Each non-root class, \( y_i \), has a binary classifier \( \phi_j \) that is associated with it. The classifier should act as a “filter” to prevent unsuitable examples from spreading out to the lower levels in the hierarchy. Hence, only the test genes that a classifier \( \phi_j \) decides to belong to \( y_i \) are passed to all the binary classifiers corresponding to the children classes of \( y_i \). While the genes that classifier \( \phi_j \) sees not to belong to \( y_i \) are “blocked” and no further analysis is carried out.

HiBiN works by first computing the prior probabilities for all the classes. Next, the children classes of the current class are extracted; then ADABOOST.MH is called on these classes. Then, Bayesian posteriors are calculated for the children classes of the current class.

To make a decision whether a gene is annotated with a particular class or not, we compute the posterior probability for each gene using the following Bayesian decision rule:

\[
\text{Decide } y_k \text{ if } P(y_k|g_i) > P(y_{k'}|g_i); \text{ otherwise decide } y_{k'}
\]

where \( y_{k'} = 0 \). The computation of \( P(y_{k'}|g_i) \) is similar to the computation of \( P(y_k|g_i) \), except that here we compute the probability of the negative class \( (g_i) \) and the likelihood given by \( P(g_i|y_k = 0) \).

4. EXPERIMENTS

4.1 Datasets

We demonstrate the performance of HiBiN for the prediction of gene functions in Saccharomyces cerevisiae. Six different data sources of biomolecular data [14] were integrated. The datasets include (i) gene expression data (Gene-Expr), two types of protein domain data; (ii) protein domain binary data (Pfam-Binary) and (iii) Pfam protein domains with log E values computed by the HMMER software toolkit (Pfam-LogE), predicted and experimentally supported protein-protein interaction data from (iv) Von Mering experiments (PPI-STRING), and from (v) BioGrid (PPI-BioGRID) and (vi) pairwise sequence similarity data (Seq-Sim). For the integration purpose, we considered the genes that are common to all datasets. Moreover, for each dataset, we selected FunCat annotated genes for the classes with at least 20 positive examples so that the set of positive examples used for training is not too small. This yielded 1901 yeast genes annotated to 168 FunCat classes distributed across 16 trees and 5 hierarchical levels.

4.2 Baseline Methods

We compared our approach to two baseline algorithms: flat integration and hierarchical single source. In flat integration, the hierarchy constraint is not taken into consideration when building the classifiers, and the integration is performed using the same framework we used for HiBiN method. While the hierarchical single source method is applied on each dataset separately.

To evaluate HiBiN, we used 3-fold cross validation with 100 boosting iterations. Moreover, we tested two strategies to compute the prior probabilities. In one strategy, we computed the prior probabilities without considering the hierarchical structure of the classes. In other words, the prior probabilities are computed by considering the set of annotated genes at each functional label with respect to the total number of genes. This version of the algorithm is called HiBiNa, where a stands for all. In the second variation, the prior probabilities are computed while the hierarchical scheme of the classes is taken into account. In other words, the prior probabilities are computed by considering the set of annotated genes at each functional label with respect to the number of genes that are annotated with the parent functional class of the label of interest. This version of the algorithm is called HiBiNp, where p stands for parent.

4.3 Evaluation Metrics

In order to evaluate our algorithm, we adopted the classical and the hierarchical performance evaluation measures. \( F_1 \) measure considers the joint contribution of both precision (P) and recall (R). \( F_1 \) measure is defined as follows:

\[
F_1 = \frac{2 \times P \times R}{P + R} = \frac{2TP}{2TP + FP + FN}
\]

(6)

where TP stands for True Positive, TN for True Negative, FP for False Positive and FN for False Negative. When \( TP=FP=FN=0 \), we made \( F_1 \) measure to equal to 1 as the classifier has correctly classified all the examples as negative examples [5]. Hierarchical measures are defined as follows:

\[
hP = \frac{1}{|\{l(P(x))\}|} \sum_{p \in \{l(P(x))\}} \frac{|C(x) \cap \uparrow p|}{|\uparrow p|}
\]

