

**A FAST ALGORITHM FOR GMIL AND ITS  
APPLICATION TO PROTEIN SUPER-FAMILY  
IDENTIFICATION**

by

**Jun Zhang**

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

December 9, 2003

A FAST ALGORITHM FOR GMIL AND ITS APPLICATION TO PROTEIN  
SUPER-FAMILY IDENTIFICATION

Jun Zhang, M.S.

University of Nebraska, 2003

Advisor: Stephen Scott

We develop an algorithm for a generalization of the multiple-instance learning model in which a bag's label is not based on a single instance's proximity to a single target point. Rather, a bag is positive if and only if it contains a **collection** of instances, each near one of a **set** of target points. This algorithm is significantly faster in practice than others in this model. We applied our algorithm on the protein sequence identification problem. Our goal is to identify new proteins in a superfamily with low primary sequence conservation. The low conservation of primary sequence in protein superfamilies such as Trx-fold makes conventional methods difficult to use. Therefore, we use structural properties to build our classifiers. These structural properties include secondary structure patterns as well as various properties of the residues in the protein sequences. We use this information to model proteins via our generalized multiple-instance learning algorithm. In 20-fold jack-knife tests, some of our models performed quite well, with high true positive and true negative rates. Since our techniques are very general, they should be applicable to other superfamilies with low primary sequence conservation.

## Acknowledgements

First, I would like to thank my advisor, Dr. Stephen Scott for advising me on the thesis work. His encouragement and patient direction have brought me to my graduation.

Second, I thank Dr. Jitender S. Deogun and Dr. Ashok Samal for serving on my committee. Thanks to Qingping Tao and Joshua Brown for helping me with GMIL-2. I also thanks Chang Wang, Jason Lee and Dr. Vadim Gladyshev for providing me the protein data.

Many thanks to my friends, Lan Kong, Xuemin Wu, Xuguo Zhou, Dan Li, Kefei Wang, Jin Fu, and Zhong Xu for being there and supportive.

I am greatly indebted to my family, especially to my parents. Without their understanding and blessing, I could not even start my study here in America in the first place.

## Contents

<b>Chapter</b>		
<b>1</b>	Introduction	1
<b>2</b>	Background and Related Work	4
	2.1 Conventional Multiple-Instance Learning . . . . .	4
	2.2 Generalized Multi-Instance Learning . . . . .	5
	2.3 The First Algorithm for GMIL . . . . .	7
<b>3</b>	Our Algorithm GMIL-2	11
	3.1 The Algorithm GMIL-2 . . . . .	11
	3.1.1 The Basic Algorithm . . . . .	11
	3.1.2 Building the Group Set . . . . .	13
	3.1.3 Selecting Representatives . . . . .	15
	3.2 Contrasting GMIL-1 with GMIL-2 . . . . .	16
<b>4</b>	Applying GMIL-2 to Protein Superfamily Identification	18
	4.1 Protein Superfamily Identification Problem . . . . .	18
	4.2 Property-Based Sequence Analysis . . . . .	19
	4.3 Experimental Results . . . . .	21
	4.3.1 Random Data Sets . . . . .	22
	4.3.2 Jack-Knife Tests . . . . .	23

	iii
4.4 Secondary Structure Alignment . . . . .	28
<b>5 Conclusion</b>	<b>33</b>

## Tables

### Table

4.1	Summary of the 20 positive sequences . . . . .	25
4.2	Summary of results on the jack-knife tests on the set of 20 Trx-fold proteins with motif-based alignment. “TP” is true positive rate, “TN” is true negative rate. . . . .	26
4.3	Summary of which sequences were found by each classifier in the 20-fold jack-knife test on motif-based alignment. “0” indicates a hit, “1” a miss. A Trx protein is considered correctly predicted if its total misses is $\leq 4$ . . . . .	27
4.4	Summary of which sequences were found by each classifier in the 20-fold jack-knife test. “H” indicates a hit, “M” a miss, and “NH” a near hit (E-value in (0.1, 1.0]). For SAM-based algorithms, Prim means primary structure, PS means predicted secondary structure, PSI means secondary structure is predicted by PSI-Pred, Pre means secondary structure is predicted by PREDATOR, U means using uniform prior, N means using new prior, M means hidden Markov model, SVM means using SVM method. . . . .	29
4.5	Summary of combined results on the jack-knife tests on the set of 20 Trx-fold proteins secondary structure sequences. “TP” is true positive rate, “TN” is true negative rate. . . . .	31

