

PROTEIN FUNCTION CLASSIFICATION WITH GENERALIZED MULTIPLE
INSTANCE LEARNING

by

Soon-Myung Jason Lee

A THESIS

Presented to the Faculty of
The Graduate College at the University of Nebraska
In Partial Fulfillment of Requirements
For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Stephen Scott

Lincoln, Nebraska

August, 2004

PROTEIN FUNCTION CLASSIFICATION WITH GENERALIZED MULTIPLE INSTANCE LEARNING

Soon-Myung Jason Lee, M.S.

University of Nebraska, 2004

Advisor: Stephen Scott

We apply the generalized multiple-instance learning (GMIL) model and its learning algorithms, GMIL-2, and recently developed kernel-based GMIL, to the protein function classification problem. With the advancement in DNA sequencing techniques, new putative proteins are added to databases much faster rate than they can be tested to determine the function. Thus a fundamental challenge in computational biology is to develop an efficient and accurate algorithm to classify the putative proteins into the family of proteins with known functions. For this task, protein sequence similarity or profiling methods such as BLAST and HMMER have dominated the field with good success. Unfortunately these popular methods perform poorly when the putative protein does not share any or shares low sequence similarities with other known proteins. In order to avoid using sequence similarities for this problem, we use structural properties of amino acids to build our GMIL classifiers. Multiple-instance learning is a powerful modeling system, which has been extensively applied in variety of fields, such as robot vision, content-based image retrieval, and drug discovery, with great success. GMIL expanded the area of application by generalizing the way MIL labels its *bags*. In genome scale protein function classification test, our method performed competitively with the algorithm DMP. In the second test using GPCR class A protein family, GMIL-2 and kernel-based GMIL performed very well.

DEDICATION

I dedicate this thesis to my parents for their endless love and support.

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor, Dr. Stephen Scott for his guidance and support for this research. I could not have asked for a better role model to follow as a researcher and as a person. I'm also deeply appreciative to Dr. Jitender S. Deogun and Dr. Etsuko Moriyama for their encouraging words and for taking me under their wings as I began my career in the field of bioinformatics.

Many thanks to Qingping Tao for the countless help he landed me for the research, and to my dear sister Lisa Lee for believing in me. My appreciation is also extended to Alicia Cho and her family for all their support.

I would like to take this moment to acknowledge the financial support from National Science Foundation (NSF) grant EPS-0091900.

Contents

1	Introduction	1
1.1	Motivation and Purpose	1
1.2	Thesis Organization	4
2	Background and Related Work	5
2.1	Machine Learning Approaches	5
2.2	Multiple Instance Learning and Its Derivatives	6
2.2.1	Generalized Multiple Instance Learning	6
2.2.2	GMIL-1	7
2.2.3	GMIL-2	9
2.2.4	Kernel-Based GMIL	11
2.3	<i>E. coli</i> Genome	12
2.4	GPCR Protein Superfamily	13
2.5	Other Protein Function Classification Methods	14
2.5.1	Protein Function Classification Problem	14
2.5.2	Property-Based Sequence Analysis	14
2.5.3	Data Mining Prediction (DMP)	16
2.5.4	GPCR Classification with SVM	18
3	Applying Generalized Multiple Instance Learning to Protein Function Classification	20
3.1	Data Preparation	20
3.1.1	Multiple Alignment Dependency	21
3.1.2	Grouping of Functional Classes	22
3.2	Train and Test Data Sets	25
3.2.1	Functional Hierarchies	25
3.2.2	<i>E. coli</i> Train and Test Data Sets	26
3.2.3	GPCR Class A Train and Test Data Sets	27
4	Experimental Results	30
4.1	Results of GMIL-2 on <i>E. coli</i>	30
4.2	Results of GMIL-2 on GPCR Class A Data Sets	37

5	Conclusions and Future Development	41
	Bibliography	44
A	Groupings of <i>E. coli</i> Functional Classes Tables	48
B	<i>E. coli</i> Functional Hierarchy Tables	51
C	GPCR Functional Hierarchy Tables	54
D	Distribution of GPCR test sequences and kernel-based GMIL results	57

List of Figures

2.1	An illustration of a rectilinear model space partitioning, and grouping of boxes. Points labeled 1 to 6 are the instances.	8
2.2	An illustration of GMIL-2 box grouping algorithm. B1 and B2 boxes are grouped into dashed-lined box formed by instances 1, 2, and 3. . .	10
2.3	An example of how collinear instances are added.	10
2.4	An example sliding window recognizer with $s = 4$, $w_i = 1$ for all i . . .	15
3.1	(a) Before sequence alignment: Positive bags (rectangles) are not aligned. (b) After sequence alignment: Positive bags are aligned with one another.	22
3.2	An example of a bad grouping scheme where the lengths of the two functional classes are too different. In such cases, MIL could learn the lengths of classes as classifiers.	23
3.3	An example of stretching out shorter functional class.	24
4.1	Accuracy vs. coverage graph on level 1 for GMIL-2(100) and GMIL-2(97). DMP result is also shown.	33
4.2	Accuracy vs. coverage graphs on level 2 for GMIL-2(100) and GMIL-2(97). DMP result is also shown.	34
4.3	Accuracy vs. coverage graphs on level 3 for GMIL-2(100) and GMIL-2(97). DMP result is also shown.	35
4.4	A histogram that shows the distribution of confidence ratings	36

List of Tables

4.1	The test set results (the numbers 1, 2, and 3 correspond to the levels in <i>E. coli</i> functional hierarchy).	31
4.2	GPCR Results from GMIL-2 and kernel-based GMIL.	38
4.3	Improvements for GMIL-2 in training group 5 (Peptide) accuracy via increase in number of instances.	39
A.1	Groupings of functional classes	49
A.2	Continued: Groupings of functional classes	50
B.1	3 levels of functional hierarchy in <i>E. coli</i>	52
B.2	Continued: 3 levels of functional hierarchy in <i>E. coli</i>	53
C.1	Functional classes in level 2 of GPCR functional hierarchy.	55
C.2	Continued: Functional classes in level 2 of GPCR functional hierarchy.	56
D.1	Olfactory test label sizes.	58
D.2	Amine test label sizes.	58
D.3	Class A orphan other test label sizes.	58
D.4	Gonadotropin releasing hormone mammals test label sizes.	59
D.5	Hormone protein test label sizes.	59
D.6	Leukotriene B4 receptor test label sizes.	59
D.7	Nucleotide like test label sizes.	59
D.8	Peptide test label sizes.	60
D.9	Prostanoid test label sizes.	60
D.10	Rhodopsin test label sizes.	61
D.11	Thyrotropin releasing hormone secretagogue test label sizes.	61

Chapter 1

Introduction

1.1 Motivation and Purpose

Protein function classification is a key aspect of genome annotation that mainly relies on detecting near or remote homology by sequence similarity. Over the years, numerous sequence similarity detecting methods, such as BLAST and FASTA, have been developed and gained popularity due to their high success rates. When no clear homologs can be detected, however, these methods tend to perform poorly. Thus alternative methods that do not rely solely on sequence similarities or sequence profiles are required for classifying proteins with low sequence conservations.

Recently, Kim et al. [14] developed a G protein-coupled receptor (GPCR) protein classification algorithm that is completely independent of sequence similarities or profiles. Kim et al. utilized various amino acid property values, such as solubility, polarity, and molecular weight to determine the classification. Their algorithm then computed summary statistics of these numeric sequences and inferred a linear discriminant function to separate GPCRs from non-GPCRs. Their results were outstanding for GPCR classification problem, indicating that the usage of property

values for classification was an excellent alternative when sequence similarities fail.

Classifying proteins that were previously classified into further detailed functional classes requires more detailed modeling of a protein family than a conventional binary classification problem. Unfortunately, mapping the sequences to their summary statistics as Kim et al. does loses significant information that reveals fundamental properties of the proteins. This is where multiple instance learning (MIL) algorithms shine through because these algorithms are capable of scaling up to multiple instances, therefore avoiding the loss of information. For this, we apply a machine learning model called generalized multiple-instance learning (GMIL) to model the protein functional classes. Then we learn and predict functional classes by using GMIL algorithms.

The multiple-instance learning (MIL) model was developed in order to model biological interactions between a molecule and a binding site of an enzyme. This work was motivated by the fact that understanding the binding site interaction is the most crucial step in drug discovery. An interacting molecule is capable of forming multiple 3D conformations; however, in order for it to activate or deactivate an enzyme, only one conformation need bind at the binding site of an enzyme. MIL first modeled each molecule by a high-dimensional vector that describes its shape, and labeled molecules that bind at a site as positive instances, and those that do not bind as negative. Next, instances (conformations of a molecule) are grouped into a bag (molecule). When a bag is labeled negative (molecule does not bind), it meant all the instances in it were negative (none of the molecule shapes fit the binding site). When a bag is labeled positive, however, it meant that at least one of the instances was positive (one of the shapes fits the binding site). The learner does not know which instance was positive. To sum up, the bag's label is a disjunction of the labels of its individual instances.

Application of MIL in other fields, such as robot vision and content-based image retrieval, followed with great success. In order to expand its applicability, Scott et

al. [17] generalized the MIL model (GMIL), allowing a bag’s label to be represented as an r -of- k threshold function rather than as a disjunction. They also introduced an algorithm (GMIL-1) for learning general geometric concepts. GMIL-1 works by first explicitly enumerating all axis-parallel boxes in the space $\{0, \dots, s\}^d$. Then it assigns boolean attributes to these boxes, and gives these attributes to Winnow algorithm developed by Littlestone [16]. Winnow learns a linear threshold unit. Although GMIL-1 showed a significant improvements over MIL, it suffered from exponential time complexity in d , the dimension of the input space.

In an effort to overcome the costly time complexity, a heuristic algorithm (GMIL-2) was developed [18], which utilized clustering of instances and other methods to significantly speed up the algorithm in practice. Tao et al. [25] further improved the applicability of GMIL by reformulating it using a kernel for a support vector machine [24].

We applied GMIL-2 and kernel-based GMIL to the protein function classification problem. GMIL-2 was tested on *E. coli* genome. GPCR class A family data was used to test both GMIL-2 and kernel-based GMIL. The results revealed that GMIL-2 performed competitively with the data mining prediction (DMP) developed by King et al. [21] on the genome-wide *E. coli* protein function classification problem. GMIL-2 performed very well on 4 of the 11 train groups created for the GPCR class A protein data by achieving an average of 96% accuracy. Although the largest train group ‘Olfactory’ was shown to be too large for GMIL-2 to train and test on, the rest of the 6 groups performed respectably. Performance of kernel-based GMIL far exceeded the GMIL-2 in 6 of the 11 groups, and tied in 2 groups.

1.2 Thesis Organization

This thesis contains five chapters. Chapter 2 will discuss the multiple-instance learning and its algorithms in more detail, followed by the background on the *E. coli* and GPCR data sets used for testing our approach. Also, other works related to protein function classification will be introduced. In Chapter 3, we will describe how the experiments were set up in detail. Then the results of our experiments will be presented and discussed in Chapter 4. Finally, we will conclude this thesis and present some suggestions on future work in Chapter 5.

Chapter 2

Background and Related Work

In this chapter, we first review general concepts of machine learning and multiple-instance learning. Background of the data sets used for the research will be discussed next, followed by reviewing other protein function classification methods.

2.1 Machine Learning Approaches

Machine learning is a model of computation that lets computers learn from experience. Machine learning techniques excel at processing large amounts of data and characterizing noisy patterns. Various machine learning algorithms have been implemented in fields such as image content retrieval, robot vision, and computational biology [4].

