

AUTOMATIC ASSIGNMENT OF PROTEIN FUNCTION
WITH SUPERVISED CLASSIFIERS

A Dissertation

by

JAE HEE JUNG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2008

Major Subject: Computer Science

AUTOMATIC ASSIGNMENT OF PROTEIN FUNCTION
WITH SUPERVISED CLASSIFIERS

A Dissertation

by

JAE HEE JUNG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Michael R. Thon
Committee Members,	Christine Elsik
	Ricardo Gutierrez-Osuna
	Thomas Ioerger
Head of Department,	Valerie E. Taylor

August 2008

Major Subject: Computer Science

ABSTRACT

Automatic Assignment of Protein Function with Supervised
Classifiers. (August 2008)

Jae Hee Jung, B.S., Dongduk Women's University;
M.S., Korea University

Chair of Advisory Committee: Dr. Michael R. Thon

High-throughput genome sequencing and sequence analysis technologies have created the need for automated annotation and analysis of large sets of genes. The Gene Ontology (GO) provides a common controlled vocabulary for describing gene function. However, the process for annotating proteins with GO terms is usually through a tedious manual curation process by trained professional annotators. With the wealth of genomic data that are now available, there is a need for accurate automated annotation methods.

The overall objective of my research is to improve our ability to automatically annotate proteins with GO terms. The first method, Automatic Annotation of Protein Functional Class (AAPFC), employs protein functional domains as features and learns independent Support Vector Machine classifiers for each GO term. This approach relies only on protein functional domains as features, and demonstrates that statistical pattern recognition can outperform expert curated mapping of protein functional domain features to protein functions. The second method Predict of Gene Ontology (PoGO) describes a meta-classification method that integrates multiple heterogeneous data sources. This method leads to improved performance than the protein domain method can achieve alone.

Apart from these two methods, several systems have been developed that employ pattern recognition to assign gene function using a variety of features, such as

the sequence similarity, presence of protein functional domains and gene expression patterns. Most of these approaches have not considered the hierarchical relationships among the terms in the form of a directed acyclic graph (DAG). The DAG represents the functional relationships between the GO terms, thus it should be an important component of an automated annotation system. I describe a Bayesian network used as a multi-layered classifier that incorporates the relationships among GO terms found in the GO DAG. I also describe an inference algorithm for quickly assigning GO terms to unlabeled proteins. A comparative analysis of the method to other previously described annotation systems shows that the method provides improved annotation accuracy when the performance of individual GO terms are compared. More importantly, this method enables the classification of significantly more GO terms to more proteins than was previously possible.

To my parents, family members and my husband Gangman Yi.

ACKNOWLEDGMENTS

I would like to thank Dr. Michael R. Thon for his continual encouragement, formidable assistance, and guidance throughout my graduate studies. Additionally, I also would like to thank my committee members, Dr. Ricardo Gutierrez-Osuna, Dr. Thomas Ioerger and Dr. Christine Elsik, for their knowledge and advice. Finally, I would like to express my sincere gratitude to all those who have prayed for me.

TABLE OF CONTENTS

CHAPTER		Page
I	INTRODUCTION	1
	A. Motivation	1
	B. Objectives	5
II	AAPFC: AUTOMATIC ANNOTATION OF PROTEIN FUNCTIONAL CLASS FROM SPARSE AND IMBALANCED DATA SETS	10
	A. Related Work	10
	B. Proposed Method	13
	1. Data Set	13
	2. Under-Sampling	14
	3. Feature Selection	15
	4. Filtering Step	16
	C. Experiments	17
	1. Feature and Instance Selection	18
	2. Test Procedure	20
	3. Comparison to Other Methods	23
	D. Taxon-Specific Training Model	24
	1. Data Sets	25
	2. Comparison of Taxon-specific Models	27
	E. Summary	33
III	PREDICTION OF GO (POGO): A MULTI-FEATURE LEARNING SCHEME FOR ASSIGNING GENE ONTOLOGY TERMS TO PROTEINS	35
	A. Related Work	35
	B. Proposed Method	39
	1. Data Set	39
	C. Experiments	42
	1. Training Procedure (Multi-layered Classifier)	42
	a. The Base-Classifiers	42
	b. The Meta-Classifier	44
	2. Testing Procedure	45

CHAPTER		Page
	3. Results	46
	D. Implementation	52
	1. Batch Job Queuing System	53
	2. System Output and Web Server	53
	3. User Interface	55
	E. Summary	57
IV	GENE FUNCTIONAL PREDICTION USING HIERARCHICAL GENE ONTOLOGY INFORMATION	59
	A. Related Work	59
	B. Protein Domain Feature	62
	1. Proposed Method	62
	a. Data Set	62
	b. Bayesian Network	64
	2. Experiment	66
	a. Training Procedure	66
	b. Test Procedure	66
	c. Filtering Step	69
	3. Results	71
	C. Multiple Features with a Multi-layered Classifier	76
	1. Data Set	76
	2. Experiment	78
	a. Training Procedure	78
	b. Testing Procedure	79
	3. Results	82
	D. Implementation	88
	E. Summary	91
V	CONCLUSION	93
	A. Summary	93
	B. Future Work	95
	REFERENCES	97
	VITA	107

LIST OF TABLES

TABLE		Page
I	Examples of randomly selected classes (GO terms) and features (InterPro terms) illustrating the imbalanced and sparse nature of the data set.	13
II	Filtering step by removing false positives.	17
III	Performance comparison of 4 different feature selection methods (SU: symmetrical uncertainty, INFO: information gain, CHI: chi-squared, ABS: absolute correlation coefficient).	19
IV	Performance comparison of 4 different under-sampling methods.	19
V	A summary of the data sets for taxon-specific models.	26
VI	Performance matrices of various data sets for fungi.	28
VII	Performance comparison of four different taxon-specific models.	30
VIII	Performance comparison of the random selected shared GO terms between fungi set and sampled UniProt.	32
IX	Performance comparison of four different base-classifiers with different feature sets.	46
X	Performance comparison of two different meta-learning methods with all GO classifiers.	47
XI	Mean classifier-based performance at each cut-off F-measure value.	48
XII	Mean protein-based performance at each cut-off F-measure value.	49
XIII	Classifier-based performance comparison of 12 shared GO terms among 409 GO terms in AAPFC and PoGO.	51

TABLE	Page
XIV	Properties of GO annotation tools: ‘o’= support completely, ‘Δ’= support partially and ‘×’= does not provide yet. GOPET does not support Cellular Component category. 55
XV	Comparison of ”WITH the filtering step” and ”WITHOUT the filtering step”. 71
XVI	Average level of annotated GO terms by the hierarchical GO-structured model with InterPro terms. 75
XVII	Number of trained GO terms in three GO categories. 76
XVIII	F-measure comparison of four different gene function annotation systems. 77
XIX	Classifier-based performance comparison in shared GO terms with the hierarchical GO-structured model using InterPro terms or multi-features. 83
XX	Classifier-based performance comparison in shared GO terms with PoGO and the hierarchical GO-structured model using multi-features set. 90

LIST OF FIGURES

FIGURE		Page
1	The glutathione biosynthesis biological process overview in the GO structure.	2
2	Flow chart for the training process.	21
3	Performance of SVMs estimated by 10-fold cross validation. (a) Cumulative count of number of false negative errors produced by 374 independent SVMs. (b) False positive errors produced by <i>Model 1</i> and <i>Model 2</i> for five randomly selected GO term data sets.	22
4	Flow chart for the testing process.	22
5	Precision of the two processes.	23
6	Comparison of the precision of the proposed classification method (AAPFC) to four other methods.	24
7	Multi-layered classifier with four feature sets: Two subsets (white and gray) are used for training and one subset (black) is used for validation. The dotted lines represent the process of testing and the solid lines in a meta-classifier are the process of training.	43
8	AAPFC and PoGO: (a) The number of GO terms at each cut-off F-measure value. (b) The number of annotated proteins at each cut-off F-measure value.	50
9	Average F-measure value at each cut-off F-measure value in AAPFC and PoGO.	52
10	System architecture in PoGO.	54
11	Web-server page for AAPFC.	56
12	Hierarchical structure of the Gene Ontology.	64

FIGURE	Page	
13	GO terms in cellular component structure annotated in CHAC_YEAST: The dark gray GO terms (GO:0005634 and GO:0005737) are what a protein has, but light gray GO terms are not included in the original training sets. From the assumption of the experiment, these light gray GO terms are considered to be training GO terms. The white GO terms are not considered for the training sets, since they are not linked to any dark gray GO terms.	65
14	Inference algorithm to annotate GO terms using the hierarchical GO-structured model with InterPro terms.	68
15	Construction of the Bayesian network from the root node in each three categories.	70
16	AAPFC and the hierarchical GO-structured model with InterPro terms: (a) The number of GO terms at each cut-off F-measure value. (b) The number of annotated proteins at each cut-off F-measure value.	73
17	AAPFC and the hierarchical GO-structured model with InterPro terms. The average F-measure at each cut-off F-measure value. . . .	74
18	Inference algorithm to annotate GO terms with the hierarchical GO-structured model with multi-features.	81
19	The hierarchical GO-structured model with InterPro terms or multi-features: (a) The number of GO terms at each cut-off F-measure value. (b) The number of annotated proteins at each cut-off F-measure value.	84
20	Average F-measure at each cut-off F-measure value in the hierarchical GO-structured model with InterPro terms and multi-features.	85
21	Number of GO terms at each cut-off F-measure value in PoGO and the hierarchical GO-structured model with multi-features.	86
22	PoGO and the hierarchical GO-structured model with multi-features set: (a) The number of annotated proteins at each cut-off F-measure value. (b) The number of annotated proteins excluding the two highly annotated GO terms.	87

FIGURE	Page
23 PoGO and the hierarchical GO-structured model with multi-features set: (a) The average F-measure at each cut-off F-measure value. (b) The average F-measure excluding the two highly annotated GO terms.	89

CHAPTER I

INTRODUCTION

A. Motivation

Since the development of high-throughput genome sequencing and gene annotation methods, large sets of genes and predicted gene products are available. However, the functions of many of these genes are still unknown, i.e., they remain unannotated. Biologists deduce protein function through experimentation but knowledge of gene function derived in this fashion is laborious and expensive. Traditionally, protein function is expressed as free text descriptions, but recently, controlled vocabularies of various types have been employed. The Gene Ontology (GO) [1], which is a controlled vocabulary of terms for annotating proteins, is used to represent protein function. Every GO term has a unique numerical identifier that represents the gene function. Each GO term is assigned to one of the three categories of *molecular function*, *biological process* or *cellular component*. These terms are organized into a Directed Acyclic Graph (DAG) which provides a rich framework for describing the function of proteins. Each GO term has a more specific GO term (child) and more than one less specific term (parent). For example, GO:0006750 in Fig. 1 has three parent terms, GO:0044427, GO:0006749 and GO:0009108, which are more general functions than GO:0006750. GO are assigned by curators who examine references in the scientific literature as well as the features of proteins. Given the wealth of genome data that are available now, one of the most important problems for researchers is to devise the best computational model that can accurately predict protein functions from the available evidence.

The journal model is *IEEE Transactions on Automatic Control*.

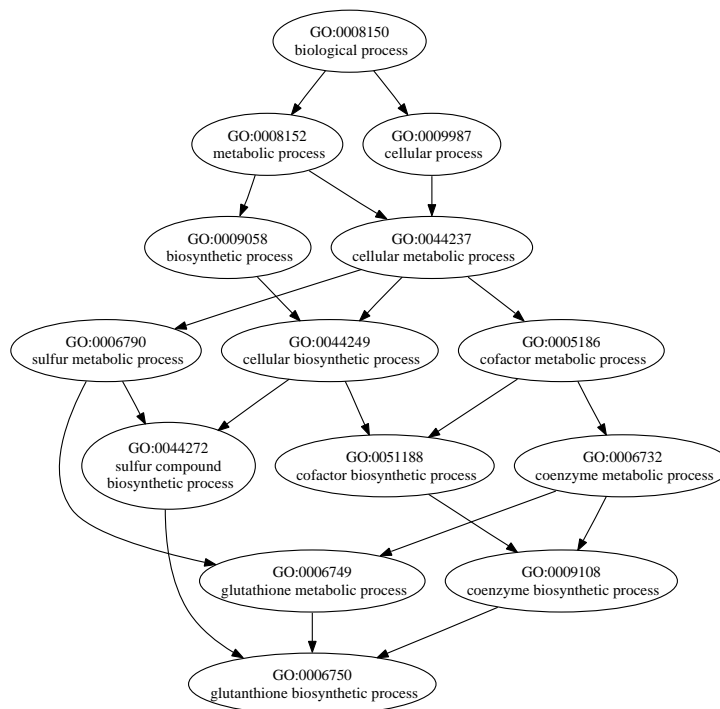


Fig. 1. The glutathione biosynthesis biological process overview in the GO structure.

Manual gene curation remains the de facto standard for high quality functional annotations, however, it is too slow and labor intensive to apply to draft genome annotations that change frequently. Nevertheless, functional annotations improve the value of hypothetical genes, enabling their use in whole genome analyses such as microarray analyses. Thus, there is a strong motivation to find efficient automated methods of the functional annotation. Automated annotation can take on several forms. Simple decision-making logic, such as transferring annotations from top BLAST hits is often used to provide preliminary annotations for proteins or expressed sequence tags (ESTs). Manually curated mapping between vocabularies such as the InterPro2GO mapping is also used to convert annotations from disparate sources into a common controlled vocabulary. The GOA project relies on several

mappings to provide automated GO annotations for users ([2, 3]).

Various kinds of annotation systems [4, 5] are being developed for automated prediction of GO terms, but most methods rely on the identification of similar proteins in large databases of annotated proteins [6, 7, 8, 9]. OntoBlast matches GO terms to new proteins using a gene association link based on similarity [6]. Goblet [7] counts the cumulative GO terms obtained from the BLAST output, and then systematically assigns common parent GO terms. The predicted terms offer the broad function rather than the specific meaning, because the predicted level in the GO structure is relatively higher than the GO terms of the BLAST hits. GoFigure [8] is a web server that searches the homology and constructs the minimum covering graph and annotating terms. These terms are assigned based on the score of BLAST *e-value* and the given threshold value. Gotcha [9] provides a normalized confidence value for the relationships between the sequence similarity search and the GO tree. All these systems are based on the assumption that if two sequences have a high degree of similarity, they have evolved from a common ancestor and so have a similar function.

The protein domain is basically localized regions of high sequence similarity that have defined functions. Many databases including Prosite, Prints, Pfam, ProDom, SMART, and PIR SuperFamily are sequence-domain-based databases. Thus, I suggest a new model with the protein domain property. The European Bioinformatics Institute (EBI) has created a federated database called InterPro (IPR) [10] which serves as a central reference for several protein families and functional domain databases. These functional domain data serve as useful resources for understanding protein function. Schug et al. [11] use ProDom [12] and the NCBI Conserved Domain Database (CDD) [13] as features and assign the most common functions in the protein family. This approach is also based on the BLAST results and assigns the most common protein domain as a function. This is a relatively conservative approach. In addition,

the InterPro consortium also maintains an InterPro2GO translation table that allows GO terms to be assigned to proteins automatically, on the basis of the domain content of the protein.

Human-curated gene annotators rely on features (evidence) derived from a variety of sources, thus, a supervised classifier that tries to emulate this approach needs to combine heterogeneous data sources. I add additional feature sets to the second classifier that I describe in this dissertation. Prediction of Gene Ontology terms with a multi-feature learning scheme (PoGO) involves adding more feature sets by employing a multi-layered classifier. I describe a multi-layered classifier which comprises a base-classifier and a meta-classifier. A base-classifier is a single classifier with each feature and a meta-classifier is combining these feature sets. In order to distinguish the original meta-learning classifier, which usually use same feature domain but different learners in the second layer, I use the term multi-layered classifier. Various studies describe the integration of many feature sets, such as combining networks of functional linkages [14, 15, 16]. These approaches are based on two genes that are functionally linked, when two genes have similar phylogenetic profiles. But this approach uses very simple, heuristic inferences. Pellegrini et al. [16] also integrate a trained SVM. Troyanskaya et al. [17] proposed using hierarchical information for the combining of data. However, this method does not consider the data properties but just merges all feature sets. Thus, I suggest a method for merging the various feature sets that considers data properties and uses a multi-layered learning classifier. In comparison to my previous approach (AAPFC), the additional heterogeneous data sources increases the classifiers performance. I will show overall performance matrices measured with sensitivity, precision, and F-measure.

One of the basic properties of two approaches described above is that each GO classifier is independent and makes no consideration of the DAG structure of the GO.

Eisner et al. have [18] shown that hierarchical information contributes to the improved classifier performance when they included ancestor and descendant relationships in their training sets. Shahbaba and Neal [19] also suggested the hierarchical model, but one in which each node has only one parent node, whose property is not enough to be applied to the GO structure. Deng et al. and Troyanskaya et al. [20, 21] employed a Bayesian network to combine the various types of data considered, however, this approach considers the combination of the different types of data sets with a Bayesian network rather than the GO structure itself. Hence, I propose two methods using a Bayesian network with a protein domain, or multi-label classifier that incorporates the relationships between the terms found in the GO DAG. This approach enables us to determine the importance of GO structural information by comparing non-hierarchical model to the hierarchical model using the same feature sets. The result will be analyzed based on the comparison between the different tools or shared GO terms in previous models.

B. Objectives

The overall objective of the proposed research is to improve our ability to automatically annotate proteins with GO terms. I propose three methods which automatically assign GO terms by applying techniques of statistical pattern recognition. Each method employs different methods with different attributes. The first method is a classifier for each GO term using protein domains as features. This simple method outperforms other publically available methods. The second approach is a multi-layered learner using multi-features, which enables us to apply proteins that have no InterPro terms. The third one is a classifier that makes use of the GO hierarchical structure. One of the major differences between those three methods is that the third

method considers GO hierarchical properties, where the two former methods assume that the GO terms are independent. Moreover, the third method uses all fungi GO terms, while AAPFC and PoGO use only restricted GO terms. Based on the three proposed methods, I evaluate the predicted accuracy in fungi.

- **Automatic annotation of protein functional class from sparse and imbalanced data sets (AAPFC)**

The availability of protein data sets annotated with GO terms and InterPro domains provides an opportunity to study the extent to which InterPro can be used to predict GO terms. However, this approach creates sparse data sets with highly imbalanced class distribution. These problems can be overcome by using standard feature and instance selection methods. In the preliminary test, I will train Support Vector Machines (SVM) with several objective functions for the feature selection and four different sampling techniques for the instance selection. Finally, I set up the optimized model with SVM and I intend to demonstrate that a supervised learning approach using InterPro terms as features outperforms the manually curated mapping table, as well as several other publically available annotation tools. including Gotcha, GOPET and GOFigure. Moreover, taxon-specific models are compared to the general model, because I use the fungi-specific models in this dissertation.

Many studies use taxon-specific data, i.e., protein data sets derived from a small number of closely related species such as fungi and bacteria [22, 20, 21]. Since these data sets are small, they are subject to over-fitting, which is one issue for taxon-specific model. Over-fitting can be caused by the amount of training, too many parameters or small training sets. In this dissertation, I will also show that taxon-specific supervised classifiers can outperform the non-specific classifiers that are trained with larger data sets. In addition, I plan to prove that taxon-specific model is not over-fitted.

Two supplementary experiments are presented. One demonstrates the effect of different amount of training on the different species. The learning models in AAPFC depend on the positive training size, because the training data is composed of the same size of number of positive and negative instances, where the positive instances are the proteins which are annotated by a specific GO term. In order to show the different performances of various training sizes on the same classifiers, I plan to use various taxon-specific data sets. However, the above experiment have different tested sets, i.e., the large taxon-specific model is tested by many proteins, resulting in being a lower performance. Thus, the other experiment creates the same sized training sets in fungi and sampled Uniprot. By creating a same size of training sets, I will be able to viably contrast the performance.

● **Prediction of GO (PoGO): A multi-feature learning scheme for assigning Gene Ontology terms to proteins**

A problem with AAPFC is that it depends on functional domain alone to assign the GO term. In order to enhance the prediction performance, multi-feature types will be employed as a feature. The treated features are protein functional domains, sequence similarities, bio-chemical properties and protein structure information. Using these feature sets, I build two-layered models which contain base-classifiers and a meta-classifier. The base-classifier is the same approach as that in AAPFC and the meta-classifier allows for further learning from the results of the base-classifier, where the result of the base-classifier is produced from the two feature subsets. Each feature in a base-classifier is trained and tested by SVM or adaboosting depending on the data properties. The binary format data is trained with the designated GO term with the Naïve bayes in a meta-classifier, since each feature set is independent. Based on 10-fold cross validation, I will compare the performances between a single classifier and a multi-layered learning classifier and demonstrate the superiority of

the multi-layered learning approach.

- **Gene functional prediction using hierarchical Gene Ontology information**

For two described approaches, I assume that the training models are independent for each GO term. However, the GO has the structure of a Directed Acyclic Graph (DAG). Moreover, the dimension reduction in the form of features and instance selection results in lost information. In order to overcome these shortcomings, I propose the hierarchical GO-structured model with InterPro terms or multi-features by the conditional probability, where the treated classifiers are whole fungi rather than restricted GO terms. A training model is the conditional probability which is the Naïve bayes probability of the functional domain features or multi-feature sets in the Uniprot, given the true or false state of the GO terms, since all features are independent. With this training model, I construct the Bayesian network and assign the gene function. A comparative analysis of the method to other previously described annotation systems shows that this method provides improved accuracy when the performances of individual GO terms are compared. The number of GO terms and the number of annotated proteins are also greatly increased because this approach is effected using embedding hierarchical information, unlike the previous two approaches.

- **Evaluation of each learning method and implementation**

I compare the performance of my new methods to several published automated GO annotation tools, including GOPET, Gotcha, GOFigure and InterPro2GO. I use sensitivity, specificity, precision and F-measure as performance metrics. Sensitivity is the proportion of GO term annotations in the training data sets that were correctly predicted by the classifier and specificity is the proportion of GO term annotations not found in the training data sets that were correctly NOT predicted by the classifier. Precision is the proportion of GO term in a training sets that original training

sets have. Sensitivity and precision are also combined into a single metric called F-measure which is used as an overall performance metric. Since most tools are web-based systems, their performance is analyzed using randomly selected proteins from my training data sets. Moreover, all proposed methods are evaluated by the 10-fold cross validation and compared using the F-measure value in a various aspect. Among the four different performance matrices, F-measure is employed for the comparison to other tools and for deciding the cut-off level.

With these four objectives, I organize the dissertation as follows. In Chapter II, I describe a method for assigning GO terms to proteins using InterPro terms as features and learning independent SVMs for each GO term. In addition, I demonstrate that a taxon-specific model outperforms any general training models through a comparison of various taxon-specific training models. In Chapter III, I review the related work on the multi-layered classifier and proposed a multi-layered method using heterogeneous data. I also show that this approach assigns GO terms to more proteins than a single feature set. In Chapter IV, I discuss the importance of the hierarchical structure for GO term classifiers. Based on the GO DAG structure, a novel classifier based on a Bayesian network is suggested. The performance metrics are compared to those of previous approaches. In Chapter V, I summarize my contributions to gene functional prediction and conclude the dissertation by discussing future works.