(7)

\[
hR = \frac{1}{|\{l(C(x))\}|} \sum_{c \in \{l(C(x))\}} \frac{|\uparrow c \cap P(x)|}{|\uparrow c|}
\]

(8)

\[
hF = \frac{2 \times hP \times hR}{hP + hR}
\]

(9)

where \( hP \), \( hR \) and \( hF \) stands for hierarchical precision, hierarchical recall and hierarchical F-measure, respectively.
$P(x)$ is a subgraph formed by the predicted class labels for the instance $x$ while $C(x)$ is a subgraph formed by the true class labels for the instance $x$. $p$ is one of the predicted class labels and $c$ is one of the true labels for instance $x$. $l(P(x))$ and $l(C(x))$ are the set of leaves in graphs $P(x)$ and $C(x)$, respectively. We also computed micro-averaged hierarchical F-measure ($hF^M_1$) and macro-averaged hierarchical F-measure ($hF^M_1$) is computed by calculating $hP$ and $hR$ for each path in the tree and then applying Equation 9. On the other hand, $hF^M_1$ is computed by calculating $hF_1$ for each path in the hierarchical structure of the classes independently and then averaging them.

The comparison between HiBiN algorithm and the flat method is based on the “per-class” $F_1$-measure that is obtained by averaging the $F_1$-measure for all the classes in the FunCat hierarchy for each dataset. In other words, an overall $F_1$-measure is obtained by computing the $F_1$-measure for each class separately and then averaging them across all the classes. Furthermore, to get more insights into the performance of the HiBiN algorithm, we performed a level-wise analysis of the precision, recall and $F_1$-measure on the baselines and the proposed algorithm. In measuring the level-wise performance, level 1 represents the top level in the hierarchy while level 5 is the deepest level in the hierarchy.

4.4 Results

In our experiments, after preprocessing the datasets as described above, the prior probabilities are computed for each label in the FunCat scheme. Prior probabilities are computed based on the number of positive examples annotated with each label obtained from all the datasets. Next, boosting classifiers are used to obtain the likelihoods from the different datasets. Finally, the Bayesian posteriors are computed to obtain the probabilities that are used to make the final decision about the classification.

Table 1 summarizes the results of the comparisons of the average per-class precision, recall and F-measure for the hierarchical single source, flat integration and HiBiN methods. The comparison between HiBiN algorithm and the flat method performed better on the PPI-BG and PPI-STRING datasets than the flat integration in terms of the per-class precision and recall, while the flat integration performed slightly better than the hierarchical single source method on all the datasets, in terms of per-class F-measure.

In Table 2, we compare the hierarchical precision, hierarchical recall, hierarchical F-micro ($hF^{\text{micro}}_1$) and hierarchical F-macro ($hF^{\text{macro}}_1$) measures for both hierarchical single source and HiBiN methods. Note that hierarchical precision and hierarchical recall are not applicable to flat integration method as flat integration does not take the hierarchy constraint into account. Hence, the obtained predictions may be inconsistent with the hierarchy constraint. As we can observe, HiBiN achieved the best results in terms of all the hierarchical measurements. To a large extent, the two variations of HiBiN have similar performances, which indicates that incorporating the parent-child relationship in the computation of the prior probabilities does not add a noticeable improvement to the overall integration scheme.