4.6	Summary of which sequences were found by each classifier in the 20-fold jack-knife test based on secondary structure sequence aligned on both right and reverse model, the prediction of a protein goes with the strongest prediction. “0” indicates a hit, “1” a miss. . . . .	32
-----	---	----

## Figures

### Figure

2.1	An example target concept of size $k = 3$ points, with three bags $X_1$ , $X_2$ , and $X_3$ . . . . .	6
2.2	An example of how GMIL-1 constructs its groups. . . . .	9
3.1	An example of how GMIL-2 constructs its groups. . . . .	12
3.2	An example of how GMIL-2 adds additional points. . . . .	15
4.1	Alignment of segments of five Trx-fold proteins, indexed by PDB ID. . . . .	19
4.2	Simple alignment of segments of five Trx-fold proteins. Proteins 1, 2, 3 and 5 are rescaled. . . . .	21

## Chapter 1

### Introduction

In the conventional multiple-instance learning (MIL) model, a bag's (boolean or real) label is entirely determined by a single instance in the bag, e.g. for boolean labels, the bag's label is a disjunction of the instances' boolean labels, each of which is typically determined by the instance's proximity to a single target point. But this is not sufficient for some problems. In the generalized multiple-instance learning (GMIL) model introduced by Scott et al. [29], the target concept is a **set** of points  $C = \{c_1, \dots, c_k\}$ , 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 points  $C' = \{c_{i_1}, \dots, c_{i_r}\} \subseteq C$  such that each  $c_{i_j} \in C'$  is near some point in  $B$ . Here  $r$  is a threshold indicating the minimum number of target points that must each be "hit" by some point from  $B$  (the same point in  $B$  could hit multiple target points). In other words, if we define a boolean attribute  $a_i$  for each target point  $c_i$  that is 1 if there exists a point  $b_j \in B$  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 iff 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 points and the final concept is a union of these points. If  $r = k$  then a conjunctive model exists, where each target point must be hit by some instance in  $B$ .

This learning model can be extended in an interesting way. If we let  $\bar{C} = \{\bar{c}_1, \dots, \bar{c}_{k'}\}$

be a set of “repulsion” points, then we can require a positive bag to **not** have any points near each point of  $\bar{C}' = \{\bar{c}'_{i_1}, \dots, \bar{c}'_{i_r}\} \subseteq \bar{C}$  in addition to having a point near each point in  $C'$ . This means that to be positive, certain feature values have to be present in some points of bag  $B$ , but also certain feature values must be **absent**.

Scott et al’s algorithm (which is called GMIL-1) for learning geometric concepts in the GMIL model assumes that each bag is a multiset of at most  $n$  points from the finite, discretized space  $X = \{1, \dots, s\}^d$ . It then enumerates all the possible  $N = s(s+1)/2^d$  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 point from  $B$  lies in  $b$  and  $a_b = 0$  otherwise. Then  $\bar{a}_b = 1 - a_b$ . These  $2N$  attributes are then given to Winnow (defined in section 2.3), which learns a linear threshold unit. 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  $d$ . Therefore GMIL-1 does not scale well to high dimension (e.g.  $d > 5$ ). But on low-dimensional data, GMIL-1 was shown to be competitive with (and often superior to) algorithms in the conventional multiple-instance learning model [29].

To allow scaling to higher dimensions, we created GMIL-2, which can be thought as an approximation of GMIL-1. GMIL-2 has time complexity that is polynomial in  $d$  (so it scales to higher dimensions), but also has complexity that is exponential in the size of the set  $\Psi$  of points used to partition the space  $X$ . Thus we also describe ways to keep  $\Psi$  small while not significantly increasing the generalization error of our algorithm. To analyze our algorithm, we applied GMIL-2 to the protein superfamily identification problem which has 8 dimensions.