CHAPTER II

AAPFC: AUTOMATIC ANNOTATION OF PROTEIN FUNCTIONAL CLASS
FROM SPARSE AND IMBALANCED DATA SETS

A. Related Work

High-throughput genome sequencing and gene annotation methods have resulted in the availability of large sets of genes and predicted gene products (proteins), and to a large extent, the functions of many of these genes are still unknown, i.e., they are unannotated. Given the wealth of genome data that are available now, one of the central problems facing researchers is the accurate prediction of protein function based on computationally obtained features of the proteins and the genes from which they are derived. Such computationally predicted functions are useful to guide laboratory experimentation and as an interim annotation, until protein function can be validated experimentally. Traditionally, protein function is expressed as free text descriptions, but recently controlled vocabularies of various types have been employed. The Gene Ontology (GO) [1] provides a controlled vocabulary or terms for annotating proteins. In addition, the GO consortium describes the relationships among the terms with a directed acyclic graph (DAG), providing a rich framework for describing the function of proteins. GO terms are often assigned to proteins by teams of curators, who examine references in the scientific literature as well as features of the proteins. One of the central problems facing computational biologists is how to emulate this process.

As the need for GO annotation increases, various kinds of annotation systems are being developed for automated prediction of GO terms [5]. Most methods rely on the identification of similar proteins in large databases of annotated proteins. GOTcha [9] utilizes properties of the protein sequence similarity search results (BLAST) such as

the p-score, for predicting an association between the protein and a set of nodes in the GO graph. Several other recently described methods, including GoFigure [8], Goblet [7], and OntoBlast [6] depend on sequence similarity searches of large databases to obtain features that are used for predicting GO terms. These tools employ only BLAST [23] results as attributes for prediction of GO terms, however, several systems utilize features besides BLAST search results. Vinayagam et al. [24, 25] suggest a method to predict GO terms using SVM and feature sets including sequence similarity, frequency score of the GO terms, and the GO term relationship between similar proteins. Al-shahib et al. [26] use amino acid composition, amino acid pair ratios, protein length, molecular weight, isoelectric point, hydrophathy and aliphatic index as features for SVM classifiers to predict protein function. King et al. [27] employ not only sequence similarity, but also bio-chemical attributes such as molecular weight, and percentage amino acid content. Pavlidis et al. [28, 29] predict gene function from heterogeneous data sets derived from DNA microarray hybridization experiments and phylogenetic profiles.

The availability of protein data sets annotated with GO terms and InterPro domains provides an opportunity to study the extent to which InterPro can be used to predict GO terms. The InterPro database contains over 12,000 entries and the GO contains over 19,000, but proteins are usually annotated with a few terms from each database, resulting in a sparse data set. In addition, a large set of proteins will contain only a few positive examples of each GO term, leading to extremely biased class distribution in which less than 1% of the training instances represent positive examples of a GO term.

Many studies have shown that standard classification algorithms perform poorly with imbalanced class distribution [30, 31, 32]. The most common method to overcome this problem is through re-sampling of the data to form a balanced data set.

Re-sampling methods may under-sample the majority class, over-sample the minority class, or use a combination of both approaches. A potential drawback of under-sampling is that effective instances can be ignored. Over-sampling, however, is not without its problems. The most common approach is to duplicate instances from the minority class, but Ling et al. [33] show that this approach often does not offer significant improvements in performance of the classifier, as compared to the imbalanced data set. The other approach is the Synthetic Minority Over-sampling Technique (SMOTE) [34], which is an over-sampling technique with replacement in which new synthetic instances are created, rather than simply duplicating existing instances. Under-sampling can potentially be used to avoid the problems of over-sampling [31, 35]. Under-sampling removes instances from the majority class to create a smaller, balanced data set. While other approaches such as feature weighting can be employed, under-sampling has the added benefit of reducing the number of training instances that are required for training, thus reducing the difficulties of training pattern recognition algorithms on very large data sets.

In this chapter, I consider the application of statistical pattern recognition techniques to classify proteins with GO terms, using InterPro terms as the feature set. I show that many of the problems associated with sparse and imbalanced data sets can be overcome with standard feature and instance selection methods. Feature selection in an extremely sparse feature space can produce instances that lack any positive features, leading to a subset of identical instances in the majority class. By selectively removing these duplicated instances, or keeping them, I trained two SVMs that have different performance characteristics. I describe a meta-learning scheme that combines both models, resulting in a more improved performance than can be obtained by using SVM alone.

Table I. Examples of randomly selected classes (GO terms) and features (InterPro terms) illustrating the imbalanced and sparse nature of the data set.

GO term	Number of Positive Examples	Number of Negative Examples
GO:0000001	22	4568
GO:0000022	15	4575
GO:0000776	12	4578
GO:0005635	35	4555

GO term	Number of Positive Examples	Number of Negative Examples
IPR000002	5	4585
IPR000009	2	4588
IPR000073	13	4577
IPR000120	2	4588

B. Proposed Method

1. Data Set

The data set used for this study was comprised of 4590 annotated proteins from the *Saccharomyces cerevisiae* (Yeast) genome obtained from the UniProt database [36]. This protein contains manually curated GO annotations as well as InterPro terms automatically assigned with the InterPro Scan [37].

The data set contains 2602 InterPro terms and 2714 GO terms with an average of 2.06 InterPro terms and 3.99 GO terms assigned to each protein. Table 1 illustrates

the imbalanced nature of the data set. In this study, each GO term was considered as an independent binary classification problem and therefore, all proteins annotated with a GO term are treated as positive instances (GO+) and the remaining proteins are treated as negative instances (GO-), resulting in a highly biased class and feature distribution. For the purpose of this study, I only considered data sets that contained at least 10 GO+ proteins. Therefore, proteins annotated only with GO terms that did not meet this criterion were removed from the data set, resulting in a reduction of the size of the data set to 4347 proteins.

2. Under-Sampling

Several methods are available for creating balanced data sets. If the features are continuous, I can perform over-sampling using methods such as SMOTE [34] which create a new interpolated value for each new instance. In this case, however, the data set is binary format so this method cannot be used. Chawla et al. [34] described that under-sampling is slightly better performance in terms of costs and class distribution. Another issue about the under-sampling is how the ratio positive versus negative to make a balanced set is optimized for training. On the point of dealing with the imbalanced data problem, Al-shahib et el. [26] applied various under-sampling rates from 0% to 100% and conclude that the fully balanced set which has the same number of positives and negatives, gives the best performance. In light of this prior work, I performed under-sampling to create fully balanced data sets for each GO term.

For each data set, I performed under-sampling of the majority class (GO-negative proteins) to create a balanced data set for SVM induction. I compared the performance of four under-sampling methods: Farthest, Nearest, Cluster and Random. In the first two cases, I used Euclidean distance, computed on the basis of the InterPro terms content of each protein as a measure of distance. The Farthest and Near-

est methods select proteins from the negative class that have the greatest and least distance from the positive class, respectively. The Cluster method first performs hierarchical clustering of the negative class where the number of clusters formed equals the number of instances in the positive class. A single protein from each cluster is selected randomly. The Random method randomly selects proteins from the negative class.

Let D_{All} be the set of all of IPR and GO data. I define the example of the data set as $D_{All} = \{(X_i, Y_j) | i, j=1, \dots, k\}$, where k is the number of proteins, and $X=(x_1, x_2, \dots, x_l) \in \text{IPR}\{0,1\}$ are feature vectors, and l is the number of InterPro (IPR) features in the data set. $Y=(y_1, y_2, \dots, y_m) \in \text{GO}=\{0,1\}$ is the class designation (GO terms) and m is the number of GO terms in the data set.

3. Feature Selection

I employed four different objective functions for feature selection: chi-squared (χ^2), information gain, symmetrical uncertainty, and the correlation coefficient. Classical linear correlation [38] measures the degree of correlation between features and classes, and ranges from -1 to 1. If the features and the class are totally independent, then the correlation coefficient is 0. The traditional linear correlation method is very simple to calculate, but it assumes that there is a linear relationship between the class and the feature, which is not always true [38]. To overcome this shortcoming, the other correlation measures based on the theoretical concept of entropy were also assessed for feature selection. Information gain is a measurement based on entropy, and measures the number of bits of information obtained for class prediction [39]. However, information gain has a non-normalized value and it is biased toward of features with more value. To compensate for this disadvantage, the symmetrical uncertainty value is normalized from 0 to 1 and un-biased in terms of feature content. The idea of

symmetrical uncertainty is based on the information gain, but the applied value is normalized and un-biased toward features with more value [38]. When calculating the contingency between features and a class of interest, the χ^2 statistic measures the lack of independence. As the χ^2 statistic values increases, the dependency between features and classes also increases [39, 38].

The features were ranked using each of the objective values, and a sequential forward selection search algorithm was used for feature selection. Forward selection was used since it is considered to be computationally more efficient than backward elimination [40, 41]. The feature inclusion threshold for 12 randomly selected data sets was determined by computing the error rate during each stepwise feature addition and finding the minimal error rate. The average threshold value for the 12 data sets was used for the remaining data sets. Chi-square(χ^2) was used to calculate the contingency between features and a class of interest, and the χ^2 statistic measures the lack of independence. As the χ^2 statistic values is larger, the dependency between features and classes is also high [39, 38].

4. Filtering Step

The filtering step is for removing the unrelated prediction in order to increase the accuracy. Table II is an example of a prediction and original set from the Uniprot in terms one InterPro term. For example, originally, an InterPro term is related to three GO terms (GO:000298, GO:000319 and GO:001295) which can be referred to as Uniprot data, but the annotated terms are GO:000298, GO:005515, GO:001295 and GO:003948. Among this prediction, two terms are related to the InterPro term but the other two are not. This means that GO:005515 and GO:003948 are considered as False Positives. Hence, those terms are filtered out in the prediction.

Table II. Filtering step by removing false positives.

GO term	Uniprot	Prediction	Remark
GO:000298	✓	✓	TP
GO:005515		✓	FP
GO:000319	✓		FN
GO:001295	✓	✓	TP
GO:003948		✓	FP

C. Experiments

Individual data sets are constructed for each GO term which are then subjected to feature selection and instance selection prior to SVM model induction. Among the possible statistical pattern recognition algorithms, I employed SVM because of its overall better performance compared to others such as the C4.5 decision tree and Naïve bayes in the preliminary experiments (data not shown). Because of the extremely sparse nature of the data set, the feature selection step can remove all InterPro terms from some proteins, resulting in proteins that completely lack features. In most cases, feature selection resulted in a large number of GO- proteins in each data set. I theorized that such a large number of redundant proteins in the data sets could lead to skewed performance of the SVM, so for each GO term, I constructed two data sets. *Model 1* refers to the SVM learned from the data set containing the redundant GO- proteins and *Model 2* refers to a smaller set in which redundant proteins were removed prior to model induction (Fig. 2). For each GOs, I reduce the InterPro feature set with chi-squared (χ^2) methods and make a balanced data set by combining with the farthest negative protein from the positive instances and positive

instances. Two procedures are independent, so both models are based on the InterPro feature selection, but *Model 1* is trained by a balanced set from the whole set and *Model 2* is trained by a balanced set from the non-zero data set. The darkest gray GO- parts are considered as negative GOs and white GO-s are predicted as true by *Model 1*. However, in the *Model 2*, which applied under-sampling with non-zero data and feature selection, GO- annotate as negative. I expected that *Model 2* would result in SVM with higher accuracy than only *Model 1*.

1. Feature and Instance Selection

I randomly selected 50 *Model 1* data sets and compared the performance of the feature selection and instance selection methods. The relative performance of the various methods was compared using error rate and AUC by 10-fold cross validation. The chi-squared method outperformed the other feature selection methods (Table III) and was used to prepare data sets for instance selection. The Farthest method provided the best instance selection performance (Table IV) and was selected to create balanced data sets for SVM induction.

I used 10-fold cross validation to compare the performance of SVMs trained using *Model 1* and *Model 2*. In general, *Model 1* SVMs had very low false negative rates but had high false positive rates whereas *Model 2* SVMs tended to have lower false positive rates (Fig. 3). On average, *Model 1* has 0.32 false negative instances per SVM but 297.54 false positive instances and 4024 true negative instances per SVM among 4347 proteins. Of the 374 SVMs trained, 84% have less than 1 false negative instance using *Model 1* (Fig. 3(a)). Therefore, I conclude that this model is effective at classifying positive instances, although it should be noted that *Model 1* trained SVMs have high false positive rates. Since properties of both models were desirable for the classifier, I developed a meta-learning scheme that incorporated both models

Table III. Performance comparison of 4 different feature selection methods (SU: symmetrical uncertainty, INFO: information gain, CHI: chi-squared, ABS: absolute correlation coefficient).

Method	Sensitivity	Specificity	AUC	Error Rate
SU	0.98	0.83	0.73	0.01
CHI	0.99	0.93	0.87	0.01
INFO	0.78	0.98	0.85	0.12
ABS	0.77	0.25	0.79	0.43

Table IV. Performance comparison of 4 different under-sampling methods.

Method	Sensitivity	Specificity	AUC	Error Rate
Farthest	0.94	0.94	0.78	0.03
Nearest	0.73	0.79	0.74	0.52
Cluster	0.90	0.90	0.77	0.09
Random	0.87	0.93	0.77	0.08

and includes a final filtering step, in order to reduce the false positive rate.

2. Test Procedure

Data flow for the prediction step is shown in Fig. 4. The dotted line in Fig. 4 represents use of *Model 1* only (Process A). The solid line represents use of both *Model 1* and *Model 2* (Process B). The filtering step is used in both cases. I focus on keeping the true positive rate as high as possible so *Model 1* is utilized as the first step. The *Model 1* classifier plays a role in excluding the most negative instances, but has the risk of making false positive classifications. Proteins classified as positive by *Model 1* are classified again using *Model 2*, thereby reducing the number of false positive proteins from 297.54 to 110.63 on average.

The third step is comprised of a decision rule that I devised based on observations I made of the data sets. Under the assumption that a positive relationship exists between GO terms and InterPro terms, I define the following decision rule: For each GO term assigned to a protein, I identify whether a training proteins exists with that GO term and an InterPro term assigned to the predicted protein. If at least one association exists, then the predicted GO term is retained, otherwise it is removed from the set of predicted GO terms.

I compared the precision of the suggested classification procedure (Fig. 5 Process B) with the precision of *Model 1* alone (Process A), where precision is measured as the number of true positive GO terms divided by the number of predicted GO terms. I randomly selected and held out 435 proteins from the training data set to use for comparative analysis of the two classification procedures. The average precision of Process A, which applies only to *Model 1*, is 0.3411, while Process B which applies both SVM model, is 0.3523.

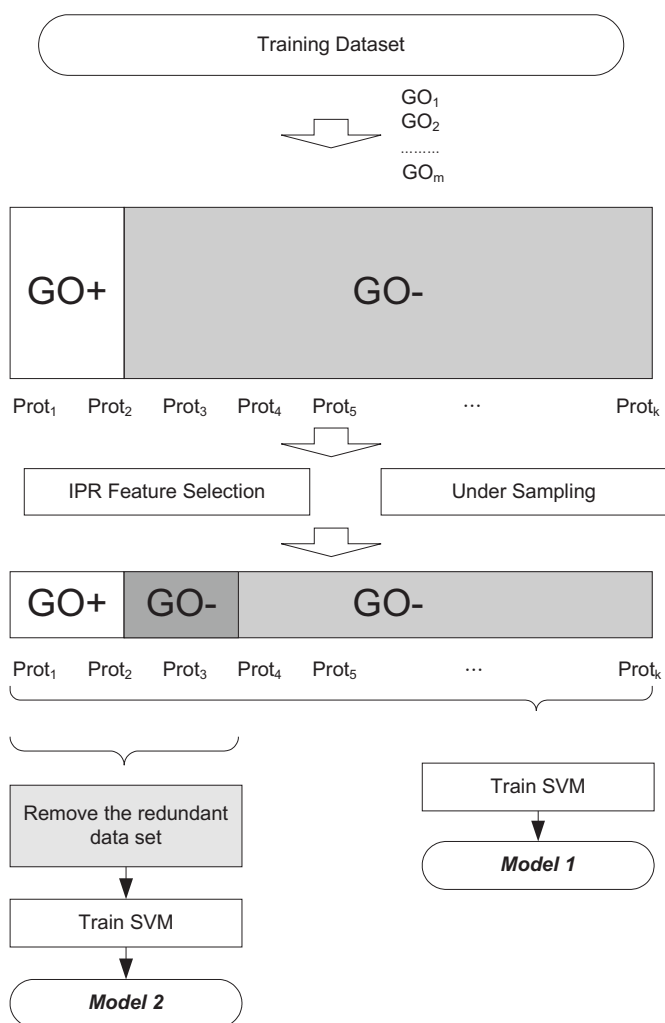


Fig. 2. Flow chart for the training process.

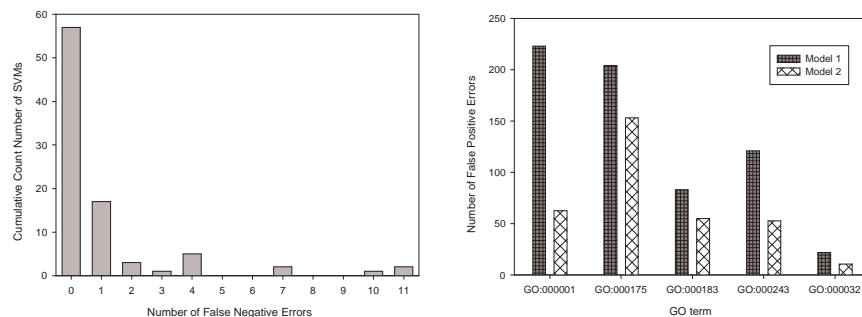


Fig. 3. Performance of SVMs estimated by 10-fold cross validation. (a) Cumulative count of number of false negative errors produced by 374 independent SVMs. (b) False positive errors produced by *Model 1* and *Model 2* for five randomly selected GO term data sets.

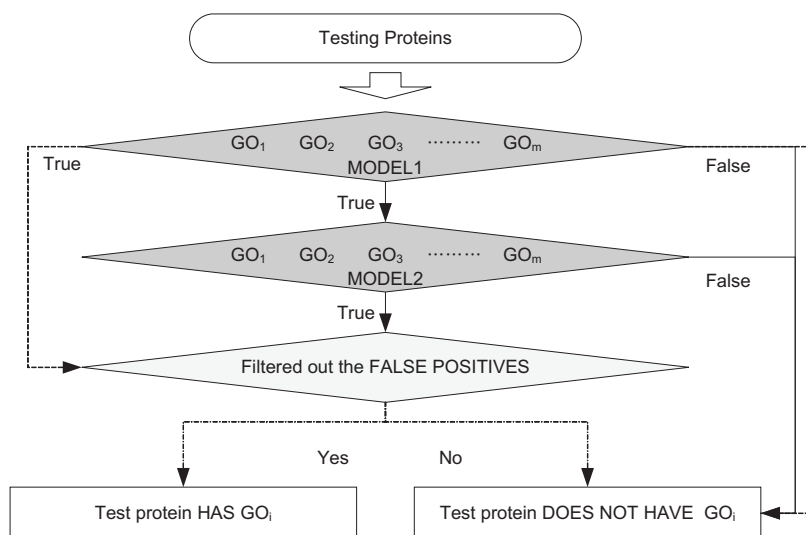


Fig. 4. Flow chart for the testing process.

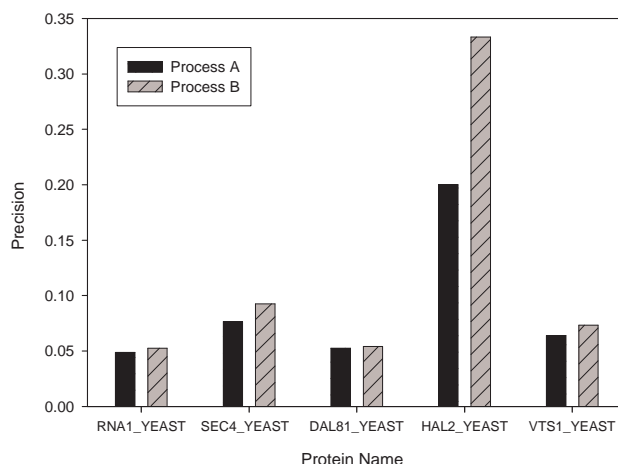


Fig. 5. Precision of the two processes.

3. Comparison to Other Methods

Using the training set, I prepared SVMs for each GO term. Precision is employed again as a metric to compare the performance of the method to that of other described methods. Most automated GO annotation methods are only available as web-based forms designed to process one protein at a time. Therefore, I randomly selected nine proteins from the hold out set to use for comparison to other published annotation methods. One exception to this is IPR2GO, which is a manually curated mapping of InterPro terms to GO terms that is maintained by the InterPro consortium. I implemented a classifier using the IPR2GO rules and estimated performance using 10-fold cross validation. The method, Automatic Annotation of Protein Functional Class (AAPFC), includes trained SVMs for every GO term in which ten or more positive protein instances could be found in the training data set. In this comparison, AAPFC had a precision of 0.3241 while that of IPR2GO was 0.1777. Additionally, the precision of three other GO annotation tools was, in most cases, considerably

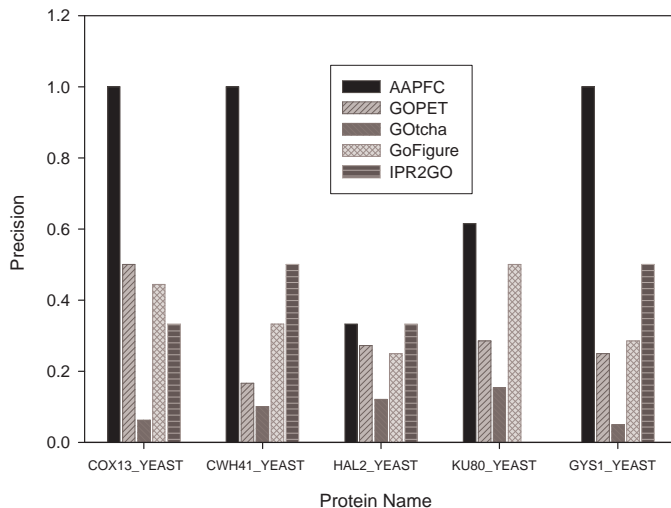


Fig. 6. Comparison of the precision of the proposed classification method (AAPFC) to four other methods.

lower than AAPFC (Fig. 6), where proteins which were not included in the known training set in the *S.cerevisiae* are randomly selected. I used the author recommended confidence thresholds of 20% and 50% for the GOTcha and GOPET methods, respectively, and employed an e-value cutoff of $e-5$ for GoFigure. On average, precision is 0.53 for AAPFC, 0.17 for GOPET, 0.05 for GOTcha, 0.29 in GoFigure, and 0.20 in IPR2GO. Surprisingly, the AAPFC outperformed IPR2GO, suggesting that there are many protein functions that can be predicted from InterPro terms, that cannot be described as a simple one-to-one translation table.

D. Taxon-Specific Training Model

The classifier described above employs the SVM algorithm to assign GO terms to proteins annotated with InterPro terms [42]. A distinguished property from other previously described approaches is that the training data sets are a taxon-specific.