A learning algorithm learns a *classifier* that can then be used to predict the label of new entities, e.g. predict the functional class of a new protein. In order to construct classifiers, a learning algorithm trains on a set of *training data* that is already labeled. A learning algorithm iteratively studies the training data set one by one to construct, alter, and fine tune a *hypothesis*, which is an approximation of the *target concept* (the function that is assumed to have originally labeled the training

data). Once the training is complete, a learning algorithm possesses a hypothesis that is an approximation of the target concept. One can use this fine-tuned hypothesis to classify unlabeled data.

2.2 Multiple Instance Learning and Its Derivatives

The multiple-instance learning (MIL) model was introduced by Dietterich et al. [10]. Their work was motivated by the drug discovery problem of predicting whether a molecule would bind at an active site of an enzyme, which results in activation or deactivation of an interacting enzyme. Since shape of a molecule largely determines binding affinity, they represented each molecule by a high-dimensional vector that describes its shape, and labeled molecules that bind at a site as positive instances and those that do not bind as negative. Thus in the MIL model, multiple instances (multiple shapes) are grouped in a bag (one molecule). A bag is labeled positive (binding) if one of the instances in that bag is positive. A bag is labeled negative only if all of the instances are negative, i.e. the bag’s label is a disjunction of the labels of its individual points (if they were known). The label of an instance in a bag is assumed to be determined by its proximity to a target instance. During the training process, the learner does not know which instances are positive for a bag to be labeled positive.

2.2.1 Generalized Multiple Instance Learning

Although MIL was successful in drug discovery and content-based image retrieval [10, 17, 19], its application was limited because a bag’s label is entirely determined by a single instance in the bag. In an effort to expand the application area of MIL, the generalized multiple-instance learning (GMIL) model was introduced by Scott et

al. [17]. In GMIL the target concept is a *set* of instances $C = \{c_1, \dots, c_k\}$, and the label for a bag $B = \{b_1, \dots, b_n\}$ is positive if and only if there is a subset of r target instances $C' = \{c_{i_1}, \dots, c_{i_r}\} \subseteq C$ such that each $c_{i_j} \in C'$ is near some instance in B . Here r is a threshold indicating the minimum number of target instances that must each be “hit” by some instance from B (the same instance in B could hit multiple target instances). In other words, if we define a boolean attribute a_i for each target instance c_i that is 1 if there exists an instance $b_j \in B$ near it and 0 otherwise, then the bag’s label is some r -of- k threshold function over the attributes (so there are k relevant attributes and B ’s label is 1 if and only if at least r of these attributes are 1). Note that if $r = 1$ then this model is the conventional multi-instance model, except that there are multiple target instances and the final concept is a union of these instances. If $r = k$ then a conjunctive model exists, where each target instance must be hit by some instance in B .

This learning model was further fortified by the use of “repulsion” instances. If we let $\bar{C} = \{\bar{c}_1, \dots, \bar{c}_{k'}\}$ be a set of “repulsion” instances, then a positive bag is required *not* to have any instances near each instance of $\bar{C}' = \{\bar{c}'_{i_1}, \dots, \bar{c}'_{i_{r'}}\} \subseteq \bar{C}$ in addition to having a instance near each instance in C' . This means that to be positive, certain feature values have to be present in some instances of bag B , but also certain feature values must be *absent*.

2.2.2 GMIL-1

When Scott et al. introduced the generalized MIL model, they also adapted an algorithm from Goldman et al. [12] to learn geometric concepts in this model. This algorithm, called GMIL-1, assumes that each bag is a multi-set of at most n instances from the finite, discretized space $X = \{1, \dots, s\}^d$. It enumerates all the possible $N = \lceil s(s+1)/2 \rceil^d$ axis-parallel boxes in X and creates two attributes for each box b :

a_b and \bar{a}_b . Given a bag $B \in X^n$, the algorithm sets $a_b = 1$ if some instance from B lies in b and $a_b = 0$ otherwise. Then $\bar{a}_b = 1 - a_b$. These $2N$ attributes are given to Littlestone's [16] algorithm Winnow, which learns a linear threshold unit. Winnow is very similar to the Perceptron algorithm [23]. It is used for learning arbitrary linear-threshold concepts that are allowed to change over time in the on-line model of learning. It updates its weights multiplicatively rather than additively, which often yields much faster convergence than Perceptron. Winnow associates a weight W_i with each boolean attribute, if $W_{tsum} \geq \theta$ (where W_{tsum} is the sum of W_i over all attributes, θ is threshold of Winnow), the bag is predicted 1, otherwise predicted 0.

The time complexity of this algorithm is $\Omega(s^{2d})$ per trial, which is exponential in both $\log s$ (the number of bits needed to describe each instance) and dimension d . To reduce the time complexity, Goldman et al. partitioned the set of N boxes into *groups* such that only one representative box would exist in each group. In order to do so, the set of groups is built by using the instances from all bags to rectilinearly partition X as shown in Figure 2.1. When the lower left corners and the upper

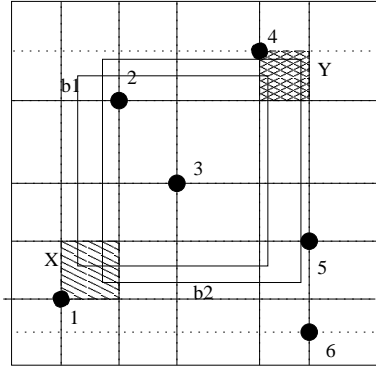


Figure 2.1: An illustration of a rectilinear model space partitioning, and grouping of boxes. Points labeled 1 to 6 are the instances.

right corners of both boxes b_1 and b_2 reside in the region X and Y in Figure 2.1 respectively, both boxes must contain the same instances. Thus both boxes contain

the same instances, and $a_{b_1} = a_{b_2}$ and $\bar{a}_{b_1} = \bar{a}_{b_2}$ for all bags. With this reduction, the exponential dependence on $\log s$ is removed. Although GMIL-1 does not scale well to high dimension (e.g. $d > 5$), on low-dimensional data, GMIL-1 was shown to be competitive with (and often superior to) algorithms in the conventional multiple-instance learning model [17].

2.2.3 GMIL-2

To allow scaling to higher dimensions, heuristic GMIL-2 was created by Tao and Scott [18], which can be thought of as an approximation of GMIL-1. GMIL-1 and GMIL-2 are similar in that they both group boxes, assign boolean attributes to these groups of boxes, and give these attributes to Winnow to learn an r -of- k threshold function over these attributes. The reduction of time complexity comes from how GMIL-2 builds the groups. Rather than taking the sum of instances from all training bags, GMIL-2 only takes a representative set of instances ϕ that capture the distribution of the training bags. Then it constructs groups by directly observing all subsets of ϕ , instead of iterating through all rectilinear partitions of X .

The basic concept of the new grouping algorithm is the same as for GMIL-1. That is, we can group boxes together only if they contain the same set of instances. Box b_1 in Figure 2.2 can be represented as a set $S = \{1, 2, 3\}$. Any box that contains instances 1, 2 and 3 only belongs to this group, such as boxes b_1 and b_2 in Figure 2.2. Therefore any subset of $\{1, 2, 3, 4, 5, 6\}$ has potential to become a group. However, not all of these subsets are valid groups. Set $\{4, 5\}$ for example, is not considered valid since any box containing instances 4 and 5 must also contain instances 2 and 3. Hence, the algorithm gathers each valid groups λ , which contains all the valid subset $S \subseteq \phi$ that does not have an instance $i \in \phi \setminus S$.

Naturally the larger the ϕ , the more accurately the true distribution of the in-

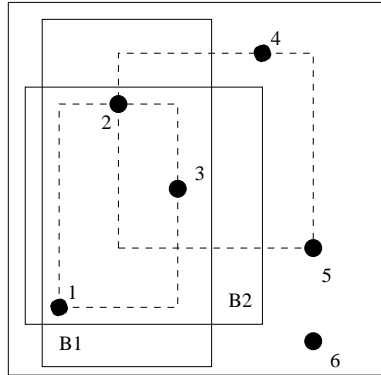


Figure 2.2: An illustration of GMIL-2 box grouping algorithm. B1 and B2 boxes are grouped into dashed-lined box formed by instances 1, 2, and 3.

stances in the bags can be represented. However, because instance representatives generated from clustering usually lie in distinctive regions from one another, increasing $|\phi|$ using only cluster representatives will drastically increase $|\lambda|$. In order to avoid this from occurring, Tao and Scott developed a heuristic that keeps $|\lambda|$ small by observing that if instances in Figure 2.2 are aligned collinearly, the number of distinct valid groups would decrease significantly. Thus once the small number of instance representatives are created, they insert many collinearly aligned random instances to the space as shown in Figure 2.3. Tao and Scott pointed out that the position

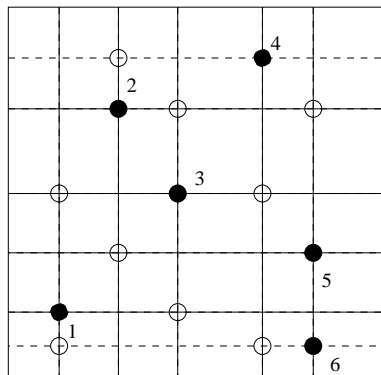


Figure 2.3: An example of how collinear instances are added.

of the random instances are determined by the instance representatives of clusters,

therefore these positions in space X should have more valuable information than any other arbitrary values.

One version of GMIL-2 has time complexity exponential in d like GMIL-1, but runs much faster than GMIL-1 in practice without sacrificing much generalization accuracy. Another version of GMIL-2 has time complexity that is polynomial in d (so it scales to higher dimensions), but has complexity that is exponential in the number of instances in each bag n and the number of bags m in the training set. Although the time complexity of GMIL-2 was significantly reduced from GMIL-1, it was still limited to low-dimensional spaces and small numbers of instances and bags. We used the latter version of GMIL-2 in order to use as many d as possible to generate accurate classifier.

2.2.4 Kernel-Based GMIL

Tao et al. [25] once again reformulated GMIL-1 to this time scale polynomially in both n and d . Their approach implements a kernel function that can be used with a support vector machine [24] to efficiently learn generalized multiple-instance concepts. A kernel that simulates GMIL-1 was formulated based on the observation that given two bags $P, Q \subseteq X$ and a mapping $\vec{\psi}_\wedge(P) = (a_1, \dots, a_N)$, where $a_i = 1$ if the corresponding box $b_i \in B_X$ contains at least one instance from both P and Q , and 0 otherwise, then if the Perceptron algorithm is used to learn a linear threshold unit, the algorithm GMIL-1 can be exactly simulated using the kernel

$$k_\wedge(P, Q) = \vec{\psi}_\wedge(P) \cdot \vec{\psi}_\wedge(Q) = |B(P \wedge Q)|, \quad (2.1)$$

where $B(P \wedge Q)$ is the set of boxes that contain an instance from P and an instance from Q . The kernel is then fed into the lightweight support vector machine (SVM) to

learn geometric multiple-instance concepts. In order to apply SVM, which is a binary classifier, to our multi-classed dataset, Tao et al. utilized multi-class SVM method described in section 2.5.4.

The performance of the new kernel-based GMIL algorithm was evaluated against GMIL-1 and GMIL-2 using several data sets. The evaluation revealed that the kernel-based GMIL far exceeded the performance of GMIL-1 and GMIL-2 in terms of generalization error and speed. Further, this new algorithm outclassed most conventional MIL algorithms on high-dimensional data ($d > 160$).

2.3 *E. coli* Genome

Escherichia coli K-12 is a prokaryotic bacterium. For several decades, *E. coli* has been extensively used as a model organism for basic studies of biochemistry, physiology, genetics and biotechnology. The genome sequence of this organism was completed in 1997 by the University of Wisconsin, Madison. In consequence, *E. coli* genome today is by far the most well-studied bacterium in terms of both genome structure and function. Moreover its functional hierarchy has probably a higher percentage of functions known from direct experimentation of any organism. Hence it is an excellent organism to experiment on.