A closer look into per-level precision, recall and F-measure

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Single Source</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPI-BG</td>
<td>HiBiN</td>
<td>0.5211</td>
<td>0.3176</td>
<td>0.4081</td>
</tr>
<tr>
<td>PPI-STRING</td>
<td>HiBiN</td>
<td>0.4882</td>
<td>0.3030</td>
<td>0.3823</td>
</tr>
<tr>
<td>Pfam-Binary</td>
<td>HiBiN</td>
<td>0.2977</td>
<td>0.1035</td>
<td>0.3901</td>
</tr>
<tr>
<td>Pfam-LogE</td>
<td>HiBiN</td>
<td>0.2244</td>
<td>0.2076</td>
<td>0.3837</td>
</tr>
<tr>
<td>Gene-Expr</td>
<td>HiBiN</td>
<td>0.2301</td>
<td>0.2107</td>
<td>0.3805</td>
</tr>
<tr>
<td>Seq-Sim</td>
<td>HiBiN</td>
<td>0.2136</td>
<td>0.2074</td>
<td>0.3545</td>
</tr>
<tr>
<td>Integration Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flat</td>
<td></td>
<td>0.3592</td>
<td>0.307</td>
<td>0.4521</td>
</tr>
<tr>
<td>HiBiN(_a)</td>
<td></td>
<td>0.7436</td>
<td>0.4535</td>
<td>0.6175</td>
</tr>
<tr>
<td>HiBiN(_p)</td>
<td></td>
<td>0.7083</td>
<td>0.4779</td>
<td>0.6222</td>
</tr>
</tbody>
</table>

Figure 2: Comparison of the per-level average precision across the five levels of the FunCat taxonomy using flat integration, hierarchical single source (applied on PPI-BG dataset), HiBiN\(_a\) and HiBiN\(_p\) methods. BG stands for PPI-BioGrid dataset.

Figure 3: Comparison of the per-level average recall across the five levels of the FunCat taxonomy using flat integration, hierarchical single source (applied on PPI-BG dataset), HiBiN\(_a\) and HiBiN\(_p\) methods. BG stands for PPI-BioGrid dataset.
Table 2: Hierarchical precision, hierarchical recall, hierarchical F-micro and hierarchical F-macro measures obtained from hierarchical single source (on each dataset separately) and HiBiN methods.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Single Source</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Micro</th>
<th>F-Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPI-BG</td>
<td>0.9259</td>
<td>0.6071</td>
<td>0.6953</td>
<td>0.7147</td>
<td></td>
</tr>
<tr>
<td>PPI-STRING</td>
<td>0.8882</td>
<td>0.5933</td>
<td>0.6846</td>
<td>0.6926</td>
<td></td>
</tr>
<tr>
<td>Pfam-Binary</td>
<td>0.8974</td>
<td>0.5912</td>
<td>0.6882</td>
<td>0.7006</td>
<td></td>
</tr>
<tr>
<td>Pfam-LogE</td>
<td>0.7340</td>
<td>0.5593</td>
<td>0.6115</td>
<td>0.6348</td>
<td></td>
</tr>
<tr>
<td>Gene-Expr</td>
<td>0.7426</td>
<td>0.5801</td>
<td>0.6414</td>
<td>0.6644</td>
<td></td>
</tr>
<tr>
<td>Seq-Sim</td>
<td>0.7274</td>
<td>0.5574</td>
<td>0.6067</td>
<td>0.6312</td>
<td></td>
</tr>
<tr>
<td>Integration methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HiBiN</td>
<td>0.9492</td>
<td>0.6298</td>
<td>0.7361</td>
<td>0.7572</td>
<td></td>
</tr>
<tr>
<td>HiBiN_p</td>
<td>0.9442</td>
<td>0.6329</td>
<td>0.7355</td>
<td>0.7579</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Comparison of the per-level average F-measure across the five levels of the FunCat taxonomy using flat integration, hierarchical single source (applied on PPI-BG dataset), HiBiN_s and HiBiN_p methods. BG stands for PPI-BioGrid dataset.

5. CONCLUSION

In this paper, we developed HiBiN, a general probabilistic framework for gene function prediction through the integration of heterogeneous data sources while maintaining the hierarchy constraint among the functions. Our results showed that the integration can improve the performance of the standard classification-based gene function prediction algorithms. Our future work includes establishing a probabilistic weighting scheme of data sources.

6. REFERENCES