Using several standard performance metrics, the trained classifier has increased accuracy compared to the InterPro2GO which is a mapping of InterPro terms to GO terms and is maintained by expert curators. Furthermore, the provided taxon-specific model has a higher F-measure value in annotation than the model trained with a larger, non taxon-specific data set. The non taxon-specific model is a general model whose training data are constituted of various organisms including fungi, bacteria, plants and so on. One possible explanation for the apparent improved performance is that it is an artifact, resulting from over-fitting as a result of using a small training data set. To determine whether the improved performance is due to over-fitting, I performed the following experiment. Classifiers are trained with several taxon-specific data sets and the performance is compared with cross-validation. Next, small data sets with sizes equivalent to the taxon-specific sets were prepared with randomly selected proteins.

1. Data Sets

Three training data sets, “Uniprot”, “Fungi”, “Fungi-expanded”, are made in order to show that the taxon-specific classifier outperforms the general, non taxon-specific set. All sets consist of annotated protein sequences from the UniProt database. I removed proteins that lacked GO annotations and removed all GO terms from the remaining proteins that were annotated with the evidence code IEA, which represents “Inferred from Electronic Annotation”. Each of the three training data sets are comprised of GO terms with positive examples, i.e., proteins that are annotated with the GO term, and negative examples, i.e., all proteins that are not annotated with the term. The “UniProt” data set is comprised of all GO terms that are annotated to at least 10 proteins regardless of the species to which the protein is assigned. The “Fungi” data set includes GO terms that were assigned to proteins from species in the kingdom Fungi that had at least 10 protein annotations from members of

Table V. A summary of the data sets for taxon-specific models.

Data Set	Proteins	InterPro terms	GO terms
Fungi	7093	3331	459
Fungi-expanded	70205	5438	1390
UniProt	119016	8414	2826
Bacteria	3282	2255	115
Plant	5669	2087	285
Vertebrata	16079	1706	1232

the Fungi. The “Fungi-expanded” data set is similar to the “Fungi” data set but includes additional positive examples of each GO term drawn from the whole UniProt database. This enabled us to include considerably more GO terms and training instances than the “Fungi” training set (Table V). Among these data sets, “Fungi” is the smallest. The “Uniprot” set is composed of 119016 proteins and 2826 GO terms, which is approximately 17 times larger than “Fungi” in proteins and 7 times larger in GO terms. In a similar manner, I prepared three more taxon-specific data sets representing “Bacteria”, “Plant”, and “Vertebrata”.

Similar to the Uniprot data set, the taxon-specific data are also made up of at least 10 protein annotations for each taxon. The last three data sets in Table V describes the provided data set. The number of proteins, InterPro terms and GO terms in “Bacteria” and “Plant” are less than in the fungal set, unlike the “Vertebrata” data set. These six data sets are used to compare the performance in both the shared GO terms and the whole data set.

2. Comparison of Taxon-specific Models

In this chapter, three different sizes of taxon-specific sets are provided. One is "Fungi", another is "UniProt" and the other is "Fungi-expanded". "Fungi-expanded" which is similar to the "Fungi" data set but includes additional positive examples of each GO term drawn from the whole UniProt database. These three sets are also used to learn classifiers using AAPFC method. Briefly, each GO term is treated as an independent classification problem, and the SVM classifier is trained for each term. The data sets are highly imbalanced, containing an overabundance of negative examples of the GO term (all proteins that are not annotated with the GO term) as compared to the positive examples, so they are treated by feature selection and instance selection. While the purpose of AAPFC is that protein functional annotation is performed by the protein domain, the aim of models with "Fungi-expanded" and "UniProt" is reported to the outperformed modeling in taxon-specific data sets.

As a performance comparison, I employ sensitivity, precision and F-measure which are performed by the 10-fold cross validation. Sensitivity is the proportion of GO term annotations in the training data set that were correctly predicted by the classifier, and specificity is the proportion of GO term annotations not found in the training data set that were correctly not assigned to the protein by the classifier. Precision is the proportion of GO terms in the training set that proteins originally have. Sensitivity and precision were also combined into a single metric called the F-measure which is calculated. To compare the performance of the data sets, I computed the mean of each performance metric over all fungal proteins in each data set (Table VI). Similarly, I measured the performance of the Interpro2GO mapping and included the results in Table VI. Each of the training data sets resulted in models that outperformed the Interpro2GO mapping. In the case of the UniProt data set, only a modest

Table VI. Performance matrices of various data sets for fungi.

Classifier	Sensitivity	Specificity	F-measure
Interpro2GO	0.0262	0.1175	0.0280
Fungi	0.2680	0.9745	0.3101
UniProt	0.1149	0.9976	0.0893
Fungi-expanded	0.2304	0.9875	0.2611

overall improvement in performance was observed compared to Interpro2GO, while the highest performance was found with the fungi data set. All performance metrics in training with the fungi data set shown an improvement over the UniProt data set.

However, the fungi classifier has approximately one-seventh the number of GO terms compared with Uniprot. Adding additional non-fungal proteins to the fungal specific GO terms (the fungi-expanded data set) creates more than three times the number of GO terms that could be trained with only a slight reduction in performance. From these experiments, the number of GO terms and the number of proteins are one of the factors used to decide on the performance, that is, a small training set can give the best results. That is a doubtful question in this section. Generally speaking, the small training sets are exposed to a higher danger of being over-fitted, which is one of the supervised learning issues. When the data set is small, which can be biased, so it is accustomed to the limited data set. The first checking point against an over-fitted issue is that comparisons of the performance in shared GO terms with various taxon-specific training data sizes. As regards taxon-specific data, I employ three different data - “Bacteria”, “Plant” and “Vertebrata”, whose data sizes are varied (Table V). The procedure of the training and the testing is the same as in AAPFC.

One important point in terms of each GO classifier is that the training data is composed of different sizes of sets, independently. In other words, each model makes a new balanced training set regardless of the whole taxon data size. The balanced sets are composed of all proteins annotated with GO terms and the same size of un-annotated proteins selected from the remaining proteins. This implies that the trained models are affected not by the whole training size but by the number of positive GO term sizes. Hence, one of final outcomes as proof against the over-fitted issue, is finding any correlation between the positive numbers and performances. If there is any relations between the performance and training data set, I induced that the performance in each model depends on the trained data size and that the model has a high chance of being over-fitted. However, no correlation exists between the number of proteins included in the training set and its performance (Table VII). For example, in Table VII, F-measure values for GO:0003677 are 0.2423, 0.6798, 0.2137 and 0.1905, respectively, for the bacteria, plant, fungi and vertebrata data sets, while the number positive proteins are 24, 430, 64, and 538 in each data set. If over-fitting has occurred, the data set which has the smallest number of positive instances should have a higher performance [43]. However, the second largest set has the best F-measure value. Another example is GO:0006508. The “Plant” taxon-specific model has the best output, but it has the largest training sets among the four taxon-specific models. This result suggests that the number of training proteins does not have much influence on the accuracy of the model.

Nevertheless, the number of positive training proteins and the performance does not have any relationship, the whole data set size has also been influenced by the performance, since the performance is measured by the 10-fold cross validation. In other words, the tested proteins are different in each validation. An average of tested proteins is 709 in fungi, and 1607 proteins in the vertebrata. In addition, the selected

Table VII. Performance comparison of four different taxon-specific models.

GO terms	Bacteria		Plant		Fungi		Vertebrata	
	Positive	F-measure	Positive	F-measure	Positive	F-measure	Positive	F-measure
	GO:0003677	24	0.2423	430	0.6798	64	0.2137	538
GO:0003723	11	0.7841	46	0.5861	77	0.3268	286	0.2176
GO:0003735	313	0.9421	264	0.9192	146	0.6521	121	0.1619
GO:0003743	16	0.3327	30	0.2720	26	0.1150	41	0.1268
GO:0003899	27	0.8287	16	0.8077	35	0.3111	20	0.1400
GO:0005524	49	0.4404	73	0.3273	24	0.1139	430	0.3192
GO:0005525	22	0.5140	43	0.7875	10	0.1349	124	0.3459
GO:0005576	11	0.6327	19	0.1760	33	0.2639	670	0.2355
GO:0005737	145	0.7581	162	0.2111	925	0.6129	1564	0.2933
GO:0006118	20	0.4047	88	0.1948	11	0.2952	101	0.1546
GO:0006281	48	0.6899	20	0.3993	60	0.2611	88	0.1526
GO:0006350	29	0.7620	12	0.3793	20	0.1284	25	0.0741
GO:0006355	36	0.4691	312	0.5317	22	0.0762	416	0.2544
GO:0006412	283	0.8635	257	0.8900	177	0.7963	155	0.1967
GO:0006413	16	0.3494	20	0.2931	32	0.1356	27	0.2163
GO:0006414	23	0.4656	18	0.6339	10	0.0455	11	0.0368
GO:0006457	38	0.7772	39	0.6495	50	0.2541	92	0.2128
GO:0006508	36	0.3186	112	0.3274	23	0.1489	250	0.3178
GO:0006950	29	0.6211	29	0.2825	68	0.2820	117	0.1064

number of GO classifiers is also different. Bacteria only has 115 GO terms, but vertebrata has 1232 GO terms. Consequently, this large number of test proteins make a low sensitivity or a low precision, thus I judge the performance again on the same training size.

The proposed approach is that Uniprot data sets which have any taxonomic group are reduced to the same size as the small taxon-specific sets. Among six used data sets in Table V, the fungi set and UniProt set can have many shared GO term, hence these two sets are trained for this experiment. The fungi set that has 459 GO terms is used by itself. UniProt sets are reduced in size by random selection, where the whole size is decided as the same size for fungi, resulting in it being composed of 411 GO terms. If the performance in the fungi set is still higher than the sample UniProt sets, the over-fitted issue in the small training sets has happened.

Table VIII is a result of 12 randomly selected classifiers from 97 GO terms which are shared with both fungi and the reduced UniProt. For example, GO:0000723 has better performance in fungi, resulting in 0.5390 and 0.5756 for the sensitivity and F-measure respectively, while the same GO term from the reduced UniProt data set has 0.021 and 0.3046 for the same measures. GO:0006468 also has a high performance in fungi. However, other GO terms have higher value in the reduced UniProt. The overall average of the shared 97 GO terms in sensitivity, specificity and F-measure is 0.1992, 0.9744, 0.2849 in the fungi set and 0.3176, 0.9961, 0.4135 in the sampled UniProt set. Overall, the taxon-specific models outperform the reduced set from the Uniprot. The overall conclusion is that taxon-specific data enables us to give more meaningful information to build an annotation model than the general case. Moreover, the models are not over-fitted.

Table VIII. Performance comparison of the random selected shared GO terms between fungi set and sampled UniProt.

GO term	Fungi				Sampled-UniProt			
	Positive	Sensitivity	Specificity	F-measure	Positive	Sensitivity	Specificity	F-measure
GO:0000723	204	0.5390	0.7787	0.5756	13	0.2021	0.9966	0.3046
GO:0005524	24	0.0619	0.7999	0.1139	1074	0.9122	0.9978	0.9526
GO:0006118	11	0.1935	0.9999	0.2952	115	0.7253	0.9978	0.8340
GO:0006270	34	0.1528	0.9862	0.2576	15	0.5417	0.9986	0.6154
GO:0006364	40	0.1143	0.9994	0.1998	68	0.5542	0.9986	0.7020
GO:0006461	26	0.1243	0.9993	0.2161	10	0.0662	0.9984	0.1188
GO:0006468	103	0.3743	0.8994	0.5231	37	0.2813	0.9976	0.4232
GO:0008565	29	0.2683	0.9999	0.4005	12	0.6833	0.9969	0.7156
GO:0009019	10	0.6750	0.9999	0.6857	25	0.7333	0.9962	0.7969
GO:0016021	33	0.1266	0.9997	0.2187	74	0.3183	0.9985	0.4788
GO:0046982	36	0.0875	0.9991	0.1573	26	0.1079	0.9967	0.1880
GO:0051082	50	0.2595	0.9997	0.3974	119	0.6821	0.9964	0.8075
:	:	:	:	:	:	:	:	:
average	0.1992	0.9744	0.2849	0.3176	0.9961	0.4135		

E. Summary

In this chapter, I propose a method for assigning GO terms to proteins using InterPro terms as features, and a learning independent SVM for each GO term. By creating two data sets, each having different properties, and learning two SVMs for each GO term, I developed a meta-learning scheme that benefits from the strengths of each model. Moreover, I investigate a taxon-specific model which is composed of proteins from species in a specific kingdom. This taxon-specific model outperformed the non taxon-specific one. In this experiment, one possible issue is the over-fitted problem in the taxon-specific model, since a small set is highly exposed to the over-fitted issue. By the various taxon-specific sizes including Bacteria, Plant and Vertebrata, I present that the proposed model is not over-fitted. Moreover, I tested on the same size of proteins by a randomly selected sample from the Uniprot set, because the number of tested proteins in various species is different.

My current strategy treats each GO term as an independent learning problem. This has some practical benefits in that individual classifiers or sets of classifiers could be learned or re-learned over time without the need to re-learn the whole system. On the other hand, this approach assumes that all GO terms are independent. Since the GO terms are organized in a DAG, a dependence among some terms is assumed. Therefore, in the chapter IV, I propose to utilize the multi-layered method as an approach to capture dependence among GO terms into the learning method. The outputs of the classifiers described here can be used as inputs to another classifier, thus enabling the dependence among GO terms to be utilized for classification. Moreover, this approach employs only one feature set. Nevertheless, the InterPro terms include much protein domain information, and more feature sets provide more robust gene annotation, because gene function does not define only one factor. Thus, in the next

chapter, the system with multi-feature sets is carried out, where the treated features are biochemical properties (amino acid content, etc.), phylogenetic profiles, sequence similarity, and others.

CHAPTER III

PREDICTION OF GO (POGO): A MULTI-FEATURE LEARNING SCHEME
FOR ASSIGNING GENE ONTOLOGY TERMS TO PROTEINS

A. Related Work

The rapid increase in the number of genome sequencing projects has led to a deluge of genomic data that are available to biologists. Automated methods are being extensively applied to rapidly annotate gene models in newly sequenced genomes and to a lesser extent manual methods are also being used. While equally as important as gene structural annotation, functional annotation of proteins is usually limited to an automatic processing with sequence similarity searching tools. As the number of genome sequencing projects continues to expand, it is becoming clear that gene annotation is becoming a central issue for biologists. The computationally obtained features of the proteins and the genes and many statistical models enable us to predict the function automatically. Gene Ontology terms (GO) especially play an important role, by providing text descriptions and controlled vocabularies of the gene function that can be applied uniformly to proteins from a wide variety of species.

One of the most popular approaches for functional annotation is to assign textual descriptions or terms from controlled vocabularies from proteins that have been identified with sequence similarity search tools such as BLAST. Similar approaches rely on protein motif and protein family databases such as PFAM and InterPro. Proteins are matched to database entries using techniques that employ profile sequence alignments, profile hidden markov models, and related methods, and then manually curated mappings between the database entries and controlled vocabularies are used to assign functional terms to the proteins. One such example is the Interpro2GO

mapping which can be used to assign GO terms to proteins that have been matched to entries in the InterPro database. The GOA project relies on several such mappings to provide automated GO annotations for users [2, 3].

Many studies rely on the identification of similar proteins in large databases of annotated proteins [7, 8, 9], but this approach assumes that homologous proteins have similar functions. Others employ various attributes [27, 24, 25] to overcome this issue. GOPET [24, 25] assigns terms from the biological process and molecular function ontologies by training a Support Vector Machines (SVM) classifier. The GO level, a path, a BLAST output, GO frequency by considering relationships, GO frequency related attributes and the annotation quality related to these attributes are employed as features, and they are normalized and are used to train the SVM. King et al. [27] apply the decision tree and Bayesian network for pattern recognition by employing not only sequence similarity, but also bio-chemical attributes such as molecular weight and percentage amino acid content. Many researches [44, 9, 28, 24, 25] are focused on the SVM instead of the decision tree, because the SVM performed better than any other traditional methods [45], hence I suggested an application that trained the SVM [42].

However, the main property of our previous work, as well as those of others, is that the classifiers utilize only one type of feature such as BLAST results or InterPro terms [7, 8, 9, 24, 25]. Feature types such as BLAST and InterPro terms provide most of the information from which protein function can be predicted. However, if protein functions are deduced from more features, the automated annotations should be more accurate. In this chapter, I describe a method which includes various features including not only sequence similarity, but also protein structure information, protein domains and biochemical properties. One issue is how to combine these heterogeneous feature sets. The value of integration of various data has been illustrated

by several studies [17, 22, 14, 15, 28, 29, 16, 21]. The traditional method is combining networks of a functional linkage [14, 15, 16]. The basic assumption is that functionally associated proteins have a common structural complex, metabolic pathway, biological process or closely related physiological function, thus they predict using phylogenetic profiles correlated to mRNA expression or domain fusion [14, 15]. Pellegrini et al. [16] also shows that two genes are functionally linked if they have similar phylogenetic profiles. Even though these approaches are very easy to understand and can be easily generalized, the approaches usually rely on very simplistic functional inferences, and are semi-automatic, and heuristic methods. Expanding on phylogenetic profiles, Pavlidis et al. [28, 29] suggest a method to predict gene function from heterogeneous data by training a kernel SVM with features derived from DNA microarray hybridization experiments and phylogenetic profiles. Troyanskaya et al. [21] design a flexible system for adding new input data, including protein-protein interaction data and transcription factor binding site data using a Bayesian framework. The employed method for the combination of different types of data is the early integration method. Barutcuoglu et al. [17] propose a two-step procedure for the functional annotation by the SVM independent classifier with various data types and integrate in its classifier the hierarchical relationship of the protein functions. Most integration systems with heterogeneous data are implemented by simply merging of the different data types by changing the format, including probability or projected value from the kernel function regardless of the properties of each data type. However, Deng et al. [20] consider this as a drawback, by putting different weights in each data set. Hence, they want to find the most effective feature set by the combination of feature sets. The weights are determined by the optimized classification accuracy. All the above approaches build only one layered classifier for the purpose of merging the various data types. In this chapter, I propose a multi-layered classifier for robust gene function annotation.

The Combiner [46] and Stacked generalization [47] are examples of meta-learning classifiers [48, 49, 50, 46, 47]. The biggest advantage of meta-method is improving the accuracy by combining multiple outputs. The basic idea of a meta-learning classifier is a construction of the classifier by taking a weighted or unweighed voting prediction with the meta-data set. In addition, the meta-classifier is able to integrate the base-classifier into one global classification, and where the products form the base-classifier, this is called the meta-data. The meta-data is usually composed of one of the classifications or some value extracted from the base-classifier, thus it serves as a feature vector to a meta-classifier. Base-classifiers use the original data as inputs while the meta-classifier uses the outputs of the base-classifiers as input. The Combiner [51] is similar to the Stacked generalization [47], which improves prediction accuracy by merging at least two base learners results. These learners are made up of at least two layers, the first layer is a training model with original data and the second layer is combined results from the output of the first layered classifier. The Stacked generalization makes a base-classifier by the one leave out method, and accomplishes “n” iterations, where “n” is the total data set size. The combiner divides two cross validations, and each validation is processed in the same way as the stacked generalization. Chan et al. [51, 48, 46] compare the performance of these two methods and show that the two methods carry a similar accuracy but there is better efficiency in the Combiner, because of the shorter execution time. The base-classifiers in the original Combiner and Stacked generalization methods are comprised of different types of classifiers, using the same data set as input. However, in this study, the base-classifiers are created using the different feature sets such as InterPro terms, sequence similarity through BLAST, bio-chemical properties and the product of the HHpred [52, 53] instead of different classifiers.

In this chapter, I describe a multi-layered classifier which comprises a base-

classifier and a meta-classifier [51] using the Combiner method. A base-classifier is a single classifier with each feature, and a meta-classifier combines these feature sets. In order to distinguish the original meta-learning classifier, which usually uses same feature domain but different learners in the second layer, I use the term multi-layered classifier. During this process, I show that the performance of a multi-layered model using four heterogeneous feature sets outperforms a single classifier which is trained by only one feature. The method for combining four different data sets is described in the method section.

B. Proposed Method

1. Data Set

AAPFC is a single learner using SVM with the protein domain as the feature set, thus the system requires that each protein has at least one InterPro term. I present a new system which adds more features including not only InterPro terms but also other feature sources. I also allow proteins that lack one or more of the feature sets, enabling many more proteins to be classified. Moreover, I employ a multi-layered classifier, to combine the heterogeneous base-classifiers. The feature sets are InterPro, bio-chemical properties, BLAST and protein structure information.

- InterPro term

InterPro terms [10] are defined in the InterPro database which is a curated protein domain database, that acts as a central reference for several protein family and functional domain databases, including Prosite, Prints, Pfam, Prodom, SMART, TIGRFams and PIR SuperFamily. InterPro terms are assigned to proteins using the InterPro Scan application. The InterPro terms comprise the terms assigned to annotated proteins from the UniProt database and includes

3339 terms.

- BLAST

The second feature set is comprised of BLAST hits derived from a BLAST search of the UniProt database of GO annotated proteins (excluding machine annotated terms) [23]. BLAST is a tool to compare the query sequence to a database of proteins and gives the *e-value* according to query options. As the *e-value* approaches 0, the query sequence and a matched sequence from the database are more similar. Several previous methods including GoFigure [8], GOblet [7], and OntoBlast [6] use sequence similarity based on BLAST results as features. GOAnno [54] is also the extension of the similarity based annotation using hierarchically structured functional categories, and similarity based functional annotation is considered as a central feature. GO terms derived from the BLAST hits (*e-value* cut-off $e-10$) are used as features. I cumulatively score each GO term and use the number as a feature value. The resulting BLAST data set contains 3182 features, and like the InterPro terms, this set is also a very sparse and imbalanced set, but the data format is an integer instead of being binary.

- Bio-chemical information

Bio-chemical properties of the proteins are used as another feature set, since other authors [26] have previously shown that they are useful for functional classification. The features I use in this study include amino acid content, molecular weight, proportion of negatively and positively charged residues, hydrophobicity, isoelectric point and amino acid pair ratio. The pepstat program in EMBOSS [55] generates protein properties based on the amino acid sequence.

- Protein structure information

The fourth feature set is protein structure as computed using the HHpred program [52] which is used for protein homology detection and structure prediction using a sequence database search with the BLAST [23] or PSI-BLAST [56] programs. First of all, they implement a pairwise comparison of the profile hidden Markov model and search the database. The databases PDB, SCOP, pFam, SMART, and SCOP [57] are employed as the target databases. The main goal of SCOP is to organize the available structures and define the evolutionary relationships between them. There are 8494 template features in the data set. After running PSI-BLAST and HHsearch with the SCOP database for each protein, I achieve several templates with scores representing the quality of the database match. The top 10 selected templates are used as features. The remaining templates are set to 0 for each protein. This data set is comprised of 8494 features.