GenProtEC is an *E. coli* genome and proteome database (<http://genprotec.mbl.edu/>). They developed a system for classification of cellular functions in *E. coli*, which is called MultiFun. It is based on the previous classification system developed by Monica Riley [1]. MultiFun has a close resemblance to engineering diagrams that illustrate the hierarchical organization of systems with general to specific functions. This “functional hierarchy system” of *E. coli* contains 6 major categories as shown in Tables B.1 and B.2 under the column ‘level 1’, which are further subdivided in a hierarchical

system into two more levels (see level 2 and level 3 columns in Tables B.1 and B.2).

Recently the transport classification system of Milton Saier [2] has been incorporated to the database, modifying the 6 major categories to following 10 categories: Metabolism, Information Transfer, Regulation, Transport, Cell Processes, Cell Structure, Location, Extrachromosomal Origin, DNA Site and Cryptic Genes. This new classification system was not available for our research, therefore the classification system with 6 major categories was used for the research.

2.4 GPCR Protein Superfamily

G-protein coupled receptors (GPCRs) form a large superfamily of proteins composed of five major classes (A-E) and more than 30 subfamilies. They are integral membrane proteins characterized by seven hydrophobic domains, predicted to represent transmembrane spanning regions. They are involved with signal transduction across cell membranes, and many medically and pharmacologically important proteins are included in this superfamily: e.g., Acetylcholine receptors, Dopamine receptors, and Opioid receptors. Other than the structural similarity (7 TM regions), sequence similarity among GPCRs is limited only in short regions, thus hindering efforts to classify a function of a potential GPCR gene by means of sequence or structural similarity.

2.5 Other Protein Function Classification Methods

2.5.1 Protein Function Classification Problem

The fundamental concept of protein function classification is to build a discrimination function (classifier) that is capable of accurately labeling each protein with its functional class. Such bioinformatics predictions are useful to experimental biologists, since it is clearly more efficient to test an accurate prediction than to randomly test for possible functions. The most popular discriminatory methods to date are the statistically based sequence similarity techniques. In these methods, a potential function is assigned to a newly sequenced protein if it has a high primary sequence similarity with functionally labeled proteins in the database. Numerous such algorithms have been developed, such as FASTA [27], PSI-BLAST [28], and HMMER [29]. Although sequence similarity methods are tremendously useful, they do not perform well when dealing with large protein classes with low sequence conservation, such as GPCR membrane proteins [14]. Therefore we introduce a new learning algorithm that uses structural properties to build classifiers. These structural properties include various properties of the residues in the protein sequences. We use this information to model proteins via algorithms in the generalized multiple-instance learning model. Then we test our algorithm's performance with *E. coli* genome and GPCR protein data sets.

2.5.2 Property-Based Sequence Analysis

Recently, Kim et al. [14] studied G Protein-Coupled Receptors by mapping each candidate protein's amino acids into following seven properties: GES hydrophathy index [11, 13], solubility [3], polarity, pI, Kyte-Doolittle index [15], α helix index [9],

and molecular weight. For each of the 7 properties, one value is computed per amino acid, so the sequence of n amino acids is transformed to a sequence of n numbers. Thus a total of $7n$ numbers were generated for the entire sequence.

These property sequences are rather noisy by nature [14]. Thus they were smoothed with a size-16 Gaussian kernel to filter out noise before they computed summary statistics (mapping to a single-instance learning model) of these numeric sequences and inferred a linear discriminant function to separate GPCRs from non-GPCRs. The “kernel” for smoothing defines the shape of the function that is used to take the average of the neighboring points. A Gaussian kernel is a kernel with the shape of a normal distribution curve. A protein sequence is described by a set of variables x_1 through x_n , and for each x_i , there is a value x_{ij} for the i th amino acid index value at the j th position. Thus x_{i1} through x_{ik} constitutes a profile of the protein in terms of the i th amino-acid property index. So each property sequence was smoothed by applying the Sliding Window Recognizer [13], which transformed the numeric sequence as follows: $x'_{ij} = \sum_{k=-s}^s w_{j-k} x_{j-k}$, where s is the kernel size and w is the kernel window (Figure 2.4). Kim et al. determined that the kernel size (win-

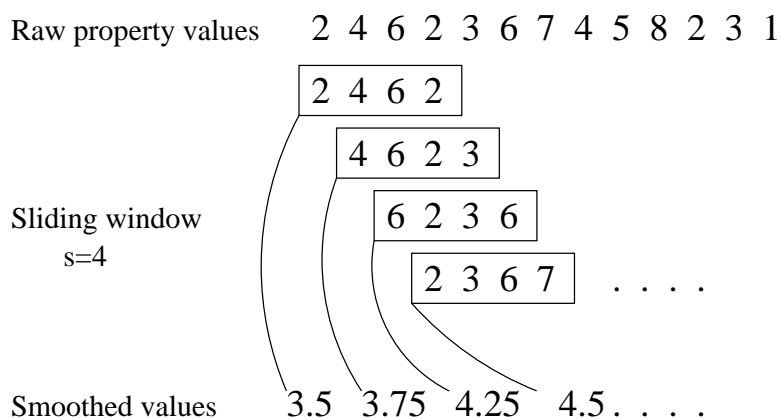


Figure 2.4: An example sliding window recognizer with $s = 4$, $w_i = 1$ for all i .

dow size) $s = 16$ performed well. This made sense with GPCR proteins, because it

characterized the typical period of transmembrane domains, which is approximately 16 to 22 amino acids long.

Kim et al. then computed summary statistics of these numeric sequences and inferred a linear discriminant function to separate GPCRs from non-GPCRs. Their method was superior to the conventional sequence homology methods for classifying GPCRs.

2.5.3 Data Mining Prediction (DMP)

King et al. [21] attempted to predict functional classes of unknown ORFs using Data Mining Prediction (DMP). DMP uses data mining/machine learning to induce rules that map from sequence to functional class. King et al. states that DMP has several advantages over the conventional sequence similarity approach (SIM) in classifying a function to a protein:

- Function can be predicted in the absence of homology to a sequence of known function.
- More general types of SIM can be utilized allowing more remote homologies to be detected.
- Explicit comprehensible rules can be produced which may provide biological insight.

The disadvantages of DMP are:

- It requires standard SIM functional assignments to bootstrap from.
- It can only identify the functional class of a protein, not its specific function.

The basic algorithm of DMP is as follows:

1. Retrieve the identified open reading frames (ORFs) and their known functional assignments.
2. Describe each ORF in the genome using a defined language.
3. Use data mining to identify frequent patterns in the descriptions of the ORFs. The Inductive Logic Programming (ILP) algorithm Warmr [26] was used.
4. Use machine learning to learn rules which map from the attributes describing the sequences to their function.
5. Use the learnt rules to (inductively) infer the functional classes of ORFs of unknown function.

Three types of information were computed to describe protein sequences. First, the sequence based attributes (SEQ) are based on the sequence's composition of residues, e.g., the number and percentage composition of residues of each type, molecular weight, and isoelectric point. Next, the sequence similarity based descriptors (SIM) were created by performing PSI-BLAST for each ORF in the *E. coli* genome. Then DMP gathered information from the results, such as homologous proteins found by PSI-BLAST, and their e-values. Finally, predicted secondary structures of the ORFs were described.

Of the three basic types of description, SIM was most effective. It gave both the highest accuracy (the number of ORFs predicted to have the correct function/the number of predictions * 100) and coverage at each of the three hierarchical function levels. Reporting the coverage of the test set was necessary to describe the performance of DMP, since it has an option to not label test sequences based on the rules King et al. set. Therefore % coverage is derived by dividing the number of predicted test sequence by the number of total test sequences, then multiplying it by 100.

SIM predicted 15-30% of the test set (about 200 out of 700 ORFs or less) with approximately 70% accuracy. STR came in second with 10% coverage with an accuracy of 59%. King et al. also used combinations of descriptor types (STR+SIM or STR+SIM+SEQ) to learn the rules; however, SIM descriptor all by itself outperformed any other combinations of descriptors.

One major flaw of DMP would be the coverage of the test set. Simply put, DMP gave up labeling test sequence when it did not meet certain conditions. Unfortunately, these conditions were not mentioned in their algorithm description.

DMP rules provide a novel way of predicting an ORF's function from its sequence. Although they cannot provide the specificity of the predictions provided by conventional SIM searches, the DMP approach has the ability to make correct predictions when standard methods fail, and to provide independent confirmation of a prediction made by standard methods [21].

2.5.4 GPCR Classification with SVM

Support Vector Machines (SVMs) are a class of statistical learning algorithms based on the theories introduced by Vapnik [5]. In 1990s, they gained a large popularity in the machine-learning community and have been applied to various learning problems [6]. Karchin et al. [7] applied SVM to classify subfamilies of GPCR classes A and C. In order to classify multi-class data such as GPCR subfamilies, Karchin et al. extended binary SVMs to multi-class SVMs by using the one-versus-rest training method. If $k > 2$ classes exist in a train set. Then k train sets are created that has one class as positive while labeling rest of classes as negatives.

In order to be distinct from BLAST and SAM, amino acids are transformed to Fisher score vectors (FSV) [8], which is an intermediate representation between a sequence and an hidden Markov model (HMM). Once mapped to FSV, they built

a library of SVMs for each protein class of interest, and scored each test sequence against the entire library. The performance of SVM was then tested against BLAST and HMM based SAM-T2K. Karchin et al. also compared the results of SVM with a nearest-neighbor method based on kernel scores and a fast SVM approximation method called SVMtree.

In two-fold cross-validation experiments, the errors per sequence at the minimum error point (MEP) were 13.7% for multi-class SVMs, 17.1% for SVMtree method, 25.5% for BLAST, and 30% for profile HMMs. The MEP is the score threshold where a classifier makes the fewest errors of both kinds (false positives plus false negatives). The percentage of true positives recognized before the first false positive was 65% for both SVM methods, 13% for BLAST, 5% for profile HMMs.

Chapter 3

Applying Generalized Multiple Instance Learning to Protein Function Classification

Careful experiment implementation is by far the most crucial part of the research. In this chapter we overview our implementation of the experiments, and discuss the data sets we chose to test our algorithm.

3.1 Data Preparation

We applied most of the steps described in section 2.5.2 taken by Kim et al. to prepare our *E.coli* and GPCR data sets prior to the machine learning process. Our data preparation pipeline is as follows:

1. Generate multiple alignments for each functional class (label) using ClustalW.
2. Group functional classes according to their sequence length similarities as shown in Table A.1 and Table A.2.

3. Map each amino acid to 8 numbers (the first number is its position in the multiple aligned sequence, and the remaining 7 numbers are the 7 physico-chemical property values derived from Kim et al.).
4. Scale up the sequence lengths of functional classes with shorter sequence lengths in each training set (see section 3.1.2).
5. Use size-16 Gaussian kernel to smooth (reduce the noise of) the property sequences (see section 2.5.2).
6. Cluster the instances by using k-means clustering algorithm. Then select the representative instances of clusters and insert collinear random instances.

3.1.1 Multiple Alignment Dependency

One distinct difference in our data preparation process is that we multiply aligned our sequences using ClustalW before transforming amino acids to the property values. Although we would like to avoid aligning sequences due to the low primary sequence similarity in both GPCR superfamily and *E. coli* genome, the representation used by our algorithms requires relative positional information. When mapping the property sequences to multiple-instance examples, the first coordinate of instance p in bag P must be related to the position in the original sequence of \vec{b} 's corresponding amino acid. E.g. if amino acid x is 25% downstream from the start of the sequence, then we would expect \vec{b} 's first coordinate to be about 25% of the way from the origin to the final point. More specifically, since all the MIL algorithms applied to this data set are looking for similar property values in certain regions of each positive bag, it is critical that these regions have similar coordinates in the first dimension, lest the algorithms fail to detect the similarities (Figure 3.1).