In conclusion, let D_{All} be the set of all of InterPro, BLAST, bio-chemical properties, protein structure information and the designated GO data. I define the example data set as $D_{All} = \{(I_i, C_i, B_i, S_i, G_i) \mid i=1, \dots, k\}$, where k is the number of proteins in fungi, 8208. $I_{1,2\dots,8208} = (p_1, p_2, \dots, p_l) \in \text{IPR}\{0,1\}$ are feature vectors for InterPro terms and l is the number of InterPro (IPR) features in the data set. $C_{1,2\dots,8208} = (c_1, c_2, \dots, c_m)$ are feature vectors for bio-chemical attributes and m is the number of bio-chemical features in the data set. $B_{1,2\dots,8208} = (b_1, b_2, \dots, b_n)$ are feature vectors for BLAST and n is the number of BLAST features in the data set. $S_{1,2\dots,8208} = (s_1, s_2, \dots, s_q)$ is feature vectors for protein structure information and q is the number of products of HHpred features in the data set. $G_{1,2\dots,8208} = (g_1, g_2, \dots, g_p) \in \text{GO}\{0,1\}$ is the class designation (GO terms) and p is the number of GO terms in

the data set. The IPR feature vector l is 3339, the BLAST feature vector, n is 3182, the bio-chemical attribute vector is 474 and the protein structure feature vector, the product of the HHpred, q is 8494, the number of GO terms in the fungi set p , is 444.

C. Experiments

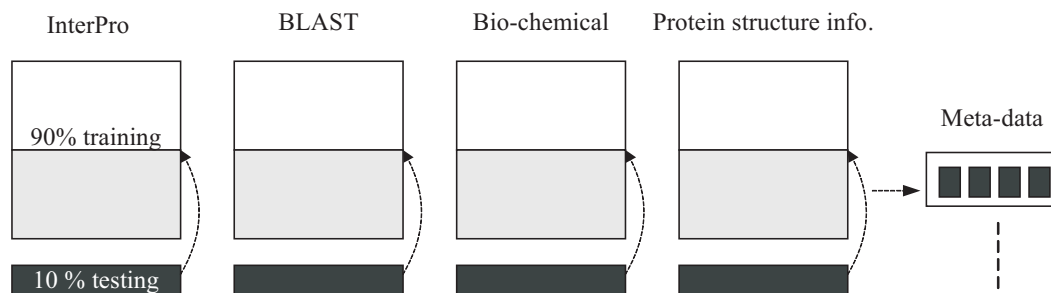
1. Training Procedure (Multi-layered Classifier)

For the multi-layered system, I create a two layered system, which is constituted by of a base-classifier with the four feature sets and a meta-classifier with the product of the base-classifier. The base-classifier serves to learn each feature set independently, while a meta-classifier combines the heterogenous feature sets (Fig. 7). The black set is the test set and both white and gray sets are training sets. In the base-classifier, the 45 % of data (both the white and gray set) are used for training, but they are learned and tested independently in the meta-level. The dotted line shows the flow of the testing process and the solid line in the meta-classifier is the process of creating meta-data. Each GO term was considered as an independent binary classification problem, thus all proteins annotated with a GO term are treated as positive instances (GO+) and the remaining proteins are treated as negative instances (GO-).

a. The Base-Classifiers

The base-classifiers are trained separately for each feature set. Among the feature sets (InterPro, biochemical properties, BLAST and the protein structure information), bio-chemical properties are the only non-sparse set of numeric features, while the others are sparse numeric or binary features. Thus, I use two different classifiers depending on the sparsity. The sparse sets are treated using the same approach as in the previous implementation of AAPFC [42]. Briefly, it is reported that performing

Base-classifier



Meta-classifier

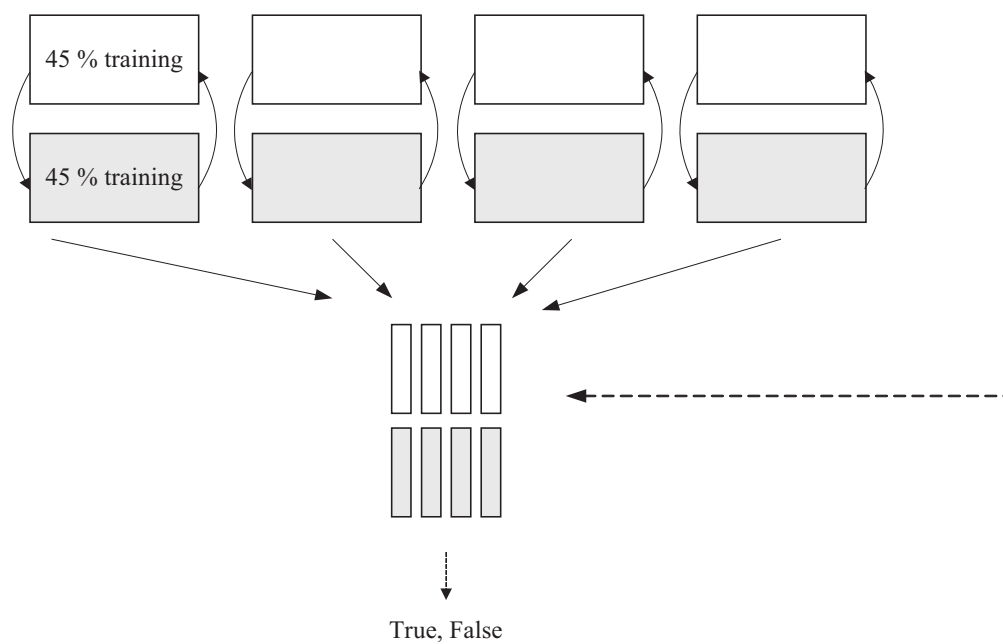


Fig. 7. Multi-layered classifier with four feature sets: Two subsets (white and gray) are used for training and one subset (black) is used for validation. The dotted lines represent the process of testing and the solid lines in a meta-classifier are the process of training.

the chi-square feature selection for the feature selection and making a fully balanced set by selecting the farthest negative instances is carried out. The classifier method is used for the SVM algorithm. The bio-chemical properties data set is also reduced using chi-square feature selection, but I employed different objective functions for the instance selection. I select three times more the nearest negative instances instead of the same size of the farthest negative instances. In the preliminary experiment, the F-measure with SVM using the farthest negative instance was 0.0064, while the adaboosting training model using three times more the nearest negative instances was 0.0173. The adaboosting model is slightly higher than the SVM training set, hence the chemical data sets are applied to the adaboosting [58]. adaboosting is a method for a weak learner algorithm by weighting the error and training it again in several iterations until the error rate is less than 0.5. This approach has an advantage of reducing the errors by executing several times in a small positive set. As a learning algorithm, a linear classifier is applied. The feature selection experiments and under-sampling and SVM, and adaboosting induction were performed with MATLAB [59] using the pattern recognition toolbox [60].

b. The Meta-Classifier

The meta-classifier is trained using the outputs from the base-classifiers. In the InterPro feature case, the total training set is mentioned as T_{I_1} . This set is randomly separated by three subsets. Two subsets $T_{I_{1_1}}$, and $T_{I_{1_2}}$ are used for training while the remaining set $T_{I_{1_3}}$ is used for the validation. The validation set contains 820 proteins and each of the two subsets are composed of 3712 proteins. When the $T_{I_{1_1}}$ subset is used as the training set for the base-classifier, $T_{I_{1_2}}$ is used for classification by the $T_{I_{1_1}}$ training model. Similarly, after learning with the $T_{I_{1_2}}$ subset, $T_{I_{1_1}}$ is projected to the $T_{I_{1_2}}$ model. However, if the feature list is empty in a tested protein,

for example, a protein does not have the InterPro term, the prediction is treated as unknown. When T_{I2} is the meta-data which is tested by T_{I1_1} and T_{I1_2} , T_{I2} is a set of true, false or unknown. Therefore, the meta-data set for the InterPro term is composed of a binary formatted prediction with the same size of the two training sets (T_{I1_1} and T_{I1_2}). The other three feature sets are trained in the same way. When I assume that T_{I1} , T_{B1} , T_{C1} and T_{S1} is the total training set of InterPro terms, BLAST, biochemical properties and protein structure information in a base-classifier respectively, I make meta-data sets which are constituted by T_{I2} , T_{B2} , T_{C2} , T_{S2} , where each T_{*1} is one more columns composed of each feature lists and each T_{*2} is one column of true or false predicted data or unknown data. The row of these data sets is the training protein size. The applied learning algorithm for the meta-learner is a Naïve bayes, which is described in section III.C.3.

2. Testing Procedure

The first step of the test procedure is computing the features and converting them to the appropriate format. The outputs from the base-classifiers for each GO term consist of four columns of feature vectors in binary format. The results of the base-classifiers are used to compute the Bayesian probability of the meta-classifier. From the Naïve bayes, true Bayesian probability is calculated by the product of all possible events. For instance, the true Bayesian probability is measured by multiplying each feature status given the true designated GO term. In each GO classifier, if the “True” probability is larger than “False”, this term is annotated as a gene function, otherwise it is discarded.

Table IX. Performance comparison of four different base-classifiers with different feature sets.

Features	Average GO terms	Sensitivity	Specificity	F-Measure
IPR	17.69	0.1893	0.9682	0.2456
BLAST	67.46	0.0251	0.8547	0.0471
Chem	37.10	0.0099	0.9213	0.0173
Structure	21.11	0.0795	0.9599	0.1304

3. Results

In this method, I learn independent base-classifiers with different feature sets rather than a different classifiers with the same feature set. Thus, I want to compare the performance to a single classifier with one feature and a multi-layered classifier with more than one feature sets in the fungal data, which allows us to compare the performance of the multi-layered learning system. The performance of two different approaches are estimated with 10-fold cross validation. Table IX summarizes the results of the base-classifier. All performance is measured by all the GO classifiers. InterPro terms are the most effective feature set to annotate GO terms, because they have the highest F-measure value. The average predicted GO terms with the InterPro terms, BLAST, bio-chemical properties and protein structure information is 17.69, 67.46, 37.10 and 21.11, respectively. The main shortcoming of the all the base classifiers is the low sensitivity, which results primarily from a high false positive rate and results in a lower F-measure value. The most extreme example is the BLAST feature set which results in approximately 67 GO terms per protein, on average.

In order to compare a single classifier to a multi-layered classifier, I need to set up a meta-classifier algorithm first. I used two different learning schemes, which are

Table X. Performance comparison of two different meta-learning methods with all GO classifiers.

Method	Sensitivity	Specificity	F-Measure
Naïve Bayes with all sets	0.2620	0.9982	0.3355
Naïve Bayes with balanced sets	0.0629	0.9358	0.1140
SVM with all sets	0.0011	0.9273	0.0022
SVM with balanced sets	0.0462	0.3696	0.0052

Naïve Bayes and SVM with all sets. Originally, the meta-data sets, which is a data set with two subsets, are imbalanced sets. The data are composed of 90% of the whole imbalanced fungi set which has small positive proteins and large negative proteins. In classical classification, imbalanced sets provide a poor performance, thus I plan to create a balanced set by selecting the farthest negative instances. Most negative instances are removed and I make a new set whose positive and negative instances are the same. With the balanced set, I also trained the SVM and Naïve Bayes for a meta-classifier. Hence, I get the four different learning schemes, which are summarized in the Table X. When I compared these models, the F-measure trained by the Naïve Bayes with the full set is 0.3355, but that with the balanced set is 0.1140. In addition, the value of the trained SVM and with the full data set is 0.0022 and those with the balanced set is 0.0550. Given four F-measure values, the Naïve Bayes with all training sets have the highest value, hence this combination is used for meta-classifier scheme.

As shown in Table IX and Table X, all F-measure values of the base-classifiers are lower than the multi-layered classifiers. The highest F-measure value with a single classifier is 0.2456 with InterPro terms, but the multi-layered learning model is 0.3335. Finally, I show that a training scheme with a meta-classifier and multi-feature sets

Table XI. Mean classifier-based performance at each cut-off F-measure value.

cut-off F-measure	Sensitivity	Specificity	F-Measure
0.8	0.8914	1.0000	0.9284
0.7	0.8015	0.9983	0.8664
0.6	0.6831	0.9966	0.7685
0.5	0.5673	0.9975	0.6632
0.4	0.5028	0.9978	0.6042
0.3	0.4358	0.9979	0.5393
0.2	0.3718	0.9980	0.4716

(PoGO) provides superior results to any single classifier.

The performance of the multi-layered classifier is reported two ways. One is a classifier-based result and the other is a protein-based result (Table XI, Table XII). The classifier-based results show the performance in each classifier, i.e., I check the error of all proteins in terms of one GO term and measure the sensitivity and F-measure. The protein-based method measures the value in each protein, which is usually used for a benchmark with other protein functional classification programs. Table XII shows the average F-measure, where the protein-based way is calculated by the selected GO terms whose F-measure value in the classifier-based value is larger than cut-off value. According to the table, a 0.7 cut-off F-measure means that I select the GO terms whose classifier-based F-measure values are higher 0.7% and I calculate the sensitivity, precision and F-measure with these GO terms in each proteins.

From this view point, I compare the classifier-based performance between AAPFC and PoGO at the same cut-off F-measure. The number of selected GO terms in each

Table XII. Mean protein-based performance at each cut-off F-measure value.

F-measure cut-off value	Sensitivity	Specificity	F-Measure
0.8	0.9808	0.9985	0.9872
0.7	0.8597	0.9904	0.8948
0.6	0.6910	0.9875	0.7493
0.5	0.6299	0.9904	0.6913
0.4	0.5893	0.9904	0.6558
0.3	0.5357	0.9892	0.6062
0.2	0.5055	0.9885	0.5759

range is shown in Fig. 8 (a), and Fig. 8 (b) is the number of annotated proteins. In both the number of GO terms and annotated proteins, PoGO has more terms and proteins, that is, I can annotated more proteins with PoGO application. Based on this experiment, I measure the average F-measure value using the protein-based method by 10-fold cross validation (Fig. 9). At each cut-off value, the average value in PoGO is slightly higher than in AAPFC.

I compare the classifier-based performance of GO terms that were trained by both AAPFC and PoGO. Table XIII shows 12 randomly selected GO terms among 409 classifiers. The average sensitivity of AAPFC is 0.2317, while PoGO is 0.3021. In the case of precision, PoGO is 0.6349 but AAPFC is 0.8562. Even though precision in PoGO is less than AAPFC, the overall F-measure value in PoGO is higher than in AAPFC. This implies that the average number of GO terms annotated by AAPFC is a larger than PoGO, resulting in higher value in the average of precision and a lower value in average sensitivity.

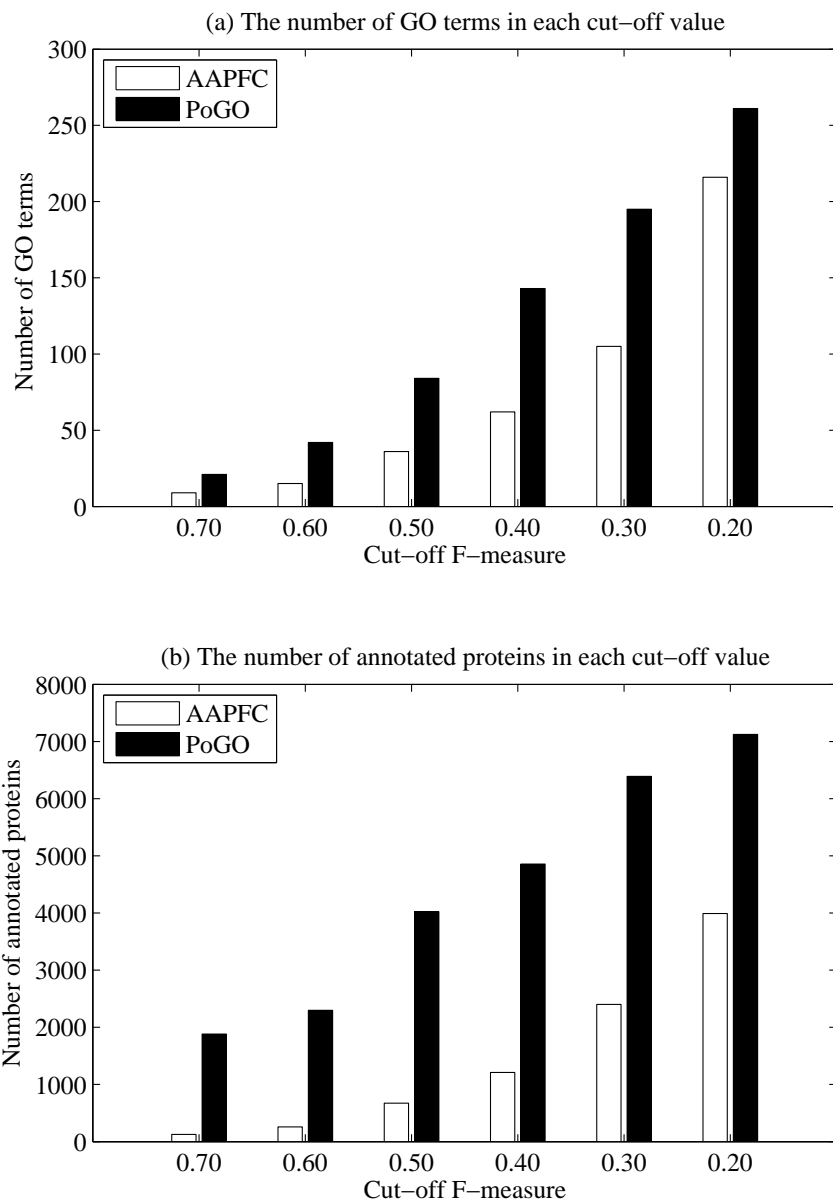


Fig. 8. AAPFC and PoGO: (a) The number of GO terms at each cut-off F-measure value. (b) The number of annotated proteins at each cut-off F-measure value.

Table XIII. Classifier-based performance comparison of 12 shared GO terms among 409 GO terms in AAPFC and PoGO.

GO terms	PoGO			AAPFC		
	Sensitivity	Precision	F-Measure	Sensitivity	Precision	F-Measure
GO:0000243	0.4167	0.7143	0.5263	0.0610	1.0000	0.1150
GO:0003729	0.6296	0.6538	0.6415	0.0629	0.9500	0.1180
GO:0005355	0.5357	0.9375	0.6818	0.4211	1.0000	0.5926
GO:0005485	0.4167	0.7143	0.5263	0.1519	0.8571	0.2581
GO:0005682	0.2105	0.8571	0.3380	0.0660	0.9286	0.1232
GO:0005740	0.1765	0.6429	0.2769	0.2766	0.9286	0.4262
GO:0006116	0.2581	0.7273	0.3810	0.1184	0.8182	0.2069
GO:0006384	0.6667	0.4615	0.5455	0.0518	1.0000	0.0985
GO:0009228	0.8000	0.8000	0.8000	0.1156	0.8947	0.2048
GO:0030503	0.2424	0.6667	0.3556	0.1375	0.9167	0.2391
GO:0043161	0.1585	0.7222	0.2600	0.0778	0.8750	0.1429
GO:0048017	0.3571	0.6250	0.4545	0.1446	0.8000	0.2449
:	:	:	:	:	:	:
Average	0.2317	0.6349	0.3021	0.1407	0.8562	0.2227

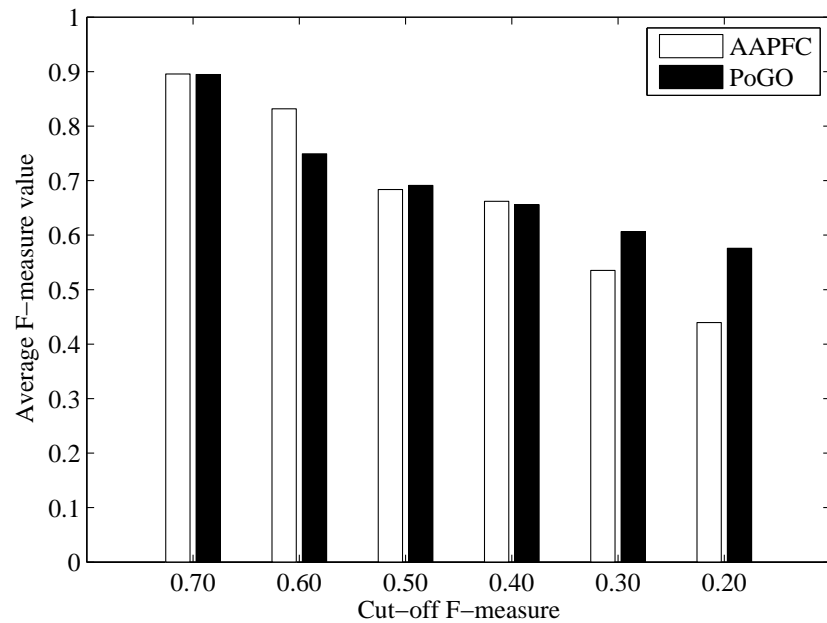


Fig. 9. Average F-measure value at each cut-off F-measure value in AAPFC and PoGO.

D. Implementation

A series of pre-processing steps, such as feature selection and under-sampling in AAPFC and PoGO are performed in MATLAB. The SVM learning and classification are also performed in MATLAB [59] using the pattern recognition toolbox [60]. AAPFC and PoGO web server is written in JavaScript, PHP and MATLAB (Fig. 10) for file handling, training, testing, and displaying web pages. A MySQL database is used for retrieving GO terms and descriptions which are then displayed on the web site.

1. Batch Job Queuing System

The web-based system for AAPFC and PoGO utilizes a queuing script in order to support a queuing system, which enables it to achieve each sequence sequentially. The submitted sets of FASTA sequences are automatically divided by each sequence. A batch script for each separate sequence is created to submit a job to a queuing system. The web process generates an unique ID for each job. But, in case of multi-sequences, the system generates an unique representative ID which encapsulates several job IDs. The web-server sends an email to an user with this unique ID so that the user can check out the result on the web-site. After a job is submitted, the web page is redirected to the result or the progress with-in 3 minutes. Even though the end-user uses a web browser to go to the other site, all jobs are submitted successfully in the queuing system and it sends all results by e-mail to prevent a wait on the web site for a lengthy period.

2. System Output and Web Server

A diagram of the full system architecture in PoGO is shown in Fig. 10. The user inputs FASTA formatted protein sequences. The gray square boxes are the tools or applications for predicting preliminary data and the pre-processed data are represented by the black square box. The web process parses this temporary file and makes a binary format data file. The generated data is attained by the feature and instance selection and predicts the GO terms. The supported supplementary data is a detailed description of GO terms and InterPro terms. The MySQL database is supported to show the GO description matched with the predicted GO terms. The results page is classified by two kinds of web pages. “Brief Overview” just supplies all predicted GO terms and InterPro terms, but “Detailed Overview” supplies all information of

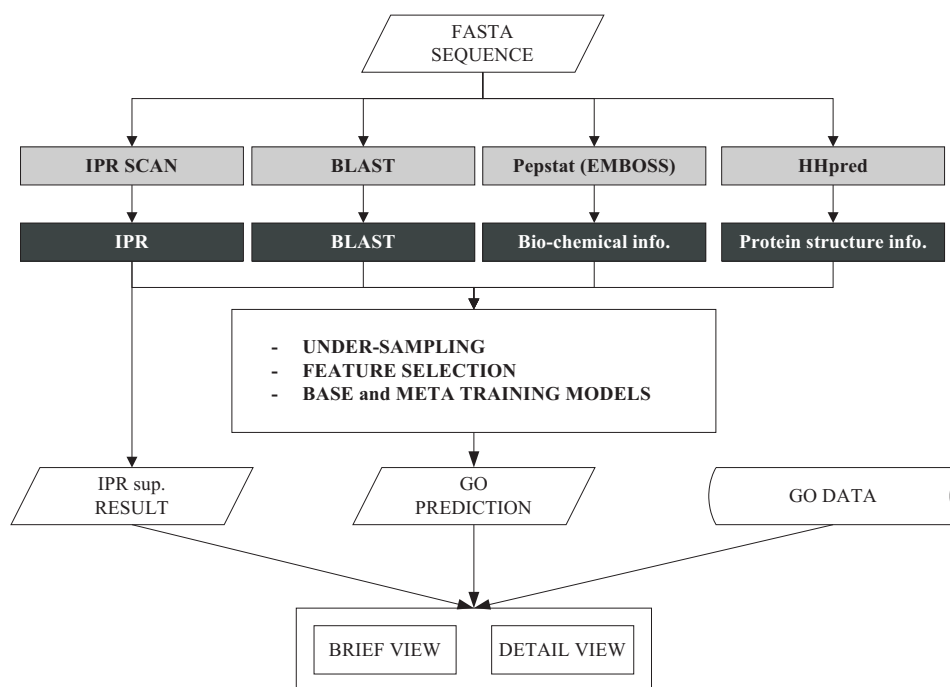


Fig. 10. System architecture in PoGO.

Table XIV. Properties of GO annotation tools: ‘o’= support completely, ‘ Δ ’= support partially and ‘x’= does not provide yet. GOPET does not support Cellular Component category.

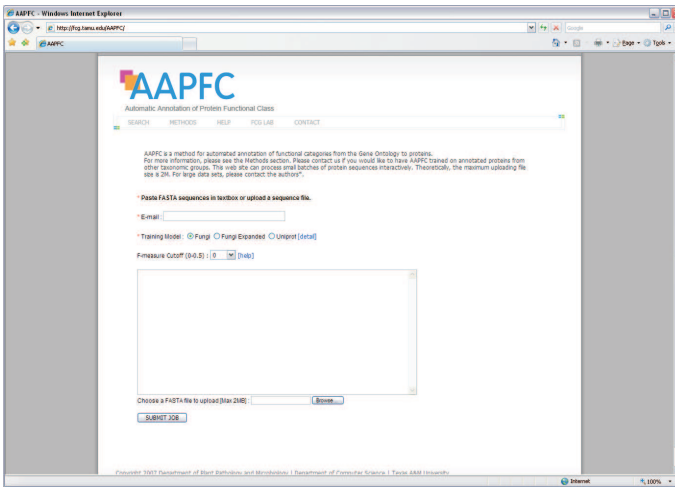
	GOPET	GOFigure	GOtcha	AAPFC, PoGO
Supporting multiple sequence	o	o	x	o
Supporting e-mail result	x	o	x	o
Supporting all GO categories	Δ	o	o	o
Tab-delimited file downloadable	x	o	x	o

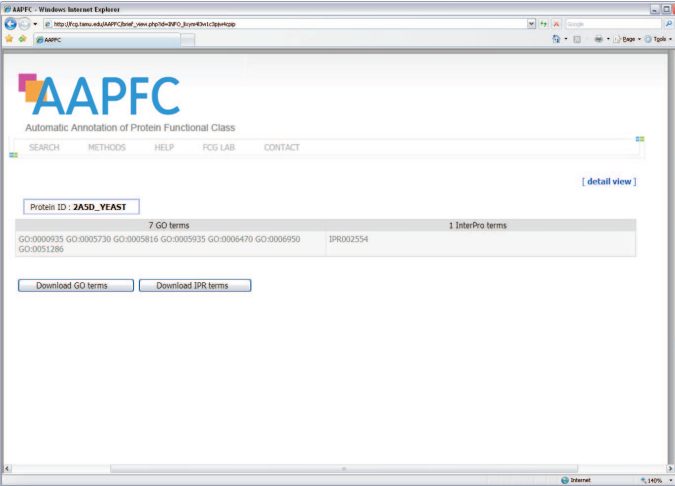
all predicted GO terms and InterPro terms.

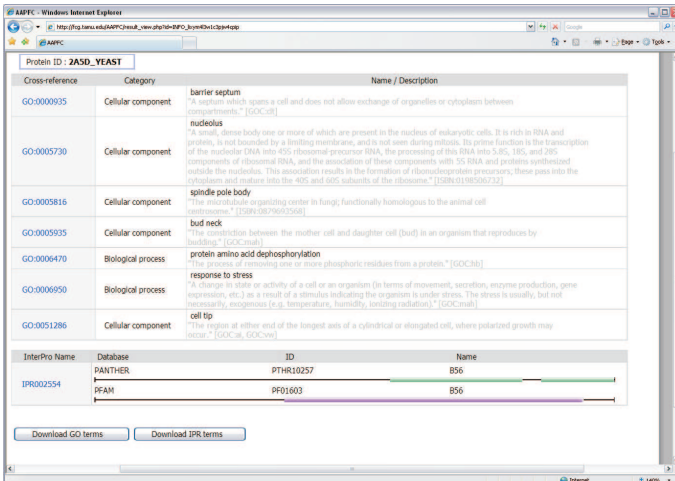
Fig. 11 (a) is an initial web server of AAPFC and PoGO and Fig. 11 (b) is the brief result for each protein which shows only GO terms and InterPro terms, and Fig. 11 (c) is the detail for each terms including a description and parsing value based on the database of InterPro terms. This result is also provided by a tab-separated file for GO terms and InterPro terms are supported to import into other applications. Table XIV is the benchmark of several GO annotation tools and the proposed tool. AAPFC and PoGO support not only multi-sequences for an input sequence, but also all GO terms in three categories. However, some other systems provide a partial functional categories or support only one sequence to process one-by-one. In this case, the tab-separated file for GO terms and InterPro terms to import into other applications and the e-mail forwarding system are advantages in AAPFC and PoGO.

3. User Interface

Users can paste protein sequences directly into the web page or upload up a file of protein sequences up to 2 MB in size (approximately 5500 proteins) for processing by

(a) 

(b) 

(c) 

Cross-reference	Category	Name / Description
GO:0000935	Cellular component	barrier septum "A septum which spans a cell and does not allow exchange of organelles or cytoplasm between compartments." [EGOC:01]
GO:0005730	Cellular component	nucleolus "A small, dense body one or more of which are present in the nucleus of eukaryotic cells. It is rich in RNA and protein, is not bounded by a limiting membrane, and is not seen during mitosis. Its prime function is the transcription of the molecular DNA into 45S ribosomal precursor RNA, the processing of this RNA into 5.8S, 18S, and 28S components of ribosomal RNA, and the association of these components with 5S rRNA and proteins synthesized outside the nucleolus. This association results in the formation of ribonucleoprotein precursors, these pass into the nucleolus and mature into the 40S and 60S subunits of the ribosome." [EGOC:01 98500732]
GO:0005816	Cellular component	spindle pole body "The microtubule organizing center in fungi; functionally homologous to the animal cell centrosome." [EGOC:01 98500730]
GO:0005935	Cellular component	bud neck "The constriction between the mother cell and daughter cell (bud) in an organism that reproduces by budding." [EGOC:01]
GO:0006470	Biological process	protein amino acid dephosphorylation "The process of removing one or more phosphoric residues from a protein." [EGOC:01]
GO:0006950	Biological process	response to stress "A change in state or activity of a cell or an organism (in terms of movement, secretion, enzyme production, gene expression, etc.) as a result of a stimulus indicating the organism is under stress. The stress is usually, but not necessarily, exogenous (e.g. temperature, humidity, ionizing radiation)." [EGOC:01]
GO:0051286	Cellular component	cell tip "The region at either end of the longest axis of a cylindrical or elongated cell, where polarized growth may occur." [EGOC:01 EGOC:02]

InterPro Name	Database	ID	Name
IPR002554	PANTHER	PTHR10257	B56
	PFAM	PF01603	B56

Fig. 11. Web-server page for AAPFC.

the server. Users can select models trained with one of the three data sets, original fungi, extended fungi and UniProt, as well as the desired F-measure cut-off among the models within each data set. This option was implemented so that users can exclude GO terms that have low F-measure values and thus have a higher probability of producing erroneous annotations. For example, in the Fungi model, 10 GO terms are removed from consideration if the user selects a cut-off of 0.05 and 77 GO terms would be removed if the user selects a cut-off of 0.10. The results are displayed in a table which contains links to a detailed page for each protein. The detailed page contains a graphical view of the InterPro annotations along with a list of predicted GO terms and descriptions.

E. Summary

In this chapter, I describe a method for assigning GO terms using a multi-layered classifier and multiple heterogeneous feature sets. By creating a training model in each feature set and integrating the heterogeneous data, I developed a multi-layered scheme that annotates GO terms more accurately. In the previous GO annotation system (AAPFC) each protein should have one or more InterPro terms, however, this approach employs more attributes, and proteins missing features are allowed. Consequently, more proteins can be used for training and testing, and ultimately, more proteins can be annotated. The performance reported using the precision and F-measure metrics is better than for AAPFC. The sensitivity, precision and F-measure in AAPFC is 0.8826, 0.6371 and 0.6835 respectively, while PoGO has 0.7290, 0.8422 and 0.7518, when I cut-off at 0.5 in the classifier-based F-measure value. The reason is that three more feature sets complement to decide a gene function. With this experiment, I demonstrate that many attributes contribute to determining gene

function.

The web server is implemented in PHP, MATLAB and pattern recognition toolbox for MATLAB. The results are displayed in a table which contains predicted GO terms and InterPro terms and links to a detailed page for each protein. The detailed page contains a graphical view of the InterPro annotations and along with a list of predicted GO terms and descriptions including the category. In addition, the tab-separated data files of the GO annotations are available for download, enabling to import of the data into other programs.

AAPFC and PoGO systems learn independent classifiers for each GO term. This architecture has the advantage in that individual GO term classifiers can be re-trained over time without the need to re-train the whole system. However, many information is loosed due to the dimension reduction. Moreover, GO terms are structured in a Directed Acyclic Graph (DAG), which describes the functional relationship between the terms. Hence, in the next chapter, I develop a system for considering GO structure with the Bayesian probability given the InterPro terms or multi-feature data set containing three feature types.

CHAPTER IV

GENE FUNCTIONAL PREDICTION USING HIERARCHICAL GENE
ONTOLOGY INFORMATION

A. Related Work

With the increasing quantities of genome sequence data, there is an equal increase in the need for automated genome annotation methods. The development of an automated method for the annotation of predicted gene products (proteins) with functional categories is becoming increasingly important, in order to present genome sequences and genome annotations to biologists in a useful way. The Gene Ontology (GO) is a controlled vocabulary of keywords and phrases for describing gene function that is organized in a Directed Acyclic Graph (DAG). Thus, many systems to perform protein functional annotation have been developed, that employ various sources of protein information as features, including protein functional sites [61, 42], sequence similarity [8, 9, 25], gene expression patterns [62, 28], and others.

The previous approaches are independent GO classifiers in each model which require enough GO terms for the validation. The treated sets are composed of proteins which have GO terms annotated by 10 or more proteins, the related GO terms and InterPro terms and other feature sets. The original number of GO terms in fungal proteins from the UniProt database is 3199, but only 459 GO terms are used for the training model in AAPFC, and PoGO has 443 GO classifiers. Around 15% of GO terms among the whole sets learn as a classifier. Hence, the first objective in this suggestion is employing more annotating GO terms. Moreover, the hierarchical GO structure, which is a good way to describe the relationship between gene functions, is not considered in both AAPFC and PoGO. This structure represents

the relation between top and down, where the higher level gene function (parent node) includes the deeper-level function (child node). As an illustration, the ancestor of "GO:0005634:nucleus", "GO:0043231:intracellular membrane-enclosed organelle", implicitly includes the "GO:0005634" function. This parent term's information enables us to describe the functional relationship. Thus, building model with the hierarchical GO structure is the second objective.

Often, automated gene annotation methods ignore the hierarchical nature of the controlled vocabulary in order to simplify the problem [42, 8, 62, 28, 25]. The use of hierarchical information for the functional prediction has been employed by several authors [17, 18, 27, 19]. Eisner et al. [18] compared four different training strategies, which are "Exclusive", "Less Exclusive", "Less Inclusive" and "Inclusive" by the ensemble classifier of SVM, probabilistic suffice tress (PSTs) and the BLAST. "Exclusive" means only GO terms are considered as the positive examples, and the others including the descendent, and the ascendant are treated as the negative examples. "Less Exclusive" is same as the "Exclusive" set except for the descendent, which is not used for neither positive training sets nor negative ones. "Inclusive" is GO terms in which all descendants are treated as the positive, but ancestors are not used in the training sets. Besides, the ancestors of GO terms are treated as the negative examples. In "Less exclusive", all ancestors of GO terms are not considered. From four different training strategies, authors point out the "Inclusive" outperformed in an overall performance. Consequently, the training sets with the hierarchical information contribute to the outperformed results. Shahbaba and Neal [19] suggest three models, which are the multinomial logit (MNL), the hierarchy based MNL and the correlated MNL (corMNL) that are introduced between the parameters of nearby classes in the hierarchy structure. MNL is simple multinomial logit but ignores hierarchal information, hierarchy based MNL defined set of nested MNL model and corMNL is

considered the prior between the parameters of near by class. Even though the correlated MNL in hierarchy outperformed the other two methods and the traditional decision tree such as C5, their assumption in this structure is that a simple tree-like structure, where one node has only one parent, is constructed. This approach is not sufficient to apply to the GO structure, because of multiple parents.

Several studies have utilized Bayesian networks for protein annotation [63]. Multi-score Association of Genes by Integration of Clusters (MAGIC) [21], uses the Bayesian network with gene expression data and protein-protein interaction data in order to annotate gene function. However, they employ it in order to join to the heterogeneous biological information rather than to apply the hierarchical GO structure. Barutcuoglu et al. [17] also used the Bayesian network for the purpose of annotation of multi-label prediction. Independent SVM classifiers are learned for each GO term which are then combined with a Bayesian network. Their approach overcomes the problem of not consulting the child classifier at all if the parent term annotates as the negative class [64]. King et al. [27] predict gene function by the relationship between GO annotations based on the decision tree and Bayesian network. However, the classifiers are trained independently given the SVM classifier and Bayesian probability of the child term. Hence, I suggest a new multi-label approach which considers all GO terms simultaneously by use of a Bayesian network.

I describe two multi-labeled approaches by considering the DAG structure of the GO using a Bayesian network. The first method uses only InterPro terms in the construction of GO structure, where the training model is a single classifier like AAPFC. The training scheme constructs the Bayesian probability matrix with InterPro feature sets. The calculation uses Naïve Bayesian probability, which is true InterPro terms given the true or false designated GO term, because each InterPro term is independent. After inference this approach to examine unlabeled proteins, several candidate

GO terms are listed. However, this temporary annotation has many false positive errors, so a filtering step processes by the relationship between GO terms and InterPro terms. Finally, the results are compared with non-hierarchical protein domain annotation (AAPFC).

The second approach employs two more feature sets, except for InterPro terms, and a multi-layered classifier is applied, thus it is composed of a base-classifier and meta-classifier. Similar to the above approach, each feature is learning independently in a base-classifier. The built probabilistic matrices are three sets, comprised of InterPro terms, BLAST and protein structure information. In addition, a meta-classifier is also trained by the Naïve Bayes with meta-data which is the product of the base-classifier. This approach has an advantage of having many feature lists for annotation of function. Or the above two models, employing the hierarchical structure with multi-features or protein domains, are contrasted with themselves in order to analyze the effect of various features in the hierarchical GO-structured model. Lastly, the comparison of multi-feature with embedding GO structures or without them (PoGO), supports the difference of the hierarchical information in the same featured domain.

B. Protein Domain Feature

1. Proposed Method

a. Data Set

This study employs InterPro terms as features, which reflect the presence of conserved functional domains in the proteins. The data set comprised annotated protein sequences from the UniProt database. Each protein has one or more InterPro terms, since this is a key feature. Each protein has previously been annotated with one or

more GO terms, but proteins that lack GO annotations are removed. In addition, annotations with the evidence code “Inferred from Electronic Annotation” (IEA), which indicates annotations derived from other automated annotation methods are also ignored. The remaining annotations are those that have been reviewed by subject matter experts. With this data set, two matrices are created, which are $G(i, k)$ and $I(i, j)$, where i is the number of proteins, k is the number of GO terms and j is the number of InterPro term. In the matrix I , if the i th protein has the j th InterPro term, $I(i, j)$ represents a binary value indicating the presence or the absence of the functional domain. The matrix G is also built in the same way. The original fungi data set consists of 6711 proteins, 3339 InterPro terms, and 3096 GO terms. However, for evaluation purposes, I performed 10-fold cross validation, hence, around 6040 proteins are used for the training set in each validation.

Each node in the GO structure represents a protein function. Nodes which are closer to the root node (higher level term) represent more general functions, while deeper-level nodes in the structure represent more specific functions. Proteins annotated with a GO term are also implicitly annotated with all of its parent GO terms. Thus, parent GO terms may also be used to describe a protein’s function. For instance, the GO_2 includes the GO_4 function in Fig. 12. Hence, if a protein has several GO terms including GO_4 , GO_6 and GO_8 displayed in dark gray nodes, this protein’s function can also be described by the parent nodes GO_1 , GO_2 , GO_3 and GO_7 , displayed in light gray. Hence, this protein’s function can be described by the set of GO terms that includes GO_1 , GO_2 , GO_3 , GO_4 , GO_6 , GO_7 , GO_8 . However, GO_5 and GO_9 , displayed in white nodes, are not included in the data set, since they are neither original GO terms nor an annotated GO terms’ parent. Finally, k in the matrix G is the list of not only original GO terms but also of the parent terms of the original GO terms. Fig. 13 is a real example of CHAC_YEAST, which includes all

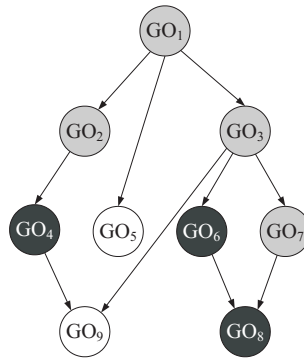


Fig. 12. Hierarchical structure of the Gene Ontology.

GO terms except for the white colored GO terms as a training model.

b. Bayesian Network

In various studies with the DAG graphical model, including bioinformatics (protein folding, gene network), document classification and image processing, Bayesian network is applied [17, 64, 65, 66, 67, 68, 69]. A Bayesian network is a conditional probability between the parent node and child in the graphical based environment. The node in the GO graph indicates the events and relationship between the nodes that stands for the conditional dependency. Usually, the probability of the root node is assigned the prior probability distribution and the other nodes are assigned with the conditional probability given the parent node, that is, all child nodes depend on the parent node probability distribution.

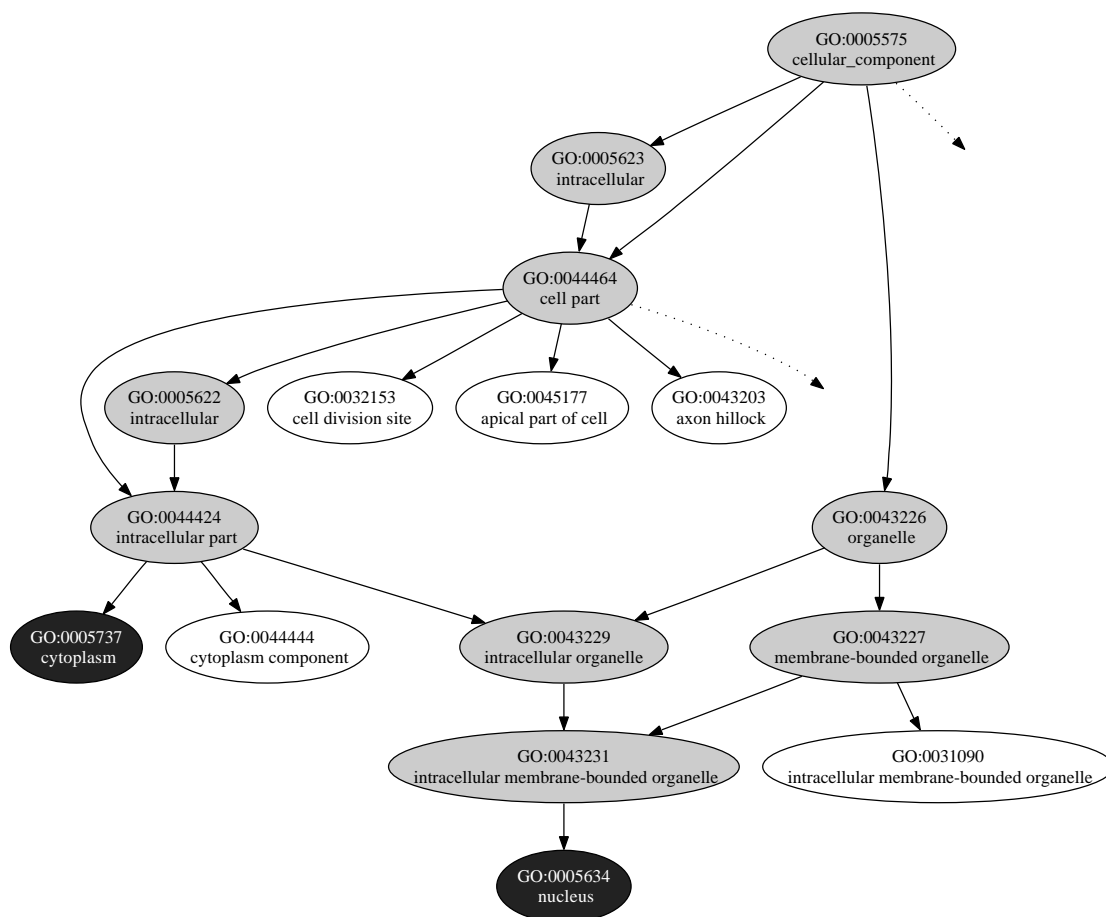


Fig. 13. GO terms in cellular component structure annotated in CHAC_YEAST: The dark gray GO terms (GO:0005634 and GO:0005737) are what a protein has, but light gray GO terms are not included in the original training sets. From the assumption of the experiment, these light gray GO terms are considered to be training GO terms. The white GO terms are not considered for the training sets, since they are not linked to any dark gray GO terms.