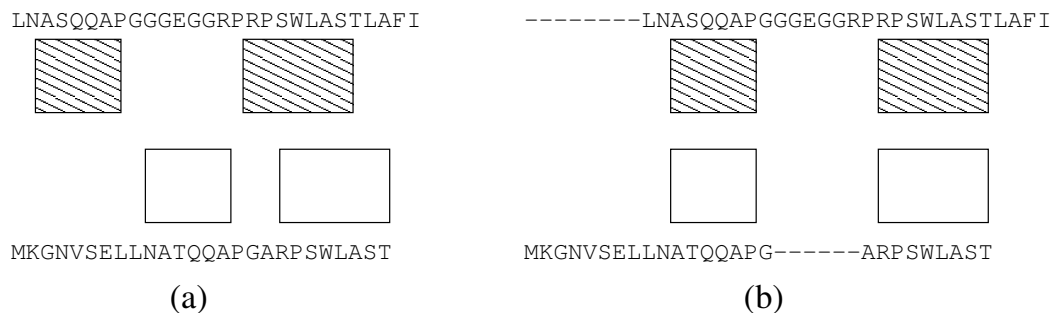


Figure 3.1: (a) Before sequence alignment: Positive bags (rectangles) are not aligned. (b) After sequence alignment: Positive bags are aligned with one another.

One alternative method that we can employ to avoid multiply aligning sequences is the pairwise alignment. Instead of attempting to align several proteins at once, the pairwise alignment aligns a pair of sequences at a time. Note that we can only use pairwise alignments with the kernel version of GMIL-1, since a kernel only considers bags pairwise. In contrast, GMIL-2, which does not use kernel, does not limit itself to pairwise comparisons so it cannot use pairwise alignments. In order for kernel-based GMIL to be comprehensive, however, all pairs of protein sequences must be pairwise aligned including with itself. Therefore if a training data is comprised of 5 protein sequences, the pairwise alignment generates up to $5 * \binom{5}{4} + 4 = 14$ alignments.

3.1.2 Grouping of Functional Classes

As shown in Tables A.1 and A.2, for *E. coli* there are 136 functional classes in level 3 functional hierarchy. Obviously the most desirable scenario would be to build classifiers using all 136 functional classes; however, the time complexity and the memory requirement of the algorithm hinder us from doing so. Therefore we divided up 136 functional classes into several training groups (see Tables A.1 and A.2) by combining protein sequences of grouped functional classes. Training group 1 in Table A.1

for example, was created by combining protein sequences from the functional class ‘SMR family’ and ‘ATP-proton motive force inter-conversion’. Within each training group, we trained a multi-class classifier. E.g. we used training group 1 to build a two-class classifier to distinguish SMR from ATP-proton motive force interconversion and we used group 3 to train a three-class classifier to distinguish Glycoprotein from Ribosomes-maturation and modification from MIP family. To ensure that our experimental results are not optimistically skewed, we did not partition the test set as we did with the train set. Instead we evaluated each classifier on each test sequence, using confidence scores to make final predictions. The confidence scores will be discussed in detail in section 3.2.2.

The main criterion for clustering functional classes was the lengths of the sequence alignments in each functional class. Specifically, proteins in functional classes are grouped into a training data if the difference in the sequence alignment lengths of each class is smaller than 30 amino acids. The reason for clustering functional classes with similar sequence lengths is because it is not desirable for the learning algorithm to learn the lengths of the protein sequences as the only criterion for classifying unknown examples. Figure 3.2 illustrates the bad grouping of two functional

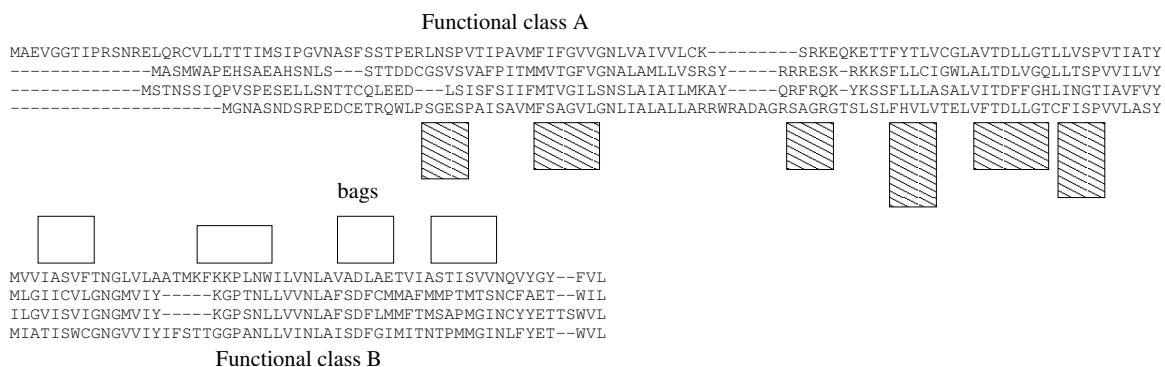


Figure 3.2: An example of a bad grouping scheme where the lengths of the two functional classes are too different. In such cases, MIL could learn the lengths of classes as classifiers.

classes. Shaded rectangles (bags) for the longer functional class A are concentrated towards the end of the sequence alignment where bags from functional class B have no instances. When the learner creates classifiers according to Figure 3.2, it becomes trivial to classify class A from class B. By avoiding such situations, the algorithm would classify unknown examples based on the 7 physico-chemical properties rather than less desirable property such as the length of a sequence. In order to further prevent this from occurring, we also scaled up (stretched out) the functional classes with shorter protein alignment lengths. In Figure 3.3 the shorter functional class B in Figure 3.2 has been stretched out to match the length of the functional class A. Let $\{G_1, G_2, \dots, G_n\}$ be the sequence alignment lengths of n functional classes in

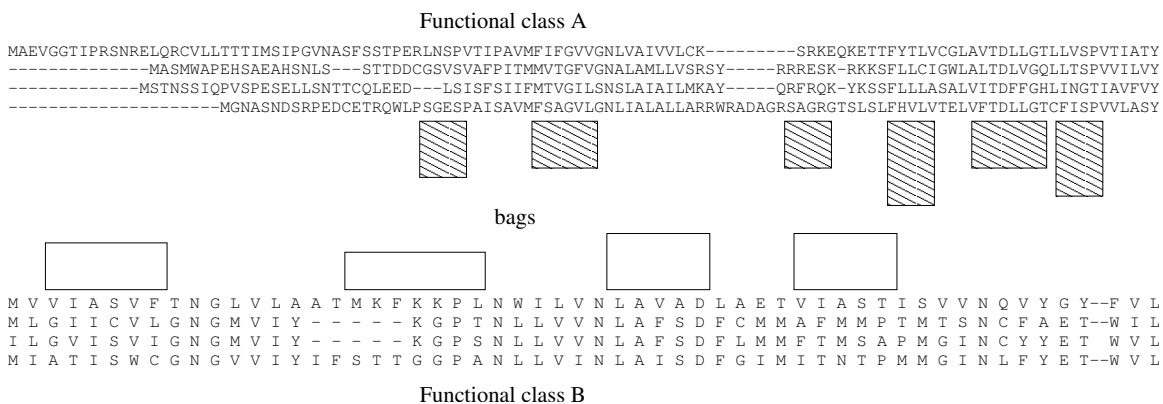


Figure 3.3: An example of stretching out shorter functional class.

one training set. If G_k is the maximum length among the n classes, we derive the scale factors $\{SF_1, SF_2, \dots, SF_n\}$ for n classes by dividing G_k with each of G_i , where $i = 1 \dots n$. Finally, let $\{S_{1,1}, S_{1,2}, \dots, S_{1,n}\}$ be the position numbers of a amino acid sequence in one of the shorter functional classes. Then this sequence is scaled up by multiplying SF_1 to $S_{1,i}$, where $i = 1 \dots n$. Training set 10 in Table A.1, for example, has three functional classes, Hemeporphyrin, MATE family, and Glutamine, grouped together. The lengths of protein sequence alignments of each functional class are 455,

459, and 469 amino acid residues long, respectively. In order to scale up the sequence alignment lengths of classes Hemeporphyrin and MATE system, we derive the scale factor for Hemeporphyrin $469/455 = 1.03$ and MATE family is $469/459 = 1.02$. Then each scale factor is multiplied to their respective protein sequence position numbers. Note that partitioning based on length and scaling are only necessary for GMIL-2. It is not necessary for the kernel if it uses pairwise alignments.

3.2 Train and Test Data Sets

To test how well our algorithm performs on a function classification problem, we used two large data sets, the *E. coli* genome and the GPCR superfamily, for training and testing. Sections 2.3 and 2.4 describe the advantages and significance of using these data sets as our training and testing sets, and further details will be addressed in the following sections.

3.2.1 Functional Hierarchies

Organizing the functions of proteins into classes is one of the most important conceptual advances in functional genomics [20]. For the *E. coli* genome we selected Riley group's (<http://genprotec.mbl.edu/>) functional hierarchy annotation. Riley functional classification is accepted as one of the most researched and thorough of all functional hierarchies. Riley's *E. coli* functional hierarchy has 3 levels. A typical example of the classification of a protein in this hierarchy is that of pyruvate formate lyase activating enzyme. This has a level 1 (most general) class of 'Metabolism of small molecules', a level 2 class of 'Energy metabolism, carbon' and level 3 (most specific) class of 'Anaerobic respiration' [21].

We obtained functional hierarchies of GPCR superfamily from www.gpcr.org.

GPCR superfamily has 5 major classes appropriately labeled alphabetically from A to E. Class A is the largest and most studied class among them, therefore we only used class A to generate our training and testing data, which will be discussed in subsection 3.2.3. Unlike the *E. coli* functional hierarchy, GPCR has 4 levels of functional classes.

3.2.2 *E. coli* Train and Test Data Sets

For *E. coli* we used the 4289 open reading frames (ORFs) or protein sequences identified by Blattner et al. [22] and took the functional assignments from the Riley group. With this information we were able to assign 3 different levels of functional classes (from most general to most specific) to each protein sequence. A total of 2110 *E. coli* protein sequences were functionally classified. We then selected 1410 sequences as the training set and the rest of the sequences (700) as the testing set. We considered the Riley’s functional classes ‘Open Reading Frames’ and ‘Miscellaneous’ to be ‘function unknown’. Therefore these sequences were not included in our training/testing set.

The training set (1410 ORFs) was partitioned into functional classes at each level. There are 6, 20, and 136 functional classes in level 1, 2, and 3, respectively as shown in tables B.1 and B.2. Therefore 1410 ORFs were separated into 6 functional classes according to their labels in level 1. In level 3, 1410 ORFs were separated into 136 functional classes. For the training sets in level 3, functional classes were grouped together as described in section 3.1.2.

For the same reason we multiply aligned each functional class in training group (see section 3.1.1), we aligned the testing set as well. Assume we have 2 functional classes in one training group. Since we do not know which functional class alignments in training group to align testing sequence T to, we would align T to both, getting bags T_+ and T_- . Each would be labeled by our classifier, and the prediction that

is most confident is the final prediction. Confidence of each prediction is defined as $|W_{totalsum} - \theta|$, where $W_{totalsum}$ is the sum of W_i over all attributes and θ is the threshold of Winnow (see section 2.2.2).

With this data preparation scheme, the number of sequences in testing set increases by a factor of n , where n = number of functional classes in each training group, e.g. if a group has 3 functional classes, the number of testing sequences would triple to 2100.

As discussed in section 2.2.3, we must select an appropriate number of representative instances of clusters N and random instances K that are collinearly aligned with N in order to begin the training process. Let G be the group of boxes. Then, for every training set we built a group set $\lambda = \{G \mid S \subseteq \phi \text{ and } S \text{ is valid subset}\}$ by generating 5 instance representatives of clusters and inserting 50 random instances. We have determined after a several test runs that using 5 cluster instances and 50 random instances yielded the optimal performance without increasing the time complexity.