2. Experiment

a. Training Procedure

The original data set is a fungi specific data set made up of 3339 InterPro terms, 6711 proteins and 3096 GO terms, but GO classifiers are increased to 4647, since all parent nodes in a GO hierarchy are also added in the training set. The experiment is performed by 10-fold cross validation, resulting in 6040 proteins being used for the training sets in each validation. The training model is the probabilistic matrices which are measured by the number of InterPro terms given the true or false GO term, since each feature is independent in each classifier. In other words, for the InterPro term j and GO term v , $P(I_j|G_{v=F})$ and $P(I_j|G_{v=T})$ are measured, where I is the InterPro term matrix and G is the GO term matrix. These probability matrices illustrate the conditional probability in terms of InterPro terms and GO terms. All possible events for j are 3099 and v is 4674, hence, whole training matrices in each validation are $2 * 3096 * 4674$. For example, in terms of "IPR007587 : SIT4 phosphatase-associated protein" and "GO:0005634 : nucleus", when the number of proteins which have IPR007587 is 10 in the fungal set and among them only 7 proteins have also GO:0005634, the Bayesian probabilities are $P(I_j|G_{v=F}) = 3/6040$, $P(I_j|G_{v=T}) = 7/6040$. In Fig. 14, $T_{prob}(v, I)$ can be calculated using this training conditional matrix tables. When the unknown tested protein has one or more InterPro term, $T_{prob}(v, I)$ can be the product of all conditional events like $\prod_{j=1}^m P(I_j|G_{v=T})$, m is all conditional InterPro terms.

b. Test Procedure

The first step to annotate the function for the unknown proteins is obtaining InterPro terms (protein functional domains) using the InterPro Scan application [37]. Next,

the InterPro feature lists are applied to the algorithm shown in Fig. 14 to each of the three graphs (*biological process*-G0:0008150, *cellular component*-G0:0005575 and *molecular function*-G0:0003674). From the root node, the Bayesian probability is calculated by constructing the Bayesian network and the gene functions are assigned. The basic idea is that if the true Bayesian probability is larger than the false given the parent's condition, this term is annotated as a protein function after the filtering process. The Bayesian network is calculated by the parent conditional event which is generated by 4.1. The posterior probability for each GO term is calculated given the InterPro terms as shown in 4.2, where v is the GO term in the GO structure.

$$P(X_1, X_2 \cdots, X_v) = \prod_{i=1}^v P(X_{i \in \{T, F\}} | Par(X_{i \in \{T, F\}})) \quad (4.1)$$

where X is the GO term in the GO structure given the selected InterPro terms, and Par stands for parent terms.

$$P(X_v) = P(G_v | I_1, \cdots, I_m) = \frac{P(G_v)P(I_1, \cdots, I_m | G_v)}{Z} \quad (4.2)$$

m is the InterPro terms determined by the InterPro Scan in a tested protein and Z is the normalized constant value. The likelihood probability in 4.2 can be simplified, since I_i is independent of $I_j (j \neq i)$.

$$P(I_1, \cdots, I_m | G_v) = \prod_{i=1}^m P(I_i | G_v) \quad (4.3)$$

The conditional probability of 4.1 can be inferred from the training set.

Fig. 15 is the process to construct a Bayesian network from the root node. In most cases, the root node (G0:0008150, G0:0005575, and G0:0003674) has a higher value in true probability, because the root node includes all child terms, hence the


```

Algorithm Predict_GO{
     $G \leftarrow$  GO Structure;
     $I \leftarrow$  InterPro terms;
     $v \leftarrow G_{top}$ ;
     $par =$  empty;
    Construct_Bayesian_Network( $v, G, I, par$ ); filter C; }

Algorithm Construct_Bayesian_Network( $v, G, I, P$ ) {
     $P(v) \leftarrow$  set  $T_{prob}(v, I)$  &  $F_{prob}(v, I)$ ;
    if  $T_{prob}(P(v)) > F_{prob}(P(v))$  {
         $C(v) \leftarrow$  add  $v$ , where  $C$  is the candidate list; }
    for each child vertex  $v'$  of  $v$  in  $G$  do {
         $par \leftarrow$  parent of  $v'$ 
        Construct_Bayesian_Network( $v', G, I, par$ ); }
    Return C;}

```

Fig. 14. Inference algorithm to annotate GO terms using the hierarchical GO-structured model with InterPro terms.

root node is assigned as true. The next visited terms are the linked child nodes. If $P(\text{GO:0005634}=\text{T}|\text{GO:0005575}=\text{T})$ is larger than $P(\text{GO:0005634}=\text{F}|\text{GO:0005575}=\text{T})$, where GO:0005575 is already assigned as a true, GO:0005634 is considered as a true annotated term. In Fig. 15, GO:0031012 and GO:0031975 are also calculated the same way, since they are linked to the root node. However, GO:0044464 has multiple parents. Therefore, the Bayesian probability of GO:0044464 can be measured, after measuring the probability of GO:0005634 first. In conclusion, from the root node to the leaf, each of the node probability is measured depending on the parent node status which is annotated as true or false. If the parent node is decided on as the true function, the child node is influenced by the true parent probability, otherwise, it is affected by the false probability. Only the true probability of the current term given the parent's condition is larger than those of the false, $P(X_1, \dots, X_{v=T})$ is greater than $P(X_1, \dots, X_{v=F})$, term v is accepted as a candidate GO term, otherwise the term v is removed from consideration and its conditional probability is included in the computation of the probability of the child terms. All GO terms in the GO structure are explored as this way and assigned it accordingly.

c. Filtering Step

After the candidate GO terms are determined, a filtering step described in section II.B.4 is applied. From all the proteins in the Uniprot set, which include not only fungi but also other species, I make a list of the occurrence of InterPro terms and GO terms. Then, I examine the candidate GO term and the assigned InterPro term for the protein and determine if a GO term-InterPro term exists in the list. If there is not a GO term-InterPro term pair in the list, then the GO term is removed from the list. This filtering step serves two purposes. 1) It prevents higher level parent GO terms that do not exist in the training sets from being assigned to annotated

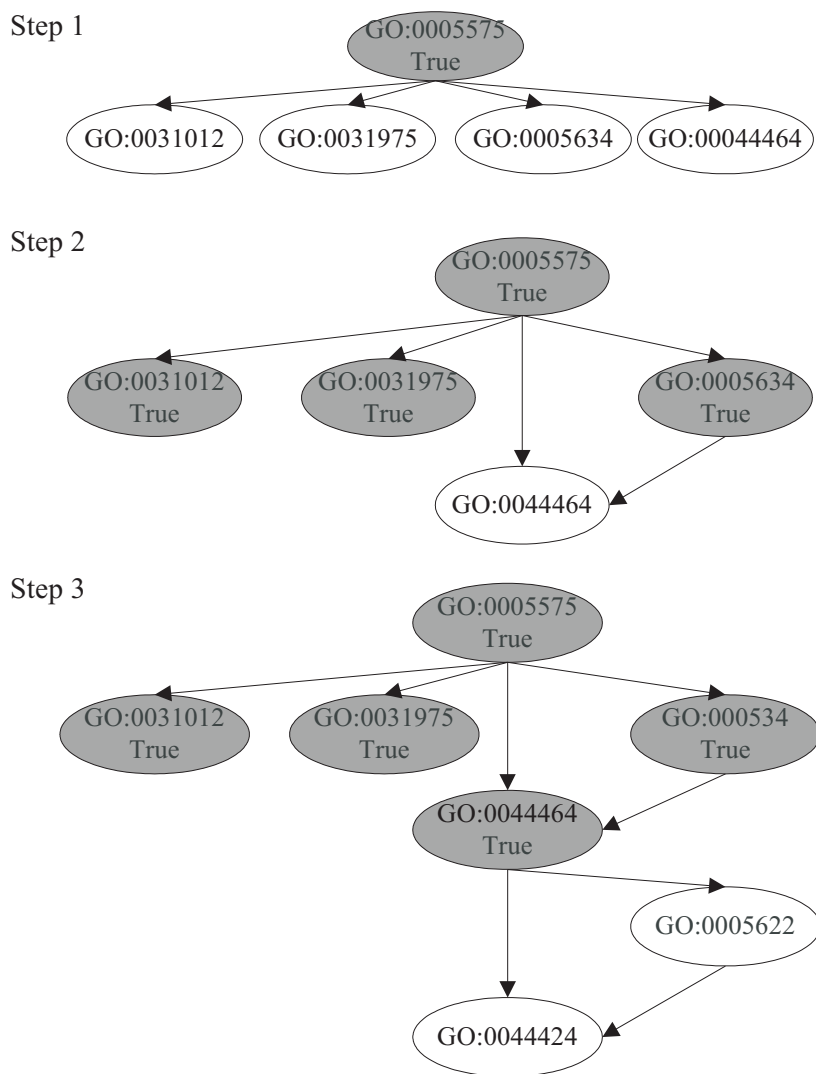


Fig. 15. Construction of the Bayesian network from the root node in each three categories.

Table XV. Comparison of "WITH the filtering step" and "WITHOUT the filtering step".

	Sensitivity	Precision	F-Measure
With filtering	0.2628	0.2894	0.4629
Without filtering	0.1765	0.2219	0.4377

proteins. While such higher level terms may have some utility, depending on the type of downstream sequence analysis, I felt it was important not to allow the annotation system to transitively assign a higher level, and less informative annotations to new proteins. 2) Protein functional domains are determinants of function, so I would expect that most of the information represented in this data set to exist as positive relationships between InterPro terms and GO terms. As I have shown previously, annotated GO terms that are not associated with an InterPro term in the training set are often false positives. Thus, this filtering step has been shown to improve the performance of the classifier (Table XV), where the performance is measured by all GO terms without any cut-off in F-measure.

3. Results

Both AAPFC [42] and this approach are implemented using InterPro terms as the features set, but the primary difference is the consideration of the hierarchical structure. Thus, a comparison of the performance of the two systems may be used to understand the importance of the GO structure in automated classifiers. The first analysis is counting the number of GO terms at each cut-off F-measure value, which is measured using the classifier-based approach. When a F-measure cut-off value is set to a high value, such as 0.7 or 0.8, the number of those GO terms is small (Fig. 16

(a)). AAPFC has only 36 GO terms with an F-measure higher than 0.5, while the hierarchical GO-structured model has 324 GO terms at the same cut-off level.

In addition, using AAPFC, only 459 total GO terms could be trained, while this method can train 1113 terms. Based on this criterion, the hierarchical training model has more GO terms than AAPFC. Fig. 16 (b) shows the number of proteins at each cut-off level. I count the proteins which have the GO terms which are represented in Fig. 16 (a). When the cut-off value is higher, the proportion of proteins is lower, due to the small number of selected GO terms. In other words, in decreasing the cut-off value, the number of proteins that can be assigned GO terms are increased. In terms of annotated number of proteins, a hierarchical GO-structured model has more proteins than AAPFC, which implies that a Bayesian network with structure predicts more GO terms and proteins, resulting in improved the performance.

Fig. 17 shows the average classifier-based F-measure at each cut-off F-measure value, where the x axis is the cut-off level and the y axis is the average F-measure value of those proteins which have the selected GO terms by the F-measure cut-off value. Although the same feature set is employed in both of the two systems, the performance in the new approach surpasses that of AAPFC, except at the two lowest cut-off levels, though these two levels have more proteins in the hierarchical GO-structured model.

Another point regarding the performance is that of the average GO level for annotated GO terms. If a gene functional application is predicted a specific function, but it is not annotated to its related specific function, it is considered that the consistency is violated. Cesa-Bianchi et al. [64] designed a scheme which satisfies the consistency by not considering the child terms if the parent term predicts as negative. While this idea satisfies with the consistency, the chance to assign the deeper-level (i.e. farther from the root node) terms is decreased. However, the annotation of

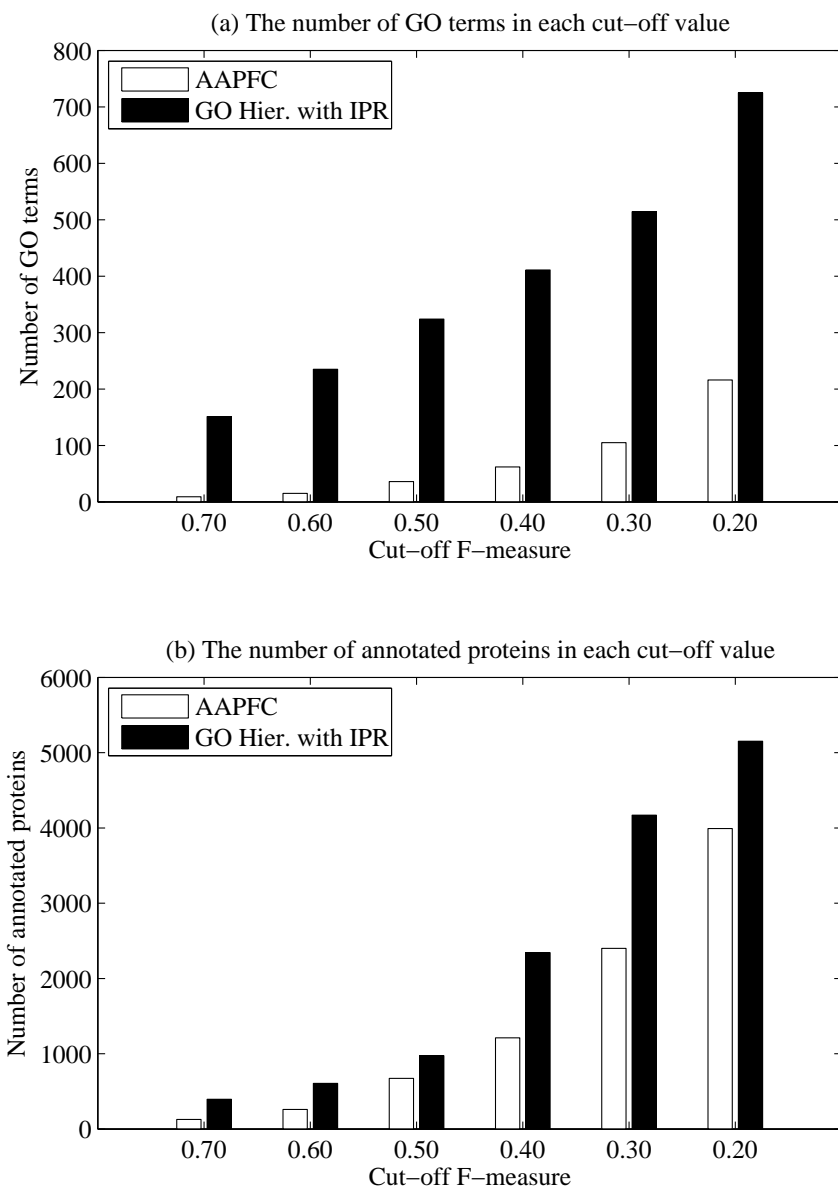


Fig. 16. AAPFC and the hierarchical GO-structured model with InterPro terms: (a) The number of GO terms at each cut-off F-measure value. (b) The number of annotated proteins at each cut-off F-measure value.

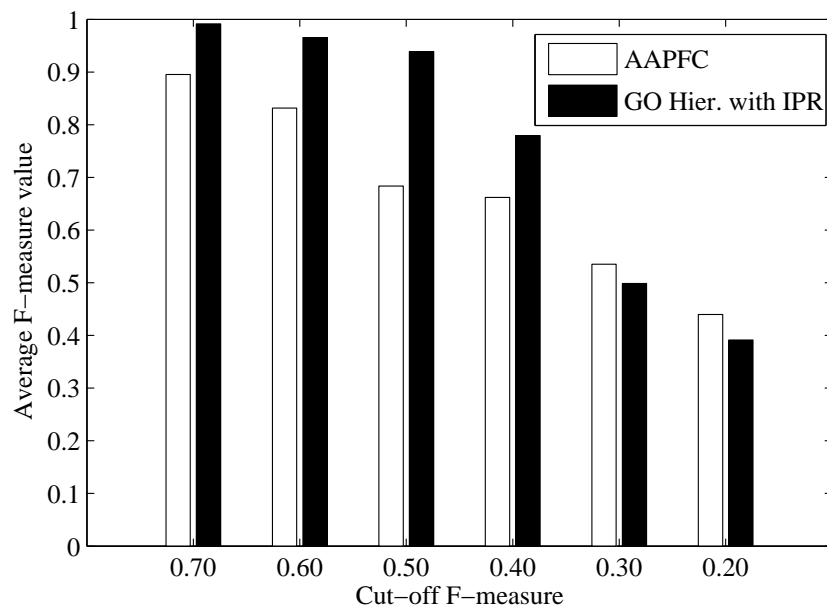


Fig. 17. AAPFC and the hierarchical GO-structured model with InterPro terms. The average F-measure at each cut-off F-measure value.

deeper-level terms is more meaningful, since the high-level terms entail a broader function and the deeper-level terms are more specific in meaning, where the level of the root assigns 0 and the child node increases the level, which is mentioned as a deeper-level term. Thus, the average level in all assigned GO terms is another checking point to evaluate a performance. If the selected GO terms are high-level in each cut-off range, the predicted terms are broad functions, thus the average level in each range should be larger or similar to the average level of whole sets. Table XVI shows the average level both for the selected cut-off level and for whole sets. The root node (GO:0008150, GO:0005575, and GO:0003674) in each category is set to 0 and the level of child terms is increased one by one. The average level of the fungal set is 6.1500, 4.6741 and 5.2714, respectively in biological process, cellular component and molecular function, and other values in various cut-off levels are also similar to this

Table XVI. Average level of annotated GO terms by the hierarchical GO-structured model with InterPro terms.

Cut-off	Biological process	Cellular component	Molecular Function
0.70	6.4137	4.4375	5.4835
0.60	6.5652	4.6296	5.4846
0.50	6.5897	4.7500	5.4011
0.40	6.4257	4.7222	5.4308
0.30	6.2517	4.6620	5.4894
0.20	6.3404	4.7281	5.3782
All training set	6.1500	4.6741	5.2714

average or are larger. This shows the proposed model predicts both a higher-level term and a deeper-level term. Hence, annotated GO terms are mixed with both higher-level functions and deeper-level functions.

Furthermore, the percentage of predicted GO terms in each category is described (Table XVII). The GO terms which belong to biological process and molecular function are predicted at around 30% in the whole training data and those relating to cellular component are predicted at around 50%.

I performed another comparison by randomly selecting 40 proteins and comparing the annotations derived from four annotation systems which employ different methods and data sources. AAPFC [42] defines the function by the InterPro terms and GOtcha [9] also annotated by the sequence similarity, and InterPro2GO is a manually curated functional mapping table between InterPro terms and GO terms. The

Table XVII. Number of trained GO terms in three GO categories.

	Total terms	Trained terms	Percentage trained
Biological Process	1481	519	35%
Cellular Component	459	205	45%
Molecular Function	1216	389	32%
Total	3156	1113	

selected probability cut-off value in Gotcha ¹ is 0.50 and in AAPFC is 0.30. The average F-measure value for the prediction in AAPFC, Gotcha and InterPro2GO is 0.19, 0.10 and 0.04 respectively, but in the new suggested hierarchical GO annotation it is 0.48. Thus, the hierarchical approach outperformed the other methods. Table XVIII shows the F-measure values of 10 of the 40 test proteins.

C. Multiple Features with a Multi-layered Classifier

1. Data Set

In the previous section IV.B, the algorithm which is the hierarchical GO-structured model with protein domain is developed. The approach needs one or more one InterPro terms in the training proteins. The proteins which have at least one InterPro terms and at least one GO terms are 6711, while fungal proteins which have at least one GO term are 8208, thus, around 1500 proteins can not be used in the training model. Hence, another new approach is claimed in order to overcome this drawback

¹<http://www.compbio.dundee.ac.uk/gotcha/gotcha.php>

Table XVIII. F-measure comparison of four different gene function annotation systems.

Protein name	GO Hier.	AAPFC	InterPro2GO	GOtcha
ASSY_YEAST	0.8889	0.2667	0.3333	0.0000
COAC_YEAST	0.4000	0.4286	0.0000	0.0000
EFTU_YEAST	0.6000	0.0952	0.5714	0.0000
GPA1_SCHPO	0.1667	0.2222	0.0000	0.2000
KAPA_YEAST	0.3243	0.0870	0.0000	0.1481
MCFS2_YEAST	0.8333	0.2352	0.0000	0.0000
NCS1_SCHPO	0.5714	0.0800	0.1429	0.0000
NSE1_SCHPO	0.9565	0.6667	0.0000	0.2307
RRP45_YEAST	0.6777	0.8571	0.0000	0.3333
TAF14_YEAST	0.3636	0.2790	0.0000	0.2307

with multi-features. In the previous experiment which is employed only InterPro terms or multi-feature sets, multi-feature learning models are outperformed. Hence, this approach also provides higher performance. The basic training scheme uses a Bayesian probabilistic model, which measures the probability given the conditional events, where the conditional events are the binary format in the feature list. In the previous chapter (PoGO), four feature sets is employed, which are InterPro terms, BLAST, protein structure information and bio-chemical properties. Among these four feature sets, all except bio-chemical information can be converted to the binary format. The bio-chemical property is a numeric value, so this feature sets are excluded in the data. Originally, InterPro terms demonstrate the binary data and BLAST

features can be indicated in a true or false format instead of as a cumulative counted value. Protein structure information can also be applied as a binary data from the SCOP database instead of as a score value. The treated data is composed of 8208 proteins and 3339 InterPro terms, 3182 BLAST and 8494 protein structure information. In addition, the previous experiment which is a single layer classification with bio-chemical information, illustrates that the annotation of gene function not much influence by this attribute. Hence, I extract this feature set. From the GO classifier point of view, similar to the previous hierarchical GO-structured model, all parent GO terms also include the training data sets. Thus, the number of GO terms that can be trained are increased to 4706 from 3182.

2. Experiment

a. Training Procedure

The basic training process with multi-feature data sets is based on the multi-layered system described in section III. Briefly, the multi-layered classifier is made up of base-classifiers in each feature set and a meta-classifier. The base-classifier servers to build the meta-data for the training set and the meta-classifier combines the meta-data. The multi-layered learning scheme enables us to merge the different feature sets, resulting in improving of the performance. The difference between PoGO and this suggestion is a base-classifier learning scheme by the hierarchical model rather than an independent learning SVM with features and instances selection. In addition, the proteins which do not have InterPro terms are also used in the training set. The experiment is performed by 10-fold cross validation.

The base-classifier in each feature set is the probabilistic model of Naïve bayes. Since the data format is binary and each feature list is independent, the Bayesian

probabilities are easily calculated. As in the learning scheme described in the previous chapter, the training model is composed of a Bayesian probabilistic matrix. Hence, the whole training models are $3339*4706*2$, $3182*4706*2$, $8492*4706*2$ for the InterPro terms, BLAST and protein structure information, respectively.

In the InterPro meta-data sets, two subsets from the InterPro training sets are separated randomly. One subset is used for the training and the other set is used for the testing and vice versa, resulting in creating the meta-data. When two subsets are tested independently, the meta-data is assigned true if the true probability is larger than false, hence it is a binary formed column. Given this algorithm, other two more feature sets are also trained. Finally, $7428*3$ probabilistic matrices are obtained, where 7428 is the number of training proteins in each validation, because the meta-data set is composed of the test results of the other subset's training model. With this meta-data, the meta-classifier is learned by the Naïve bayes. The three binary column sets and the designated GO classifier can be computed the by the Bayesian probability, resulting in it being constituted by $3*4706*2$ Bayesian probabilistic matrices for the meta-classifier.

b. Testing Procedure

The basic formula for the base classifier is the same as formula 4.1, where three feature sets are tested separately in a base-classifier. Since I have three feature sets, the base classifier in each GO term is represented by X_v^j , where j is one of the InterPro terms, BLAST and protein structure. If $P(X_1^j, X_2^j \cdots, X_{v=T}^j)$ is larger than $P(X_1^j, X_2^j \cdots, X_{v=F}^j)$, the v^j is assigned true, otherwise false, this is annotated as M_v^j . Therefore, M_v^j is the meta-data executed by the base-classifier. After that, a meta-classifier using this meta-data is accomplished for the purpose of the integration of heterogeneous data. The test process with the meta-classifier is accomplished by

$\prod_{j=1}^3 (G_{v=T,F} | M_v^j)$ in terms of GO terms v , since each feature is independent and the treated feature sets are three.

$$P(X_1^j, X_2^j \dots, X_v^j) = \prod_{i=1}^v P(X_{i \in \{T,F\}}^j | Par(X_{i \in \{T,F\}}^j)) \quad (4.4)$$

X_v^j are GO terms in the GO structure with the feature sets, Par means parent terms and j is features - InterPro terms, BLAST and protein structure.

$$P(X_v^j) = P(G_v^j | I_1, \dots, I_k) = \frac{P(G_v^j) P(I_1, \dots, I_k | G_v^j)}{Z} \quad (4.5)$$

I_k are feature lists which the tested protein has according to feature j such as IPR_1, IPR_2 . Z is the normalized constant value. The likelihood probability in 4.5 can be simplified as in 4.6, since I_i is independent of $I_j (j \neq i)$.

$$P(I_1, \dots, I_k | G_v^j) = \prod_{i=1}^k P(I_i | G_v^j) \quad (4.6)$$

Given three M_v^j data, the meta-classifier, $\prod_{j=1}^3 (G_{v=T,F} | M_v^j)$, can be applied. If $\prod_{j=1}^3 (G_{v=T} | M_v^j)$ is larger than $\prod_{j=1}^3 (G_{v=F} | M_v^j)$, v is finally assigned as a candidate term.

As follows, when I assume that GO_1 is assigned true in Fig. 12, GO_2 is calculated by $P(GO_{1=T}, GO_2^I)$ in the InterPro terms, where $P(GO_{1=T}, GO_2^I)$ is $\prod_{i=1}^l P(I_i | GO_2) P(GO_{1=T} | GO_2)$, where l is the InterPro terms the protein has. If $P(GO_{1=T}, GO_{2=T}^I)$ is larger than $P(GO_{1=T}, GO_{2=F}^I)$, the M_2^I is considered true. M_2^B and M_2^F are the meta-data from the BLAST and protein structure information. These sets are also treated as the same approach which is described for InterPro term. The three decision labels (M_2^I, M_2^B, M_2^F) from the base-classifier are tested by the meta-classifier.

Algorithm Predict_GO{

$G \leftarrow$ GO Structure;

$I \leftarrow$ InterPro terms, $B \leftarrow$ BLAST, $F \leftarrow$ structure;

$v \leftarrow G_{top}$;

Calculation_GO(v, I, B, F)

Filter C ; }

Algorithm Calculation_GO(v, I, B, F) {

$P(v) \leftarrow$ set $T_{prob}(v, I) \ \& \ F_{prob}(v, I) \ \& \ T_{prob}(v, B) \ \& \ F_{prob}(v, B) \ \& \ T_{prob}(v, F) \ \& \ F_{prob}(v, F)$;

if $T_{prob}(P(v, I)) > F_{prob}(P(v, I))$ $M_I = T$ else $M_I = F$

if $T_{prob}(P(v, B)) > F_{prob}(P(v, B))$ $M_B = T$ else $M_B = F$

if $T_{prob}(P(v, F)) > F_{prob}(P(v, F))$ $M_F = T$ else $M_F = F$

if $T_{prob}(P(v, M_I, M_B, M_F)) > F_{prob}(P(v, M_I, M_B, M_F))$ {

$C(v) \leftarrow$ add v , where C is the candidate list; }

for each child vertex v' of v in G do {

Construct_Bayesian_Network(v', G, I, B, F, P); }

Return C ;} }

Fig. 18. Inference algorithm to annotate GO terms with the hierarchical GO-structured model with multi-features.

In the meta-learner, overall true or false probability is calculated by the product of all probability. If $P(GO_{2=T}|M_2^I) * P(GO_{2=T}|M_2^B) * P(GO_{2=T}|M_2^F)$ is larger than $P(GO_{2=F}|M_2^I) * P(GO_{2=F}|M_2^B) * P(GO_{2=F}|M_2^F)$, then GO_2 is considered as a candidate term, otherwise GO_2 is discarded. The overall algorithm is described in Fig. 18.

3. Results

In this results section, I compare the performance from two points of views. The first comparison is the hierarchical GO-structured model with only InterPro terms and multiple features, which allows us to figure out the effect of the different feature sets. First of all, the experiment is performed on the shared GO terms. The shared GO terms in both applications are 967. The overall averages in sensitivity, precision and F-measure are 0.3147, 0.5323, 0.3537 in the model with InterPro terms, but multi-feature models are 0.3468, 0.5615, 0.3749. The multi-feature models are slightly better than a individual single learner with InterPro terms. Table XIX shows 12 shared GO terms from the 967 GO terms.

Fig. 19 shows the number of GO terms and the number of annotated proteins at each cut-off F-measure value, where the F-measure is calculated by the classifier-based approach. The training models with the multi-features have more GO terms in each range, especially, in the low cut-off level (Fig. 19 (a)). In the 0.2 cut-off value, the model with InterPro terms has 725 terms, but model with multi-features has 829 terms. As the number of GO terms are increased, the related annotated proteins are also increased (Fig. 19 (b)). Fig. 20 summarizes the average F-measure. For a ranges except 0.4 and 0.5, the overall value is similar or slightly better in model with multi-feature sets.

The second comparison is multiple feature learning model with the hierarchical GO-structured model or without it (PoGO). This comparison shows the meaning

Table XIX. Classifier-based performance comparison in shared GO terms with the hierarchical GO-structured model using InterPro terms or multi-features.

GO terms	GO Hier. with InterPro			GO Hier. with multi-feature set.		
	Sensitivity	Precision	F-Measure	Sensitivity	Precision	F-Measure
GO:0000155	0.5000	1.0000	0.6667	0.6667	1.0000	0.8000
GO:0003755	0.2105	0.3636	0.2667	0.5714	0.3636	0.4444
GO:0004772	0.7500	1.0000	0.8571	0.7500	1.0000	0.8571
GO:0005384	0.3333	0.1667	0.2222	0.5000	0.1667	0.2500
GO:0005515	0.3323	0.4847	0.3942	0.3714	0.6303	0.4674
GO:0005666	0.3478	0.4000	0.3721	0.3704	0.4348	0.4000
GO:0005736	0.4348	0.4762	0.4545	0.3529	0.5714	0.4364
GO:0005884	0.4000	0.5714	0.4706	1.0000	0.5714	0.7273
GO:0006506	0.3333	0.2778	0.3030	0.7000	0.3500	0.4667
GO:0006513	0.2143	0.1579	0.1818	0.4375	0.3500	0.3889
GO:0030600	0.6667	0.5714	0.6154	0.7000	1.0000	0.8235
GO:0051017	0.1667	0.5000	0.2500	0.4000	1.0000	0.5714
:	:	:	:	:	:	:
Average	0.3147	0.5323	0.3537	0.3468	0.5615	0.3749

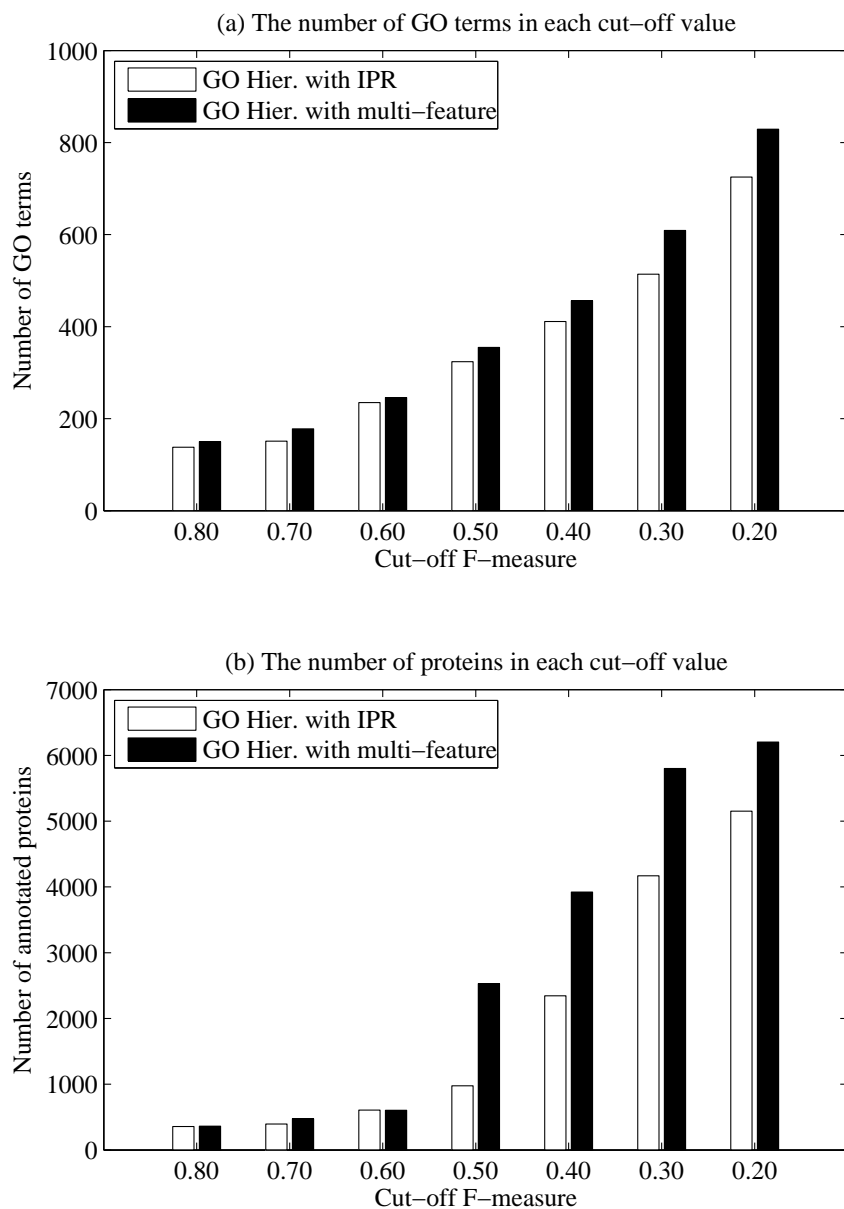


Fig. 19. The hierarchical GO-structured model with InterPro terms or multi-features: (a) The number of GO terms at each cut-off F-measure value. (b) The number of annotated proteins at each cut-off F-measure value.

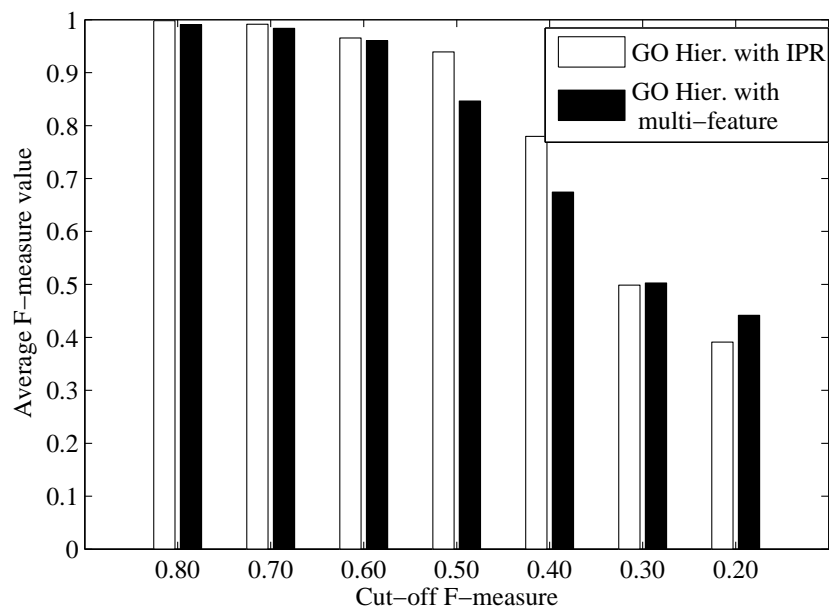


Fig. 20. Average F-measure at each cut-off F-measure value in the hierarchical GO-structured model with InterPro terms and multi-features.

of the hierarchical GO structure in the multi-feature sets. In section IV.B, the hierarchical GO-structured model with InterPro terms outperformed that without an embedding GO structure, thus GO-structured model with multi-features is also better than others (Fig. 21, Fig. 22). Fig. 21 shows the number of GO terms at each cut-off F-measure value. The multi-feature sets have more GO terms in each range. The hierarchical model has many more GO terms, because PoGO is trained to only 444 classifiers, but the hierarchical GO structure with multi-features is trained to all fungi GO terms.

Based on these GO terms, the number of annotated proteins is described (Fig. 22 (a)). However, unlike the previous result, without embedding the hierarchical information has more proteins in each range. The reason is that two GO terms (GO:0005515 : protein binding, GO:00058209 : cytosol) are annotated in many proteins.

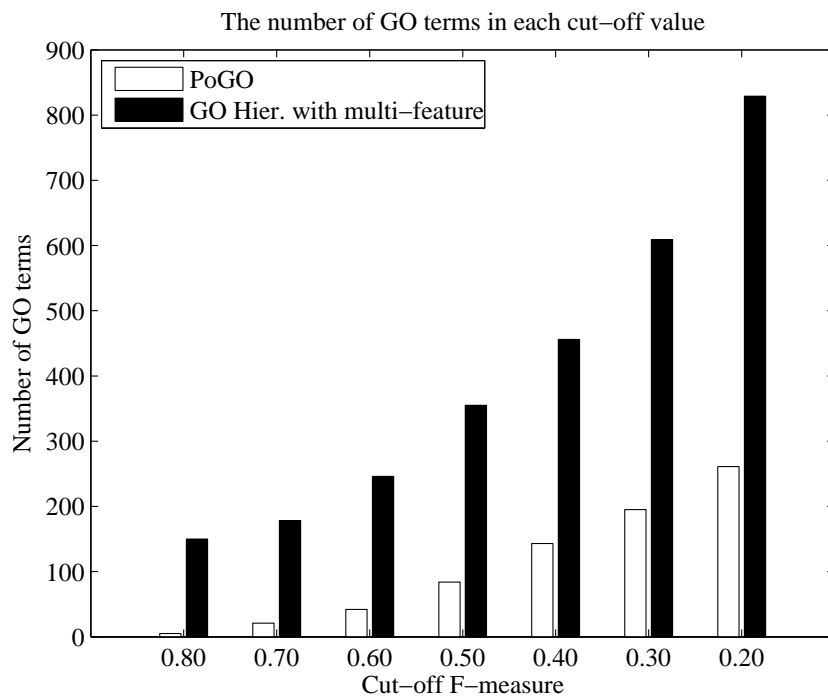


Fig. 21. Number of GO terms at each cut-off F-measure value in PoGO and the hierarchical GO-structured model with multi-features.

Among 8208 proteins, 2072 proteins have `GO:0005515` and 1650 proteins have `GO:0005829`, that is, the number of annotated proteins in each range (Fig. 22 (a)) depends on these two terms. If two terms have a high F-measure value in terms of a classifier-based approach, the number of annotated proteins are also effected in a lower cut-off F-measure. In PoGO classifier, the classifier-based F-measure value in `GO:0005515` is 0.7166. and that of `GO:0005829` is 0.5723. However, model with an embedding GO structure has values of 0.4674 and 0.3903, respectively. If I excluded these two terms, I have another figure (Fig. 22 (b)).

During this process, the average protein-based F-measure is also compared in each range shown in (Fig. 23(a)), where the used proteins are Fig. 22(a). The average

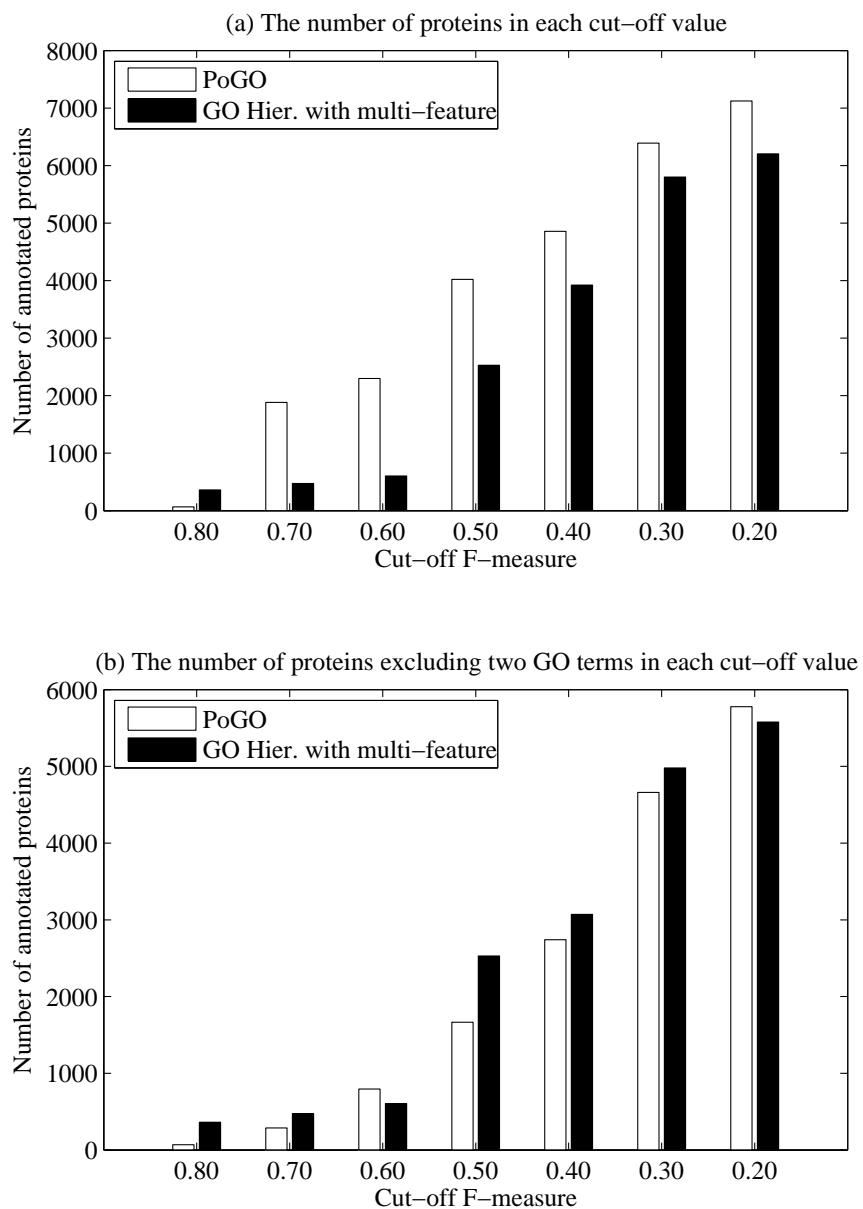


Fig. 22. PoGO and the hierarchical GO-structured model with multi-features set: (a) The number of annotated proteins at each cut-off F-measure value. (b) The number of annotated proteins excluding the two highly annotated GO terms.

F-measure with the reduced proteins (Fig. 22(b)), which excluded the two highly annotated GO terms, is described in Fig. 23(b). In both comparisons, a GO hierarchical structure with multi-features outperforms that without an embedding structure.

However, model without hierarchical structure also provides a good enough performance. This stand for if there are enough proteins to build the training set, even though hierarchical GO structure is not embedded, the learning scheme provides outperformed result. To elucidate this statement, I also compared the shared GO terms both in PoGO and in the hierarchical information with multi-features. In most cases, performance matrices in PoGO have outperformed those with an embedding GO structure. The overall average for sensitivity, precision and F-measure in PoGO are 0.2433, 0.6339, 0.3127, while the hierarchical GO with multi-features are 0.2418, 0.3533, 0.2476. In Table XX, 12 GO terms among 411 shared GO terms are shown.

From this result, I know that if GO terms have enough positive proteins, i.e., classifier with many annotated proteins, then a meta-classifier without an embedding GO structure also provides good performance. However, most of the GO terms in the fungi set are very sparse and rare, thus a model applied with the GO hierarchical structure is more reasonable in order to assign gene functions.

D. Implementation

The calculation of training models by computing the Bayesian probability were performed with MATLAB [59]. Other script for parsing the Uniprot, making the data set, multiplying the Bayesian probability and and filtering process used php.

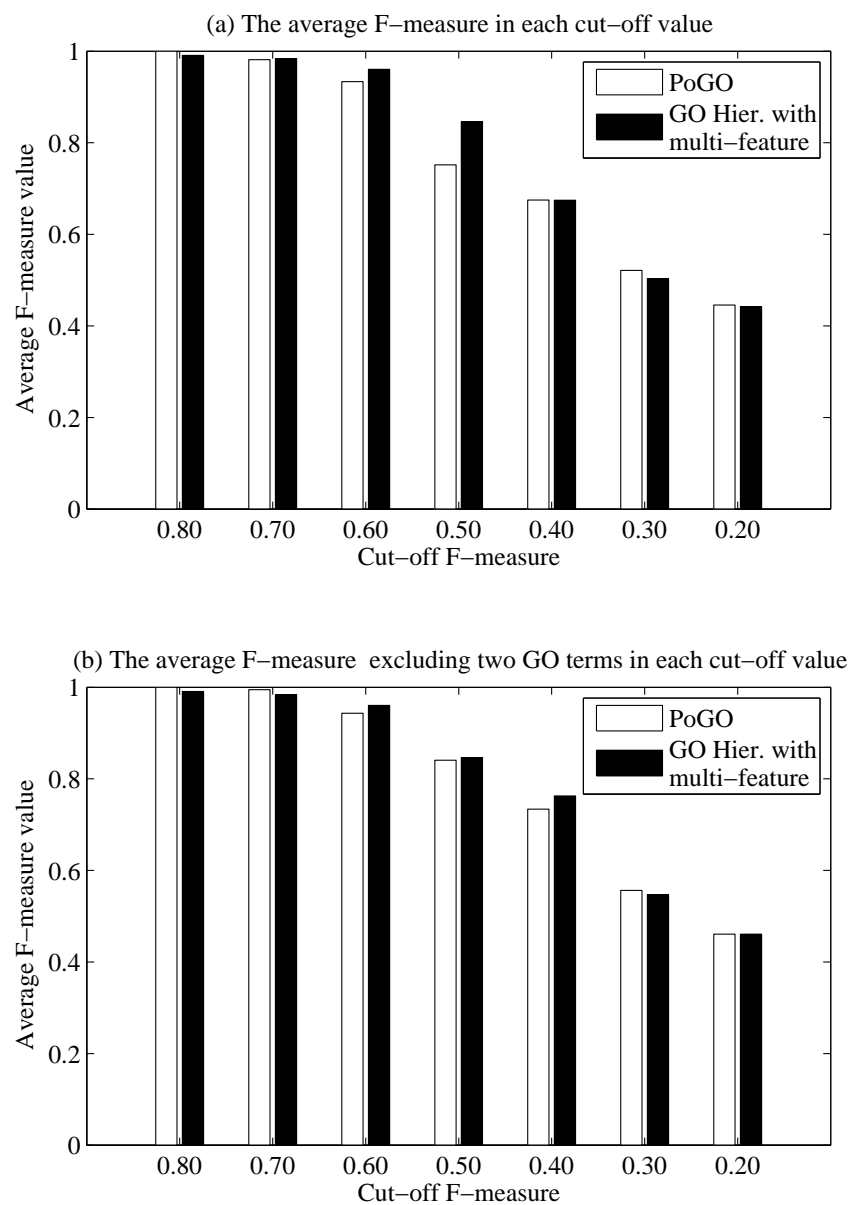


Fig. 23. PoGO and the hierarchical GO-structured model with multi-features set: (a) The average F-measure at each cut-off F-measure value. (b) The average F-measure excluding the two highly annotated GO terms.