3.2.3 GPCR Class A Train and Test Data Sets

As mentioned in section 2.4, GPCR superfamily has been a difficult family of proteins to classify using conventional sequence similarity methods. Thus it was a good candidate to test our algorithm on. The level 1 (most general) of the GPCR functional hierarchy is classified into 5 classes. Among them, class A is the largest and most well studied GPCR class. Therefore we focused our test only on this class of proteins. We first partitioned the class A proteins according to the level 2 functional hierarchy classification, which has 16 functional classes (see Tables C.1 and C.2). As shown in Tables C.1 and C.2, most of the level 2 functional classes are further classified into more specific level 3 functional classes, e.g. ‘Nucleotide-like’ GPCR proteins are further broken down to ‘Adenosine’ and ‘Purinoceptors’. Naturally these level 3

functional class labels became the labels for each training group partition. 5 of the 16 functional classes, ‘Cannabinoid’, ‘Platelet activating factor’, ‘Lysosphingolipid-LPA-EDG’, ‘Melatonin’, and ‘Viral’, however, did not have any labels (no level 3 classifications), as shown in Tables C.1 and C.2. Since it is not possible to train and test on the data without any labels, these 5 functional classes were taken out of the experiment, thus a total of 11 partitions were generated for the experiment.

All class A protein sequences were obtained from the Swiss-Prot, through the links provided by www.gpcr.org. Then we randomly selected $\frac{1}{3}$ of the total number of sequences in each functional class to create the testing set. The rest of the sequences in each functional group ($\frac{2}{3}$ of sequences in each level 2 functional class) was set as the training set.

We took a different approach to align train and test sets than the method we used for the *E. coli* genome data. We wanted to stay away from aligning sequences as much as possible, thus the entire train set sequences were multiply aligned, instead of generating several alignments separated by the labels, then scaling up the shorter sequence alignments. GPCR sequences within each group were very similar in length, further reducing the concerns mentioned in section 3.1.1.

For the reason described above, there was no need for scaling up the shorter sequence alignments, and with only one training group alignment, we no longer had to generate several different alignments of each test sequence, which previously caused doubling and tripling of the number of the testing sequences depending on the number of labels in each training group.

Regarding the building of the cluster and random instances, we began with the same number of clusters and random instances as the *E. coli* experiments (5 clusters and 50 random instances), but since smaller datasets let us increase the number of cluster and random instances, we tested out various combinations of cluster and

random instances as will be discussed in the next chapter.

Chapter 4

Experimental Results

4.1 Results of GMIL-2 on *E. coli*

The discrimination ability of GMIL-2 was first tested using 1410 training and 700 testing *E. coli* sequences. We knew the correct functional labels for both training and testing sets. As mentioned in section 3.2.2, each test sequence was aligned to all functional class multiple alignments. Level 3, for example, has 136 functional classes, thus 136 different alignments for each testing sequence are generated. Each of the alignments was then labeled and confidence scores, $|W_{totalsum} - \theta|$, were generated by our classifier. For each test sequence, we ranked the 136 confidence scores. Then since we know the true function label of all test sequences, we looked at how the correct function label for a test sequence ranked among the 136 scores. If the correct functional class is ranked 1, our algorithm predicted the function of the test sequence correctly.

Table 4.1 illustrates the performance of our algorithm (GMIL-2). We included the performance of the DMP algorithm developed by King et al. (see section 2.5.3) for comparison. Two different results, GMIL-2(100) and GMIL-2(97), of our algorithm

Table 4.1: The test set results (the numbers 1, 2, and 3 correspond to the levels in *E. coli* functional hierarchy).

Algorithm	Accuracy %			coverage %			Num of correct predictions		
	1	2	3	1	2	3	1	2	3
GMIL-2(100)	19	33	10	100	100	100	130	232	67
GMIL-2(100 adj)	21	43	15	29	26	16	43	79	17
GMIL-2(97)	25	35	18	100	100	100	176	247	126
GMIL-2(97 adj)	26	43	26	29	26	16	53	79	29
DMP(SIM)	75	74	69	29	26	16	152	135	77

The numbers 1, 2, 3 correspond to the levels in *E. coli* functional hierarchy (level 1 the most general, level 3 the most specific). The test set accuracies are: (the number of test sequences predicted to have the correct function/number of predictions) * 100. The test coverages are: (the number of test sequences predicted to have a function/total number of sequences in test set) * 100. The number of correct predictions is the number of test sequences predicted to have the correct function. There were 700 sequences of known function in the test set. GMIL-2(100 adj) and GMIL-2(97 adj) uses simulated coverage to adjust the number of test sequences GMIL-2 predicted.

are shown. GMIL-2(100) is the result derived by only considering the correct predictions (GMIL-2 assigned the function label correctly to a test sequence). GMIL-2(97) includes the “near hits”, which means the confidence score of the correct function label of a test sequence placed in top 3 percentile. GMIL-2(97) will be discussed in detail later.

The true comparison between GMIL-2 and DMP can be made using the number of correct predictions. In levels 1 and 3, DMP performed slightly better than GMIL-2(100) by correctly predicting 18 and 10 more sequences, respectively. GMIL-2(100), however, outperformed DMP in level 2 by correctly predicting 93 more sequences. Since it was possible to report the “near hit” predictions, we included the number of test sequences that GMIL-2 almost correctly predicted (predictions that placed the correct labels in top 3 percentile of the confidence score). Reporting the “near hit” predictions is rather useful than one may think when dealing with large number of functional classes such as in level 3. In level 3 there are 136 functional classes,

therefore when a correct function label places in 97 percentile of the confidence score, it means GMIL-2 was able to narrow the possible function for the unknown protein from 136 to 4. We selected to report 97 percentile as the “near hit” predictions since in level 3, 97 percentile reports test sequences with confidence scores placed in 4th or better.

As shown in Table 4.1, GMIL-2(97) outperformed DMP in all levels. Especially level 1 and 3 results improved significantly. This result is promising due to the fact that we used a naive approach to multiply align both train and test sets in order to align the boxes (see section 3.1.1). If we improve the alignments in both train and test set or even eliminate the need for multiply aligning sequences prior to learning process, there is a great chance that the number of predictions in GMIL-2(100) would resemble GMIL-2(97) results.

Although the coverage of GMIL-2 is always 100%, for comparison purposes we simulated coverage of King et al. and the results are shown with labels GMIL-2(100 adj) and GMIL-2(97 adj) in Table 4.1.

In order to simulate the coverage, we first ranked the highest confidence scores of each test sequence. Then in level 2 in Table 4.1, for example, we selected top 26% of the test sequences from the ranked list. Finally the accuracy was derived by counting the number of correct predictions within the top 26%. The trends between accuracy and coverage for GMIL-2(100) and GMIL-2(97) using the method described above are shown in Figures 4.1, 4.2, and 4.3 for all levels.

The reason for applying the aforementioned method to simulate the coverage of King et al. was because it was the most unbiased method to select the sub portions of the total test sequence. Unless we understand the rules that King et al. used to determine whether to predict or not to predict, we cannot fully compare the GMIL-2(100 adj) and GMIL-2(97 adj) against DMP fairly. Therefore the only fair way to

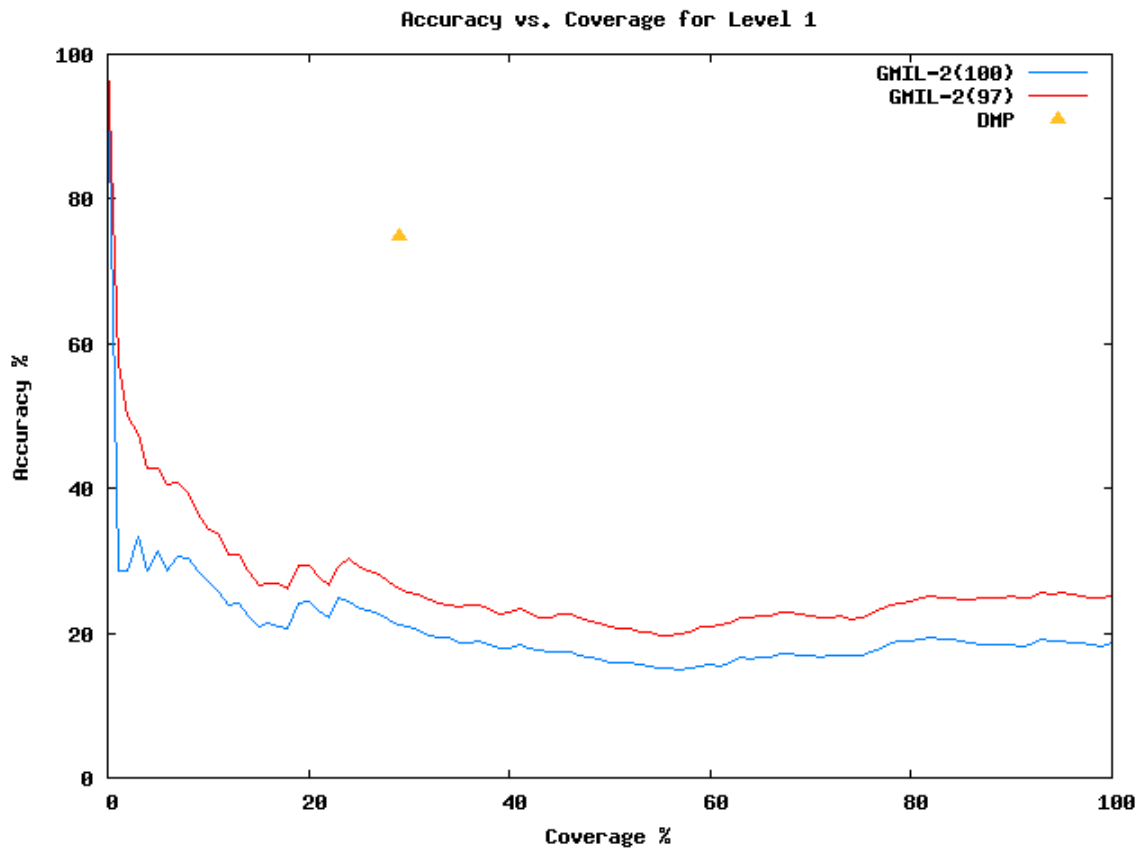


Figure 4.1: Accuracy vs. coverage graph on level 1 for GMIL-2(100) and GMIL-2(97). DMP result is also shown.