Table XX. Classifier-based performance comparison in shared GO terms with PoGO and the hierarchical GO-structured model using multi-features set.

GO terms	PoGO			GO Hier. with multi-feature		
	Sensitivity	Precision	F-Measure	Sensitivity	Precision	F-Measure
GO:0000105	0.5862	1.0000	0.7391	0.6667	0.7059	0.6857
GO:0005743	0.2093	0.2769	0.2384	0.2077	0.3462	0.2596
GO:0005758	0.4043	0.6333	0.4935	0.3846	0.1667	0.2326
GO:0005773	0.1587	0.3704	0.2222	0.0680	0.2593	0.1077
GO:0005938	0.1504	0.7727	0.2519	0.0451	0.5000	0.0827
GO:0006122	0.5000	0.7273	0.5926	0.3000	0.2727	0.2857
GO:0006400	0.7059	0.8571	0.7742	0.3333	0.3929	0.3607
GO:0006487	0.5758	0.5938	0.5846	0.4118	0.4375	0.4242
GO:0006865	0.3095	0.8667	0.4561	0.3182	0.8235	0.4590
GO:0016192	0.3500	0.7241	0.4719	0.0856	0.6552	0.1514
GO:0042493	0.2204	0.6273	0.3262	0.1391	0.1455	0.1422
GO:0045045	0.1667	0.5417	0.2549	0.1404	0.6667	0.2319
:	:	:	:	:	:	:
Average	0.2433	0.6339	0.3127	0.2418	0.3533	0.2476

E. Summary

In the previous two chapters, independent GO classifiers are trained, but a GO structure form (DAG), which points out that a parent's term has a relationship with the child term, is not considered in the training scheme. Thus, I propose two methods for assigning GO terms to proteins using InterPro terms and multi-features with Bayesian network learning in order to embed GO hierarchical structure. The Bayesian network frame is a graph based model demonstrating the Bayesian probabilistic relationship between random variables.

Many studies have also used this approach, but usually employ it for the purpose of integrating heterogeneous data. However, I use the GO structure for GO structural properties by constructing GO structure in each category. In addition, through the filtering step, the false positive errors are removed from the candidate lists. This hierarchical GO-structured approach with InterPro terms provides improved performance than that with no GO structural model (AAPFC), when the F-measures of individual GO terms are compared, but more importantly, it enables us to include more GO terms in the classifier, which in turn allows many more proteins to be annotated. In addition, this approach satisfies the consistency of prediction, i.e., it does not predict only high-level (parent) GO terms nor only deeper-level (child) GO terms. If this consistency does not meet, the predicted function is located in the high-level.

Afterwards, a combination model which associates with multi-feature sets and the hierarchical model with multi-layered schemes is built. In the base-classifier, three feature sets are predicted by the Bayesian network with GO hierarchical structure and overall, two probabilities given true or false GO terms are calculated in the meta-learner. The hierarchical GO-structured model with multi-feature outperforms that with InterPro terms. The multi-feature learning model also contributes more

GO terms and more annotated proteins than without the hierarchical modeling with multi-feature sets (PoGO), but the overall F-measure in the shared GO terms is less than in PoGO. Given this result, I analyze that if GO terms are annotated in enough proteins, the modeling without the hierarchical structure is also well-fitted for annotation. However, most GO terms are so sparse that the hierarchical GO-structured model is needed for the gene functional annotation.

CHAPTER V

CONCLUSION

A. Summary

Automated gene function annotation is an important issue for biologists, since the development of high-throughput genome sequencing and gene annotation methods. Traditionally, protein function is expressed as free text descriptions, but recently controlled vocabularies of various types have been employed. The Gene Ontology (GO) provides a controlled vocabulary of terms for annotating proteins. Methods for the automated annotation of gene function often depend on sequence similarity, but this assumes that homology has a similar function. In this dissertation, I have studied three methods for assigning the function automatically, using pattern recognition and statistical techniques with various feature sets.

In AAPFC, I developed an application for assigning GO terms to proteins using InterPro terms as features and learning an independent SVM for each GO term. This approach enables the sparse and imbalanced nature of the data set to be overcome by dimension reduction. The training model has high accuracy compared to InterPro2GO which is a manually curated mapping of InterPro terms to GO terms that is maintained by the InterPro consortium. In addition, it has been shown that the taxon-specific models outperform the non-specific models. However, this approach only utilizes one source of features. Therefore, in the next work, I utilize a multi-layered classifier as an approach in order to add more features to the learning method.

More features sets including sequence similarity, micro-array data, and chemical information can be added in PoGO. Furthermore, an integrated system for combining heterogeneous features are proposed. A model that utilizes multiple feature types

overcomes the shortcomings of AAPFC. On the other hand, the first two approaches are for training independent classifiers for each GO term and only a limited number of terms can be classified as there is not enough data in the training set. In addition, much information is lost by dimension reduction. The next approach overcomes these problems.

The third approach considers the hierarchical nature of the GO, since GO contains DAG structure. With the Bayesian network, The hierarchical GO-structured model with InterPro term and a multi-feature are implemented. This approach provides improved performance when the F-measure of individual GO terms are compared, but more importantly, it allows for the inclusion of more GO terms in the classifier which in turn enables many more proteins to be annotated. This is because the restricted GO terms which are annotated 10 or more proteins are trained in the AAPFC and PoGO. However, when the shared GO terms in PoGO and the proposed method are compared, the overall F-measure value is slightly higher in PoGO. From this result, it is apparent that a multi-layered classifier without the GO structure will be well trained if each GO classifier has enough training model. However, most GO terms are very sparse. The combination of a system which merges a multi-layered learning model with multi-features for the many annotated GO terms, and the hierarchical structure scheme in a sparse data set, provide a rich assigning term. One shortcoming of this approach is that the training model is very sensitive to positive numbers, because the data set is very sparse. In particular, when a GO term is annotated by only one or two proteins, the Bayesian probability in the training model varies in whether this protein belongs to the test set or training set. Hence, the future plan is measuring the average training value in each GO classifier using 10-fold cross validation to provide a more robust training model.

B. Future Work

For multi-feature sets, the InterPro terms, BLAST, bio-chemical information and protein structure were employed. In addition to these feature sets, other attributes such as Uniprot keyword, are available. Uniprot keyword is also a controlled vocabulary in the UniProt /Swiss-Prot entries. There are 10 categories of keyword and currently there are all lists are 926. Each protein may have more than one keyword. This feature can be added PoGO, because the data property is also sparse but the feature lists are not large like InterPro terms. The attribute of Uniprot keyword is also contribute to the assignation of the gene function. Similar to the InterPro terms, the GO consortium supports the keyword mapping table, which also enables the comparison of performance.

For the extension of annotation, the database pre-computed information about fungal protein cluster (FPC-DB) will be provided. Many databases are built to search for a protein properties including protein domain, sequence similarity and so on. However, the purpose of this database is used for the source of features for learning protein function classifiers as well as a mechanism for making protein annotations available to researchers. The provided data in the database is species, GO terms, InterPro terms, gene family and multiple sequence alignment of each family. Using the automatic assignment of protein functional tools including AAPFC, PoGO and the hierarchical model with multi-feature sets, GO terms are annotated for unknown fungal proteins. Other information such as alignments or InterPro terms are earned by several bioinformatical applications. These pre-computed data sets are presented for each family.

In addition, the sequence similarity is analyzed using the GO-structured information. This approach measures the distance of GO terms in two pair wise proteins

and finds the most similar proteins. The distance is calculated by maximum level in the GO structure. The result can be compared to the classical sequence similarity methods such as BLAST, and provides verification of annotated GO terms.

REFERENCES

- [1] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, “Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium.” *Nature Genetics*, vol. 25, no. 1, pp. 25–29, 2000.
- [2] E. Camon, M. Magrane, D. Barrell, D. Binns, W. Fleischmann, P. Kersey, N. Mulder, T. Oinn, J. Maslen, A. Cox, and R. Apweiler, “The Gene Ontology Annotation (GOA) Project: implementation of GO in SWISS-PROT, TrEMBL, and InterPro,” *Genome Research*, vol. 13, no. 4, pp. 662–672, 2003.
- [3] E. Camon, M. Magrane, D. Barrell, V. Lee, E. Dimmer, J. Maslen, D. Binns, N. Harte, R. Lopez, and R. Apweiler, “The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology,” *Nucleic Acids Research*, vol. 32, pp. D262–266, 2004.
- [4] I. Friedberg, “Automated protein function prediction—the genomic challenge,” *Brief Bioinformatics*, vol. 7, no. 3, pp. 225–242, 2006.
- [5] G. Pandey, V. Kumar, and M. Steinbach, “Computational approaches for protein function prediction,” Department of Computer Science and Engineering, University of Minnesota, Twin Cities, Tech. Rep. TR 06-028, 2006.
- [6] Z. Günther, “Ontoblast function: From sequence similarities directly to potential functional annotations by ontology terms,” *Nucleic Acids Research*, vol. 31, no. 13, pp. 3799–3803, 2003.

- [7] S. Hennig, D. Groth, and H. Lehrach, “Automated gene ontology annotation for anonymous sequence data,” *Nucleic Acids Research*, vol. 31, no. 13, pp. 3712–3715, 2003.
- [8] S. Khan, G. Situ, K. Decker, and C. J. Schmidt, “Gofigure: automated gene ontology annotation,” *Bioinformatics*, vol. 19, no. 18, pp. 2484–2485, 2003.
- [9] D. Martin, M. Berriman, and G. Barton, “GOtcha: a new method for prediction of protein function assessed by the annotation of seven genomes,” *BMC Bioinformatics*, vol. 5, no. 1, p. 178, 2004.
- [10] N. J. Mulder, R. Apweiler, T. K. Attwood, A. Bairoch, A. Bateman, D. Binns, P. Bork, V. Buillard, L. Cerutti, R. Copley, E. Courcelle, U. Das, L. Daugherty, M. Dibley, R. Finn, W. Fleischmann, J. Gough, D. Haft, N. Hulo, S. Hunter, D. Kahn, A. Kanapin, A. Kejariwal, A. Labarga, P. S. Langendijk-Genevaux, D. Lonsdale, R. Lopez, I. Letunic, M. Madera, J. Maslen, C. McAnulla, J. McDowall, J. Mistry, A. Mitchell, A. N. Nikolskaya, S. Orchard, C. Orengo, R. Petryszak, J. D. Selengut, C. J. A. Sigrist, P. D. Thomas, F. Valentin, D. Wilson, C. H. Wu, and C. Yeats, “New developments in the interpro database,” *Nucleic Acids Research*, vol. 35, pp. D224–228, 2007.
- [11] J. Schug, S. Diskin, J. Mazzairelli, B. P. Brunk, and J. Stoeckert, Christian J., “Predicting Gene Ontology Functions from ProDom and CDD Protein Domains,” *Genome Research*, vol. 12, no. 4, pp. 648–655, 2002.
- [12] F. Corpet, F. Servant, J. Gouzy, and D. Kahn, “ProDom and ProDom-CG: tools for protein domain analysis and whole genome comparisons,” *Nucleic Acids Research*, vol. 28, no. 1, pp. 267–269, 2000.

- [13] A. Marchler-Bauer, A. R. Panchenko, B. A. Shoemaker, P. A. Thiessen, L. Y. Geer, and S. H. Bryant, “CDD: a database of conserved domain alignments with links to domain three-dimensional structure,” *Nucleic Acids Research*, vol. 30, no. 1, pp. 281–283, 2002.
- [14] E. M. Marcotte, M. Pellegrini, H.-L. Ng, D. W. Rice, T. O. Yeates, and D. Eisenberg, “Detecting protein function and protein-protein interactions from genome sequences,” *Science*, vol. 285, no. 5428, pp. 751–753, 1999.
- [15] E. M. Marcotte, M. Pellegrini, M. J. Thompson, T. O. Yeates, and D. Eisenberg, “A combined algorithm for genome-wide prediction of protein function.” *Nature*, vol. 402, no. 6757, pp. 83–86, November 1999.
- [16] M. Pellegrini, E. M. Marcotte, M. J. Thompson, D. Eisenberg, and T. O. Yeates, “Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles,” *National Academy of Sciences*, vol. 96, no. 8, pp. 4285–4288, 1999.
- [17] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, “Hierarchical multi-label prediction of gene function,” *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.
- [18] R. Eisner, B. Poulin, D. Szafron, P. Lu, and R. Greiner, “Improving protein function prediction using the hierarchical structure of the gene ontology,” in *2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, La Jolla, CA, USA, 2005, pp. 354–363.
- [19] B. Shahbaba and R. Neal, “Gene function classification using bayesian models with hierarchy-based priors,” *BMC Bioinformatics*, vol. 7, no. 1, p. 448, 2006.

- [20] X. Deng, H. Geng, and H. H. Ali, “Learning yeast gene functions from heterogeneous sources of data using hybrid weighted bayesian networks,” in *Fourth International IEEE Computer Society Computational Systems Bioinformatics Conference*, Stanford University, CA, USA, 2005, pp. 25–34.
- [21] O. G. Troyanskaya, K. Dolinski, A. B. Owen, R. B. Altman, and D. Botstein, “A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *Saccharomyces cerevisiae*),” *National Academy of Sciences*, vol. 100, no. 14, pp. 8348–8353, 2003.
- [22] A. Clare and R. D. King, “Predicting gene function in *Saccharomyces cerevisiae*,” *Bioinformatics*, vol. 19, no. S2, pp. ii42–ii49, 2003.
- [23] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, “Basic local alignment search tool,” *Jorunal Molecular Biology*, vol. 215, no. 3, pp. 403–41, 1990.
- [24] A. Vinayagam, C. del Val, F. Schubert, R. Eils, K. Glatting, S. Suhai, and R. Konig, “GOPET : a tool for automated predictions of Gene Ontology terms,” *BMC Bioinformatics*, vol. 7, p. 161, 2006.
- [25] A. Vinayagam, R. Konig, J. Moormann, F. Schubert, R. Eils, K. Glatting, and S. Suhai, “Applying support vector machine for gene ontology based gene function prediction,” *BMC Bioinformatics*, vol. 5, p. 116, 2004.
- [26] A. Al-Shahib, R. Breitling, and D. Gilbert, “Feature selection and the class imbalance problem in predict protein function form sequence,” *Applied Bioinformatics*, vol. 4, no. 3, pp. 195–203, 2005.
- [27] O. D. King, R. E. Foulger, S. S. Dwight, J. V. White, and F. P. Roth, “Predicting gene function from patterns of annotation,” *Genome Research*, vol. 13, no. 5, pp.

- 896–904, 2003.
- [28] P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy, “Gene functional classification from heterogeneous data,” in *International Conference on Research in Computational Molecular Biology*, Montréal, Québec, Canada, 2001, pp. 249–255.
- [29] P. Pavlidis, J. Weston, J. Cai, and W. S. Noble, “Learning gene functional classifications from multiple data types,” *Journal of Computational Biology*, vol. 9, no. 2, pp. 401–411, 2002.
- [30] N. Japkowicz and S. Stephen, “The class imbalance problem: a systematic study,” *Intelligent Data Analysis Journal*, vol. 6, no. 5, pp. 429–449, 2002.
- [31] M. Kubat and S. Matwin, “Addressing the curse of imbalanced training sets: one-sided selection,” in *Proc. 14th International Conference on Machine Learning*, Nashville, TN, USA, 1997, pp. 179–186.
- [32] G. M. Weiss, “Mining with rarity: a unifying framework,” *SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7–19, 2004.
- [33] C. X. Ling and C. Li, “Data mining for direct marketing: problems and solutions,” in *Fourth International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 1998, pp. 73–79.
- [34] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence and Research*, vol. 16, pp. 321–357, 2002.
- [35] J. Zhang and I. Mani, “kNN approach to unbalanced data distributions: a case study involving information extraction,” 2003.

- [36] A. Bairoch, R. Apweiler, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L.-S. L. Yeh, "The universal protein resource (UniProt)," *Nucleic Acids Research*, vol. 33, pp. D154–159, 2005.
- [37] E. Quevillon, V. Silventoinen, S. Pillai, N. Harte, N. Mulder, R. Apweiler, and R. Lopez, "Interproscan: protein domains identifier," *Nucleic Acids Research*, vol. 33, pp. W116–120, 2005.
- [38] L. Yu and H. Liu, "Feature selection for high-dimensional data: a fast correlation-based filter solution," in *Twentieth International Conference on Machine Learning*, Washington, DC, USA, 2003, pp. 856–863.
- [39] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Fourteenth International Conference on Machine Learning*, Nashville, US, 1997, pp. 412–420.
- [40] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
- [41] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
- [42] J. Jung and M. R. Thon, "Automatic annotation of protein functional class from sparse and imbalanced data sets," in *Data Mining and Bioinformatics, First International Workshop*, ser. Lecture Notes in Computer Science, vol. 4316. Seoul, Korea: Springer, 2006, pp. 65–77.
- [43] E. Falkenauer, "On method overfitting," *Journal of Heuristics*, vol. 4, no. 3, pp. 281–287, 1998.

- [44] D. P. Lewis, T. Jebara, and W. S. Noble, “Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure,” *Bioinformatics*, vol. 22, no. 22, pp. 2753–2760, 2006.
- [45] J. Huang, J. Lu, and C. X. Ling, “Comparing naive bayes, decision trees, and svm with auc and accuracy,” in *Third IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 553– 556.
- [46] D. Fan, P. Chan, and S. Stolfo, “A comparative evaluation of combiner and stacked generalization,” in *AAAI-96 Workshop on Integrating Multiple Learned models*, Portland, OR, USA, 1996, pp. 40–46.
- [47] D. H. Wolpert, “Stacked generalization,” *Neural Network*, vol. 5, no. 2, pp. 241–259, 1992.
- [48] P. K. Chan and S. J. Stolfo, “A comparative evaluation of voting and meta-learning on partitioned data,” in *International Conference on Machine Learning*, Tahoe, CA, USA, 1995, pp. 90–98.
- [49] A. K. Jain, R. P. W. Duin, and J. Mao, “Statistical pattern recognition: A review,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 4–37, 2000.
- [50] S. Džeroski and B. Ženko, “Is combining classifiers with stacking better than selecting the best one?” *Machine Learning*, vol. 54, no. 3, pp. 255–273, 2004.
- [51] P. K. Chan and S. J. Stolfo, “Experiments in multistrategy learning by meta-learning,” in *Second International Conference on Information and Knowledge Management*, Washington, DC, USA, 1993, pp. 314–323.

- [52] J. Söding, “Protein homology detection by HMM-HMM comparison,” *Bioinformatics*, p. bti125, 2004.
- [53] J. Soding, A. Biegert, and A. N. Lupas, “The HHpred interactive server for protein homology detection and structure prediction,” *Nucleic Acids Research*, vol. 33, no. suppl_2, pp. W244–248, 2005.
- [54] F. Chalmel, A. Lardenois, J. Thompson, J. Muller, J. Sahel, T. Leveillard, and O. Poch, “GOAnno: GO annotation based on multiple alignment,” *Bioinformatics*, vol. 21, no. 9, pp. 2095–2096, 2005.
- [55] L. I. Rice, Peter and A. Bleasby, “EMBOSS: The European Molecular Biology Open Software Suite ,” *Trends in Genetics*, vol. 16, no. 6, pp. 276–277, 2000.
- [56] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, “Gapped blast and psi-blast: a new generation of protein database search programs,” *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.
- [57] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, “SCOP: a structural classification of proteins database for the investigation of sequences and structures,” *Journal of Molecular Biology*, vol. 247, pp. 536–540, 1995.
- [58] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” in *European Conference on Computational Learning Theory*, Jerusalem, Israel, 1995, pp. 23–37.
- [59] Matlab. [Online]. Available: <http://www.mathworks.com/>
- [60] Pattern recognition toolbox for MATLAB. [Online]. Available: <http://cmp.felk.cvut.cz/~xfrancv/stprtool/>

- [61] B. Hayete and J. Bienkowska, “Gotrees: predicting go associations from protein domain composition using decision trees,” in *Pacific Symposium Biocomputing*, Kohala Coast, HI, USA, 2005.
- [62] X.-L. Li, Y.-C. Tan, and S.-K. Ng, “Systematic gene function prediction from gene expression data by using a fuzzy nearest-cluster method,” *BMC Bioinformatics*, vol. 7, no. Suppl 4, p. S23, 2006.
- [63] D. Heckerman, D. Geiger, and D. M. Chickering, “Learning bayesian networks: The combination of knowledge and statistical data,” *Machine Learning*, vol. 20, no. 3, pp. 197–243, 1995.
- [64] N. Cesa-Bianchi, A. Conconi, and C. Gentile, “Regret bounds for hierarchical classification with linear-threshold functions,” in *17th Annual Conference on Learning Theory*, Banff, Canada, 2004, pp. 93–108.
- [65] S.-J. Cho and J. H. Kim, “Bayesian network modeling of hangul characters for on-line handwriting recognition,” in *Seventh International Conference on Document Analysis and Recognition*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 207 – 211.
- [66] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafe, A. Perez, and V. Robles, “Machine learning in bioinformatics,” *Brief Bioinformatics*, vol. 7, no. 1, pp. 86–112, 2006.
- [67] J. M. F.-L. Luis M. de Campos and J. F. Huete, “A layered bayesian network model for document retrieval,” *Lecture Notes in Computer Science*, vol. 2291, pp. 339–343, 2002.
- [68] A. Raval, Z. Ghahramani, and D. L. Wild, “A Bayesian network model for

protein fold and remote homologue recognition,” *Bioinformatics*, vol. 18, no. 6, pp. 788–801, 2002.

- [69] Y. Tamada, S. Kim, H. Bannai, S. Imoto, K. Tashiro, S. Kuhara, and S. Miyano, “Estimating gene networks from gene expression data by combining Bayesian network model with promoter element detection,” *Bioinformatics*, vol. 19, pp. ii227–236, 2003.

VITA

Jae Hee Jung received her B.S. in statistics and computer science from the Dongduk Women's University, Korea, in 2000. She received the M.S. degree in Computer Science at Korea University, Korea, in 2002. She worked at LG Electronics Company between 2002 and 2003. She was a Ph.D. student in the Department of Computer Science at Texas A&M University since September 2003, and she received her Ph.D. in August 2008. Her research interest is computational biology, bioinformatics. She can be reached at jaeheejung@tamu.edu and physical address is 540 Southwest Parkway #D, College Station, TX, 77840. The address of Department of Computer Science is Texas A&M University TAMU 3112 College Station, TX 77843

The typist for this dissertation was Jae Hee Jung.