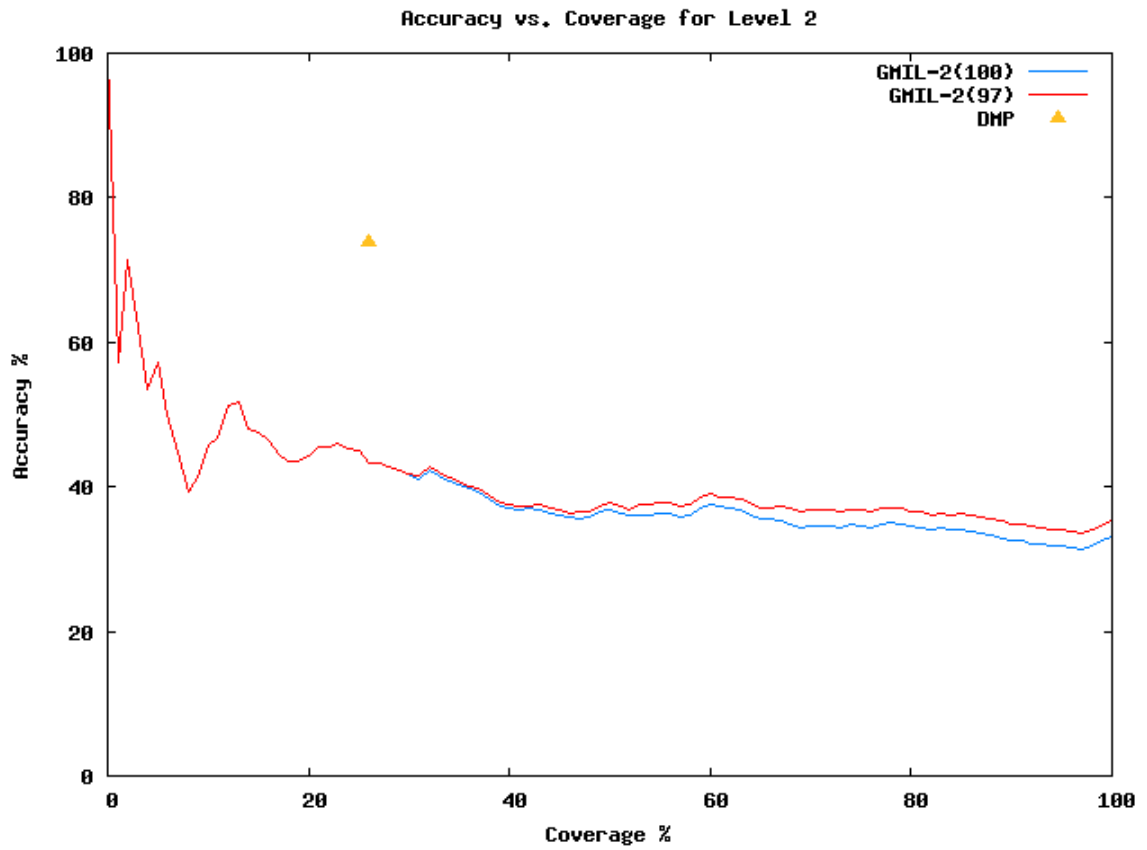


Figure 4.2: Accuracy vs. coverage graphs on level 2 for GMIL-2(100) and GMIL-2(97). DMP result is also shown.

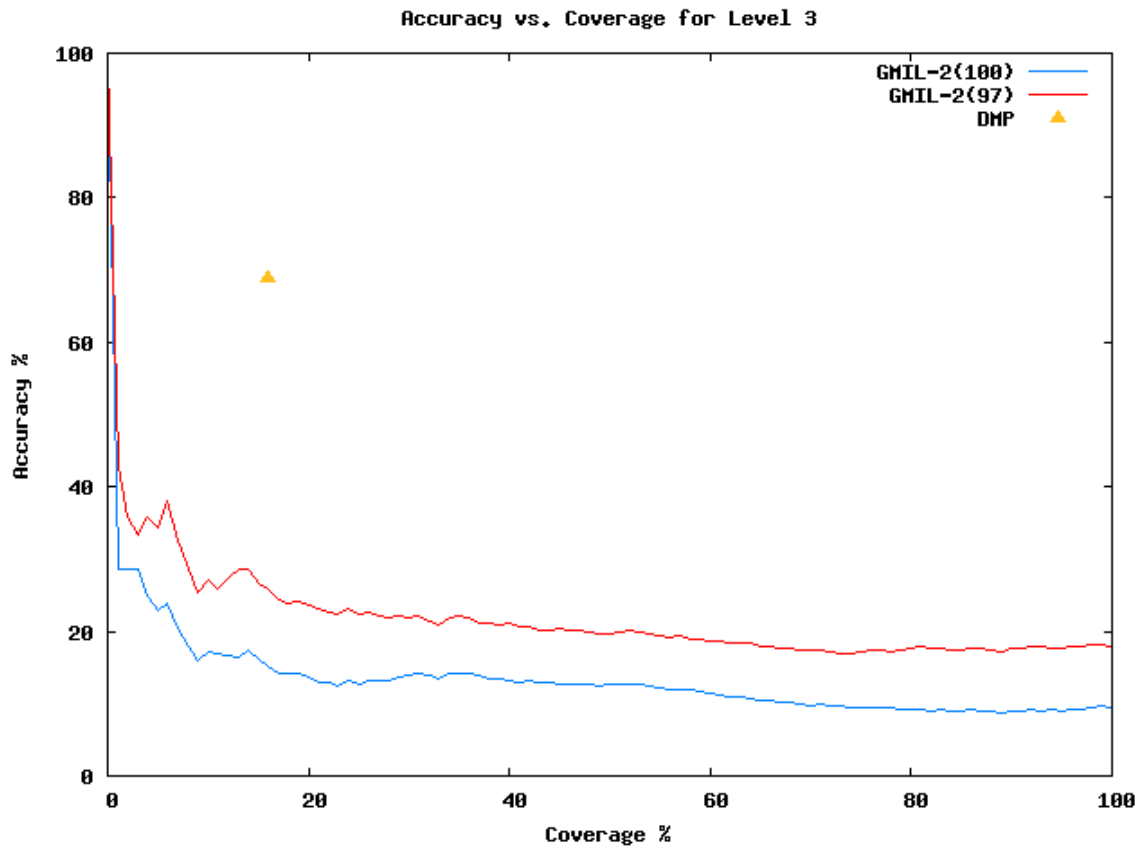


Figure 4.3: Accuracy vs. coverage graphs on level 3 for GMIL-2(100) and GMIL-2(97). DMP result is also shown.

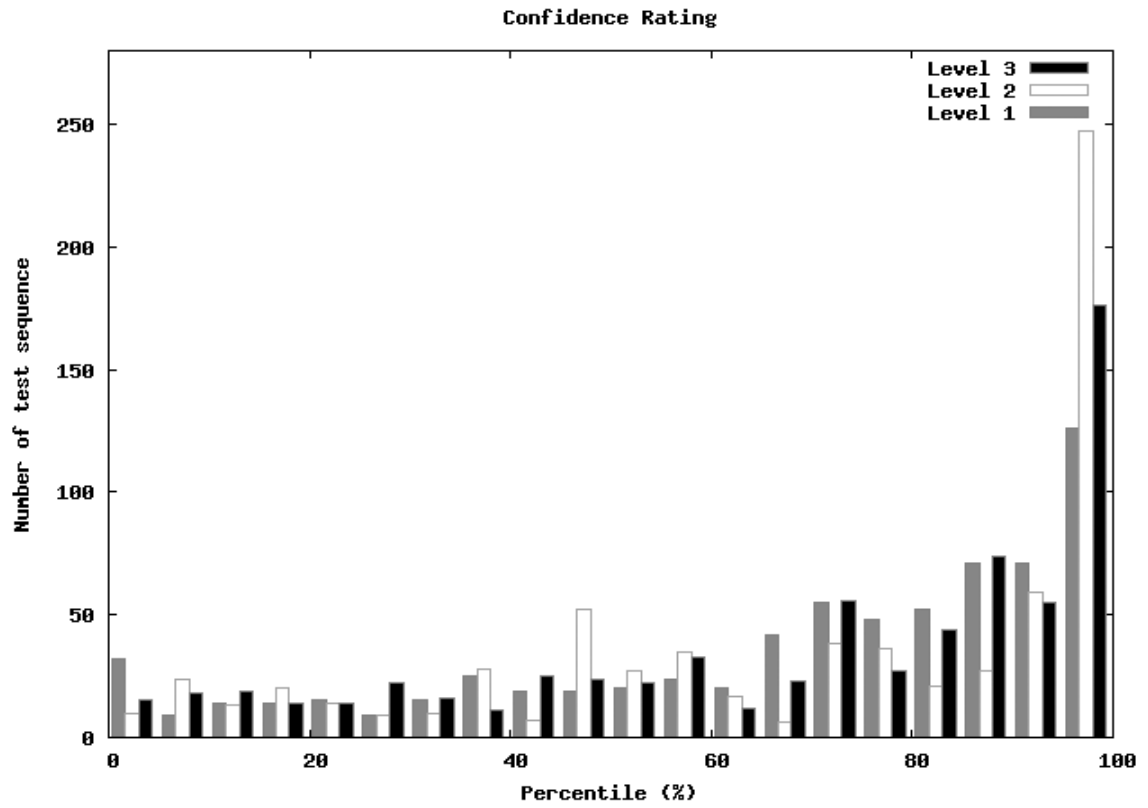


Figure 4.4: A histogram that shows the distribution of confidence ratings

compare the two results is by once again by observing the total number of correct predictions shown in Table 4.1.

GMIL-2 performed best in level 2. This is to some extent expected, as there are many more functional classes to discriminate between in level 3, and in level 1 the heterogeneity of the class becomes too great to derive a good classifier.

Figure 4.4 illustrates the distribution of confidence scores (percentile) in the entire testing sequences in each level of functional hierarchies. In all 3 levels of functional hierarchies, the largest number of testing sequences had confidence scores of 95% or better. More specifically 176, 247, and 126 test sequences in level 1, 2, and 3 belonged to top 5 percentile (scored 95% or better). The majorities (74%, 94%, and 67% respectively) of the test sequences were in fact correct predictions as listed in Table 4.1.

4.2 Results of GMIL-2 on GPCR Class A Data Sets

Recall that in our *E. coli* experiment, each test sequence was aligned to all 136 functional classes. For the GPCR experiment, we wished to avoid aligning sequences as much as possible. Therefore only one multiple alignment for each train set was created. ‘Nucleotide like’ train set in Table 4.2 for example, is made up of 2 functional labels (more detailed functions). Protein sequences in these 2 functional labels were combined to create the train set of 49 sequences. The test set of 25 sequences was generated by randomly selecting $\frac{1}{3}$ of the total available sequences of each 2 functional labels in ‘Nucleotide like’. 49 train sequences were then multiply aligned and each of 25 test sequences were aligned to the train sequence alignment.

The advantages of the new data preparation scheme are that we no longer had

Table 4.2: GPCR Results from GMIL-2 and kernel-based GMIL.

Training group	Accuracy %		Number of		
	GMIL-2	Kernel GMIL	train set	test set	classes
1 (Nucleotide like)	96	100	49	25	2
2 (Hormone Protein)	78	78	28	18	5
3 (Gonadotropin)	75	96	21	12	2
4 (Rhodopsin)	95	98	207	105	4
5 (Peptide)	64	-	547	270	31
6 (Amine)	64	78	255	129	7
7 (Prostanoid)	91	95	39	21	3
8 (Thyrotropin Releasing)	88	88	16	8	3
9 (Class A orphan other)	78	100	112	62	11
10 (Leukotriene B4 recep)	100	67	4	3	2
11 (Olfactory)	-	-	965	415	14

to create several different alignments of each test sequence. Thus the number of test sequences stayed the same, instead of increasing by the number of labels in the train set as it did in *E. coli* (see subsection 3.2.2). This further meant that we no longer had to rank the confidence scores across training groups. We simply ranked weighted sums within each group, e.g., in Hormone Protein there are 5 classifiers, therefore 5 predictions (confidence scores) are generated for each test sequence. GMIL-2 selects the maximum score was the final prediction, and report the % accuracy also reported by GMIL-2.

The GPCR experiment results from GMIL-2 are mixed, as shown in Table 4.2. Experiment groups 1, 4, 7, and 10 had high prediction accuracies (above 90%). Note that group 10 results are difficult to interpret since the size of the train and test sets were very small, as shown in Table 4.2. Groups 2, 3, 8, and 9 had moderate prediction accuracies (70-89%), meanwhile train groups 5 and 6 had the lowest accuracies (64%). Recall that we used the GMIL-2 that is capable of scaling to high dimensions, enabling us to use 8 dimensional data. Unfortunately this GMIL-2 version has time complexity exponential in the number of instance representatives and collinear random instances.

Table 4.3: Improvements for GMIL-2 in training group 5 (Peptide) accuracy via increase in number of instances.

Rep.	Instances	Rand Instances	Accuracy %
5	5	50	51
6	6	60	59
7	7	55	64

The size of train group 11 was too large for GMIL-2 to learn classifiers, caused by the exponential increase in the time complexity while training. For this reason we could not get any results for group 11 for GMIL-2.

Note that the exact same train and test sequence alignments were used for kernel-based GMIL, enabling us to directly compare the results between the two algorithms. Kernel-based GMIL outperformed GMIL-2 in 6 of the 10 train groups by more accurately classifying functions, and tied in 2 groups as illustrated in Table 4.2. Significant improvements were evident in groups 3, 6, and 9. Prediction improvements in groups 6 and 9, especially, are more significant since they had so many labels to deal with. Although the accuracy for the group 10 was reduced with kernel-based GMIL, this was insignificant since the train and test set size for group 10 was very small.

Test set distribution in each of the level 3 functional classes along with the distribution of the number of correct predictions in each class are listed in Appendix D. Tables in Appendix D revealed that most errors in each group are concentrated among classes with relatively small test sequences. Since the size of the train set and the test set are proportional as discussed in section 3.2.3, it further meant that kernel-based GMIL did not perform well on the classes with small number of train sequences. Intuitively, this observation makes sense since the smaller the number of train sequences, the less accurately GMIL can learn.

In an attempt to diagnose the cause of low prediction accuracies for GMIL-2,

we looked at the size of train and test sets, along with the number of labels in each experiment set. The set 5 (Peptide) most likely had too many labels for the algorithm, relative to the number of representative instances and collinear random instances. As a consequence, GMIL-2 could not learn distinct classifiers to discriminate from one sequence from the other. This conjecture was proven to be correct due to the fact that the accuracy of the set 5 improved as the number of representative instances and collinear random instances were increased, as shown in Table 4.3. Although we would have liked to further increase the number of both instances, the time complexity was exponentially increasing as well, thus it became extremely difficult to carry out the experiment.

Chapter 5

Conclusions and Future Development

Conventional MIL has proven to be very effective in modeling problems. Coupled with an efficient algorithm, it can be applied to variety of applications, such as robot vision and content-based image retrieval, and has already proved to be powerful in those fields of applications. Although the development of MIL was motivated by modeling biological problems, extending it to other biological problems such as protein function classification required generalization of how MIL labels its bags. Scott et al. achieved a generalization of MIL, and developed an algorithm GMIL-1 to learn geometric concepts in it. Although the performance of GMIL-1 was superior to algorithms in MIL, it did not perform well in applications that required high dimension (e.g. $d > 5$). Two versions of heuristic algorithm GMIL-2 were developed to speed up GMIL-1. Although the time complexity improved significantly from GMIL-1, it was still limited to small numbers of instances and bags, or to low-dimensional spaces depending on the versions.

We applied GMIL-2 to the *E. coli* genome-wide protein function classification

and GPCR class A function classification problems. With the *E. coli* data, the performance of GMIL-2 model built on the 7 property profiles using primary sequence position information was competitive with DMP developed by King et al. Observing the “near hits” results (GMIL-2(97)) in Table 4.1 was very encouraging, since it would mean outperforming DMP by almost 2 folds in most of the levels. Two areas can be immediately improved to possibly achieve the result similar to GMIL-2(97) in Table 4.1. First, multiple alignments of train and test set can be improved. The better the bags align, the more accurate model will be generated, which would lead to better prediction accuracies. Second, the number of representative instances and collinear random instances can be increased, even though the run time and memory will increase significantly. The more instances are present in the space X , the better GMIL-2 will detect the minute differences in labels, which will lead to creation of better discriminating classifiers. Third, kernel-based GMIL can be applied, which has consistently outperformed GMIL-2 in GPCR and other areas [25].

For the GPCR class A data, we attempted to minimize the need for multiply aligning sequences. Thus only one multiple alignment for each train set was generated, and only one alignment version of test sequences existed. 4 of the 11 train groups (1, 4, 7, and 10) achieved the average accuracy of 96%, and another 4 train groups (2, 3, 8, and 9) achieved average accuracy of 80%. We realized that train group Olfactory was too large to apply GMIL-2. Kernel-based GMIL far exceeded the performance of GMIL-2. It had better prediction accuracies for 6 of 11 train groups. It tied two other groups.

For the GPCR class A data, we varied the number of representative instances and collinear random instances to achieve the result shown in Table 4.2. As expected the general trend was that the accuracy of GMIL-2 increased as the number of instances increased as shown in Table 4.3.

Since kernel-based GMIL does not have an exponential time complexity and based on the performance comparison, it is mostly likely to replace GMIL-2 as the main working platform for applications. More specifically, it is capable of scaling up to very high dimensions ($d > 160$) without complications. In our application, this means the number of properties (currently 7) can be increased by a large factor by using the amino acid property databases, such as AAindex [30]. We speculate this would greatly enhance the sensibility of detecting distantly related proteins.

Recall that we multiply aligned sequences for GMIL-2 and kernel-based GMIL experiments; however, since kernel-based GMIL is capable of training with pairwise alignments, experiments that eliminate the need for multiple alignments are currently in the works. It would be an interesting comparison to determine how multiple alignments affected GMIL-2 and kernel-based GMIL. Experiments that apply kernel-based GMIL on *E. coli* genome data are also currently in the works.

Bibliography

- [1] M. Riley. Function of the gene products of Escherichia coli. *Microbiol. Rev.*, 57:862–952, 1993.
- [2] M. H. Saier. A functional-phylogenetic classification system for transmembrane solute transporters. *Microbiol. Mol. Biol. Rev.* 64:354–411, 2000.
- [3] T. Brown. *Molecular Biology Labfax*. Academic Press, second edition, 1998.
- [4] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [5] V. N. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Nauka, Birmingham, AL, 1979.
- [6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press. New York, 2000.
- [7] R. Karchin, K. Karplus, and D. Haussler. Classifying g-protein coupled receptors with support vector machines. *Bioinformatics* 18:147–159, 2002.
- [8] T. Jaakkola, and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems* 11, Morgan Kaufmann, San mateo, CA, 1998.
- [9] G. Deleage and B. Roux. An algorithm for protein secondary structure prediction based on class prediction. *Protein Engineering*, 1:289–294, 1987.

- [10] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997.
- [11] D. M. Engelman, T. A. Steitz, and A. Goldman. Identifying nonpolar transmembrane helices in amino acid sequences of membrane proteins. *Annual Review of Biophysics and Biophysical Chemistry*, 15:321–353, 1986.
- [12] S. A. Goldman, S. K. Kwek, and S. D. Scott. Agnostic learning of geometric patterns. *Journal of Computer and System Sciences*, 6(1):123–151, February 2001.
- [13] G. von Heijne. Membrane protein structure prediction: Hydrophobicity analysis and the positive-inside rule. *Journal of Molecular Biology*, 225:487–494, 1992.
- [14] J. Kim, E. N. Moriyama, C. G. Warr, P. J. Clyne, and J. R. Carlson. Identification of novel multi-transmembrane proteins from genomic databases using quasi-periodic structural properties. *Bioinformatics*, 16(9):767–775, 2000.
- [15] J. Kyte and R. F. Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157:105–132, 1982.
- [16] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [17] S. D. Scott, J. Zhang, and J. Brown. *On Generalized Multiple-Instance Learning*. Technical Report UNL-CSE-2003-5, Dept. of Computer Science, University of Nebraska, 2003.

- [18] Q. Tao and S. D. Scott. A faster algorithm for generalized multi-instance learning. *Proc. 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 550-555, 2004.
- [19] Q. Zhang, S. A. Goldman, W. Yu, and J. E. Fritts. Content-based image retrieval using multiple-instance learning. In *Proc. 19th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2002.
- [20] S.C.G. Rison, T. C. Hodgman, and J. C. Thornton. Comparison of functional annotation schemes for genomes. *Functional and Integrative Genomics*, 1:56-69, 2000.
- [21] R. D. King, A. Karwath, A. Clare, and L. Dehaspe. The utility of different representations of protein sequence for predicting functional class. *Bioinformatics*, 17:445-454, 2001.
- [22] F.R. Blattner, G. Plunkett, C.A. Bloch, N.T. Perna, V. Burland, M. Riley, J. Collado-Vides, J.D. Glasner, C.K. Rode, G.F. Mayhew, J. Gregor, N.W. Davis, H.A. Kirkpatrick, M.A. Goeden, D.J. Rose, B. Mau, and Y. Shao. The complete genome sequence of *Escherichia coli* K-12. *Science*, 277:1453-1461.
- [23] F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.* 65:386-407. (Reprinted in *Neurocomputing* (MIT press, 1988).).
- [24] B. Schölkopf, A. Smola, *Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.
- [25] Q. Tao, S. D. Scott, N. V. Vinodchandran, and T. Osugi. SVM-Based generalized multiple-instance learning via approximate box counting. *Proc. 21st International Conference on Machine Learning*, pages 799-806, 2004.

- [26] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining.*, pages 20-36, 1998.
- [27] W. R. Pearson, D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acda. Sci*, 85:2444-2448, 1988.
- [28] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389-3402, 1997.
- [29] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.
- [30] S. Kawashima, H. Ogata, M. Kanehisa. AAindex: amino acid index database. *Nucleic Acids Res*, 27:368-369, 1999.

Appendix A

Groupings of *E. coli* Functional Classes Tables

Table A.1: Groupings of functional classes

training set	functional classes
1	SMR family, ATP-proton motive force interconversion
2	biotin carboxyl carrier protein BCCP, Basic proteins-synthesis,modification, Plasmid-related functions
3	Glycoprotein, Ribosomes-maturation and modification, MIP family
4	Pantothenate, Misc glucose metabolism, CDFfamily, Cellkilling, Ribosomal proteins-synthesis modification Riboso, S-adenosyl methionine
5	Nucleotide interconversions, 2-Deoxyribo nucleotide metabolism, Lipoate, Cysteine, other, VIC family
6	Chorismate, Histidine, Cobalamin, Lipoprotein, Riboflavin, Proline, Outer-membrane channel, Pyridoxine, Phenylalanine
7	Aspartate, Drug analog sensitivity, Serine, ArAAP family, Surface polysaccharides antigens, Glycine, Alanine, Lipo polysaccharide
8	DAACS family, Colicin-related functions, Dcufamily, STPfamily, Glutamate
9	Lysine, Menaquinone ubiquinone, Folicacid, Tryptophan, GntP family, Sugar-nucleotide biosynthesis conversions
10	Hemeporphyrin, MATE family, Glutamine
11	DcuC, Trk system, GPH family
12	Oxidative branch pentosepwy, POT family, Arginine, SSS family, NCS2 family
13	Nucleotide hydrolysis, Amino acids, MFS family, NhaA family, Membrane-bound ATPsynthase, Thioredoxin, glutaredoxin, glutathione
14	Leucine, Detoxification, Glyoxylate bypass, Pyridine nucleotide
15	APC family of transport protein, Innermembrane, Glycolysis, SulP family
16	Asparagine, CPA2 family, Fatty acids, Sulfur metabolism, Osmotic adaptation
17	ABC superfamily atp bind, Entner-Douderoff, Valine, ABC superfamily perim, Aminosugars
18	Thiamin, Sugar-specific PTS system, Chaperones, Salvage of nucleosides and nucleotides, Chemotaxis and mobility, Molybdopterin, Electron transport, ABC superfamily membrane
19	Aerobic respiration, Non-oxidative branchpentosepwy, BCCT superfamily, Phosphoruscompounds, Surfacestructures
20	Degradation of polysaccharides, Pyruvate dehydrogenase

Table A.2: Continued: Groupings of functional classes

training set	functional classes
21	Proteins-translation and modification, Biotin, General PTS family, Fattyacid and phosphatidic acid biosynth, Poly amine biosynthesis
22	Amines, Pool multipurpose conversions of intermedmetm, FeoB family
23	Mechanism not stated, Adaptations atypical conditions, Global regulatory functions
24	Gluconeogenesis, Methionine, Polysaccharides-cytoplasmic, Threonine
25	Phospholipids, Arsfamily
26	Mureinsacculus peptido glycan, Fermentation, Aminoacyl tRNA synt RNA modifcn
27	TCAcycle, Chromosome replication
28	Degradation of proteins peptides glyco, Protein modufication, Phage-related functions and prophages, Protein modufication
29	Carbon compounds, RNDfamily, Degradation of RNA
30	Pyrimidine ribo nucleotide biosynthesis, Outer membrane constituents
31	DNA-replication repair restr modifcn, Degradation of DNA
32	Enterochelin, Purineribo nucleotide biosynthesis, Anaerobic respiration
33	Cell division, Transposon-related functions, RNA synthesis modification DNA transcriptn

Appendix B

E. coli Functional Hierarchy Tables

Table B.1: 3 levels of functional hierarchy in *E. coli*.

Level1	Level2	Level3
Cell processes	Protection responses	Drug analog sensitivity, Detoxification, Cell killing
	Transport binding proteins	SMR family, MIP family, CDF family, VIC family, Outermembrane channel, ArAAP family, DAACS family, Dcu family, STP family, GntP family, POT family, SSS family, NCS2 family, MFS family, NhaA family, Membrane-bound ATP synthase, APC family of transport protein, SulP family, CPA2 family, ABC superfamily atp bind, ABC superfamilyperi perm, Sugar-specific PTS system, ABC superfamily membrane, BCCT superfamily, General PTS family, FeoB family, Mechanism not stated, Ars family, RND family, MATE family, DcuC, Trk system, GPH family
	Chromosome replication	Chromosome replication
	Cell division	Cell division
	Chemo taxis and mobility	Chemo taxis and mobility
	Folding and ushering proteins	Chaperones
	Adaptation	Adaptations atypical conditions, Osmotic adaptation
Global regulatory functions	Global regulatory functions	Global regulatory functions
Metabolism of small molecules	Energy metabolism carbon	Anaerobic respiration, Glycolysis, Fermentation, TCA cycle, Aerobic respiration, Electron transport, Pyruvate dehydrogenase, Oxidative branch pentosepwy
	Central intermediary metabolism	Sulfur metabolism, Pool multi purpose conversions of intermed metm, Glyoxylate bypass, Sugar-nucleotide biosynthesis conversions, Entner-Douderoff, Phosphorus compounds, Salvage of nucleosides and nucleotides, Nucleotide interconversions, 2-Deoxyribo nucleotide metabolism, Amino sugars, Non-oxidative branch pentosepwy, Misc glucose metabolism, Gluco neogenesis, other, Poly amine biosynthesis, Nucleotide hydrolysis, S-adenosyl methionine
	Nucleotide biosynthesis	Purine ribonucleotide biosynthesis, Pyrimidine ribonucleotide biosynthesis

Table B.2: Continued: 3 levels of functional hierarchy in *E. coli*.

Metabolism of small molecules (continued)	Degradation of small molecules	Carbon compounds, Aminoacids, Amines, Fatty acids, ATP-proton motive force interconversion
	Amino acid biosynthesis	Asparagine, Histidine, Valine, Methionine, Lysine, Arginine, Aspartate, Leucine, Serine, Phenylalanine, Cysteine, Glycine, Chorismate, Tryptophan, Proline, Threonine, Glutamine, Glutamate, Alanine
	Biosynthesis of cofactors carriers	Pyridoxine, Folicacid, Thioredoxin glutaredoxin glutathione, Cobalamin, Riboflavin, Menaquinoneubiquinone, Molybdopterin, Biotin, Thiamin, Heme-porphyrin, Pyridinenucleotide, biotincarboxylcarrier-proteinBCCP, Enterochelin, Pantothenate, Lipoate
	Fatty acid biosynthesis	Fatty acid and phosphatidic acid biosynth
Extra chromosomal	Laterally acquired elements	Phage-related functions and prophages, Transposon-related functions, Colicin-related functions, Plasmid-related functions
Macromolecule metabolism	Macromolecule degradation	Degradation of RNA, Degradation of DNA, Degradation of proteins peptides glyco, Degradation of polysaccharides
	Macromolecule synthesis modif	Lipopoly saccharide, Proteins-translation and modification, DNA-replication repair restr modifcn, Aminoacyl tRNA syntRNA modifcn, Poly saccharides-cytoplasmic, Phospholipids, Lipoprotein, Basicproteins-synthesis modification, RNA synthesis modification DNA transcriptn, Protein modufication, Glycoprotein
Structural elements	Cell envelop	Mureinsacculus peptido glycan, Surface structures, Outer membrane constituents, Surface polysaccharides antigens, Innermembrane
	Ribosome constituents	Ribosomalproteins-synthesis modification Riboso, Ribosomes-maturation and modification

Appendix C

GPCR Functional Hierarchy Tables

Table C.1: Functional classes in level 2 of GPCR functional hierarchy.

Func. classes	Labels within each class (level 3 functional classes)
Amine	Muscarinic acetylcholine, Adrenoceptors, Dopamine, Histamine, Serotonin, Octopamine, Trace amine
Peptide	Angiotensin, Bombesin, Bradykinin, C5a anaphylatoxin, Fmet-leu-phe, APJ like, Interleukin-8, Chemokine, Cholecystokinin CCK, Endothelin, Melanocortin, Duffy antigen, Neuropeptide Y, Neurotensin, Opioid, Somatostatin, Tachykinin, Vasopressin-like, Galanin like, Proteinase-activated like, Orexin and neuropeptides FF,QRFP, Urotensin II, Adrenomedullin, GPR37/endothelin B-like, Chemokine receptor-like, Neuromedin U like, Somatostatin-and angiogenin-like peptide, Allatostatin C/ drostatin C, Melanin-concentrating hormone receptors, Prokineticin receptors, Other peptide receptors
Hormone Protein	Follicle stimulating hormone, Lutropin-choriogonadotropic hormone, Thyrotropin, Gonadotropin type I, Gonadotropin type II
Rhodopsin	Rhodopsin Vertebrate, Rhodopsin Arthropod, Rhodopsin Mollusc, Rhodopsin Other
Olfactory	Olfac II fam 1 / MOR125-138:156, Olfac II fam 2 / MOR256-262,270-285, Olfac II fam 3 / MOR255, Olfac II fam 4 / MOR225-248, Olfac II fam 5 / MOR172-224,249,254, Olfac II fam 6 / MOR103-105,107-119, Olfac II fam 7 / MOR139-155, Olfac II fam 8 / MOR161-171, Olfac II fam 9 / MOR120, Olfac II fam 10 / MOR263-269, Olfac II fam 11 / MOR106,121-122, Olfac II fam 12 / MOR250, Olfac II fam 13 / MOR253, Olfac unclassified class II, Olfac I fam 51-52 /MOR1-42, Olfac FOR-like (fish), Olfac XOR-like (frog)
Prostanoid	Prostaglandin, Prostacyclin, Thromboxane
Nucleotide-like	Adenosine, Purinoceptors
Cannabinoid	None
Platelet activating factor	None
Gonadotropin-releasing hormone	type I, type II like, other
Thyrotropin-releasing hormone and Secretagogue	Thyrotropin-releasing hormone, Growth hormone secretagogue, Growth hormone secretagogue like, Ecdysis-triggering hormone (ETHR)

Table C.2: Continued: Functional classes in level 2 of GPCR functional hierarchy.

Func. classes	Labels within each class (level 3 functional classes)
Melatonin	None
Viral	None
Lysosphingolipid and LPA (EDG)	None
Leukotriene B4 receptor	BLT 1, BLT 2
Class A Orphan/other	Platelet ADP and KI01 receptors, SREB, Mas proto-oncogene and Mas-related (MRGs), RDC1, EBV-induced, ORPH, LGR like, GPR, GPR45 like, Cysteinyl leukotriene, GP40 like

Appendix D

**Distribution of GPCR test
sequences and kernel-based GMIL
results**

Table D.1: Olfactory test label sizes.

Label	Number of	
	test sequence	correct sequence
OFOR like fish	23	-
OIfam5152MOR142	56	-
OIfam10MOR263269	25	-
OIfam11MOR106121122	6	-
OIfam12MOR250	3	-
OIfam1MOR125138156	37	-
OIfam2MOR256262270285	62	-
OIfam3MOR255	6	-
OIfam4MOR225248	56	-
OIfam5MOR172224249254	99	-
OIfam6MOR103105107120	35	-
OIfam7MOR139155	18	-
OIfam8MOR161171	33	-
OunclassifiedclassII	23	-
OXORlikefrog	4	-

Table D.2: Amine test label sizes.

Label	Number of	
	test sequence	correct sequence
Acetylcholine	15	15
Adrenoceptors	36	36
Dopamine	22	21
Histamine	12	12
Octopamine	6	3
Serotonin	33	32
Trace amine	7	7

Table D.3: Class A orphan other test label sizes.

Label	Number of	
	test sequence	correct sequence
Cysteinyl leukotriene	4	4
GP40like	3	3
GPR45like	3	3
GPR	19	19
LGRlike hormone receptors	5	5
Mas proto oncogene	2	2
Mas related receptors MRGs	11	11
ORPH	2	2
Platelet ADPKI 01 receptors	7	7
RDC1	3	3
SREB	3	3

Table D.4: Gonadotropin releasing hormone mammals test label sizes.

Label	Number of	
	test sequence	correct sequence
Gonadotropin-releasing-hormone-Mammals	4	4
Gonadotropin-releasing-hormone-others	8	7

Table D.5: Hormone protein test label sizes.

Label	Number of	
	test sequence	correct sequence
Follicle stimulating hormone	5	5
Gonadotropin type II	2	0
Gonadotropin type I	2	0
Lutropin-chorio gonadotropic hormone	3	3
Thyrotropin	6	6

Table D.6: Leukotriene B4 receptor test label sizes.

Label	Number of	
	test sequence	correct sequence
Leukotriene B4 receptor BLT1	2	0
Leukotriene B4 receptor BLT2	1	1

Table D.7: Nucleotide like test label sizes.

Label	Number of	
	test sequence	correct sequence
Adenosine	10	10
Purinoreceptors	15	15

Table D.8: Peptide test label sizes.

Label	Number of	
	test sequence	correct sequence
Adrenomedullin G 10D	1	-
Allatostatin C drostatin C	2	-
Angiotensin	10	-
APJlike	2	-
Bombesin	5	-
Bradykinin	7	-
C5a-anaphylatoxin	6	-
CCK	7	-
Chemokine receptor	4	-
Chemokine	95	-
Endothelin	8	-
Fmet-leu-phe	6	-
Galanin like	8	-
GPR37-endothelin B	4	-
Interleukin-8	9	-
Melanocortin	21	-
Neuromedin U like	7	-
Neuropeptide Y	16	-
Neurotensin	3	-
Opioid	15	-
Orexinneuro peptide FF	6	-
Proteinase-activated	6	-
So-angioge-like pep	1	-
Somatostatin	9	-
Tachykinin	11	-
Urotensin II	2	-
Vasopressin-like	13	-

Table D.9: Prostanoid test label sizes.

Label	Number of	
	test sequence	correct sequence
Prostacyclin	2	1
Prostaglandin	17	17
Thromboxane	2	2

Table D.10: Rhodopsin test label sizes.

Label	Number of	
	test sequence	correct sequence
Rhodopsin Arthropod	22	22
Rhodopsin Mollusc	3	1
Rhodopsin Other	10	10
Rhodopsin Vertebrate	70	70

Table D.11: Thyrotropin releasing hormone secretagogue test label sizes.

Label	Number of	
	test sequence	correct sequence
Growth-h-like	1	0
Growth-h	2	2
Thyro-h	5	5