The rest of this thesis is organized as follows. In Chapter 2 we introduce background and discuss related work, including the original algorithm GMIL-1. After that we describe our new algorithm GMIL-2 in Chapter 3. In Chapter 4 we discuss our application of GMIL-2 on protein superfamily identification problem and evaluate our algorithm. In Chapter 5

we give our conclusions [29].

## Chapter 2

### Background and Related Work

#### 2.1 Conventional Multiple-Instance Learning

The conventional MIL model was introduced by Dietterich et al. [6]. In their model, each bag (multiple-instance example) is classified as positive if and only if at least one of its elements is labeled as positive by the target concept. Their work was motivated by the problem of predicting whether a molecule would bind at a particular site. They argued empirically that axis-parallel rectangles (APRs) are good hypotheses for this and other similar learning problems. This MIL model has been extensively studied [38, 2, 21, 20, 22, 36, 26, 35, 24, 1], along with extensions for real-valued labels [7, 25]. In most MIL work, the label of a bag depends only on the label of a single point, and the label of each point typically is assumed to depend on a single target point. Exceptions include some work of Maron et al. [21, 22] in which a target concept can be a disjunction over multiple points. They also (in a subset of their experiments) mapped each pair of instances to a new instance and added spatial information about the instance pair, which defined a pairwise conjunctive type of learning model. However, they found that allowing more than 2 disjuncts in the target concept or taking more than 2 instances at a time proved computationally very difficult.

In other work, De Raedt et al. [24] generalizes MIL in the context of inductive logic programming and defines an interesting framework connecting many forms of learning [4]. One of his generalizations allows relations between instances. However, the transformations

he gives between the models have exponential time and space complexity.

## 2.2 Generalized Multi-Instance Learning

In the concept class of  $d$ -dimensional geometric patterns (introduced by Goldman et al. [10]), each **bag** (multiple-instance example) is a multi-set of points in  $d$ -dimensional space. Loosely speaking, each concept in the class can be thought of as a set of bags that visually resemble each other. The notion of resemblance between two bags  $P$  and  $Q$  is formalized by the **Hausdorff metric** [14]. The Hausdorff distance between bags  $P$  and  $Q$  is

$$\max \left\{ \max_{\vec{p} \in P} \min_{\vec{q} \in Q} \{dist(\vec{p}, \vec{q})\}, \max_{\vec{q} \in Q} \min_{\vec{p} \in P} \{dist(\vec{p}, \vec{q})\} \right\}, \quad (2.1)$$

where  $dist(\vec{p}, \vec{q})$  is the distance under some norm between points  $\vec{p}$  and  $\vec{q}$ . In other words, if each point in  $P$  reports the distance to its nearest neighbor in  $Q$  and each point in  $Q$  reports the distance to its nearest neighbor in  $P$ , then the Hausdorff distance is the maximum of these distances. A concept is the set of all bags within some distance  $\gamma$  under the Hausdorff metric of some “ideal” bag<sup>1</sup> of  $\leq k$  points (Figure 2.1 gives a one-dimensional example with  $\gamma = 1$  and  $k = 3$ ). We can generalize Equation (2.1) by allowing  $dist$  to be a weighted norm, and the weights can vary for each point in the model. By letting  $dist$  be the weighted infinity norm, a target concept  $C$  is a set of  $\leq k$  axis-parallel boxes and a bag  $B$  is positive if and only if (1) every point of  $B$  lies in some box of  $C$ , and (2) every box of  $C$  contains a point from  $B$ .

Intuitively, the above concept class captures the notion of visual similarity quite effectively. However, in pattern recognition applications, this class is not robust against noise or occlusions. Instead, often what is employed is what can be termed **ranked half-Hausdorff**. First, the distance from the model to the pattern is computed, but not vice-versa, since it is assumed that the model is accurate, but that the pattern may include

---

<sup>1</sup> In pattern recognition parlance, the ideal bag is the “model” to which we compare the “images”.

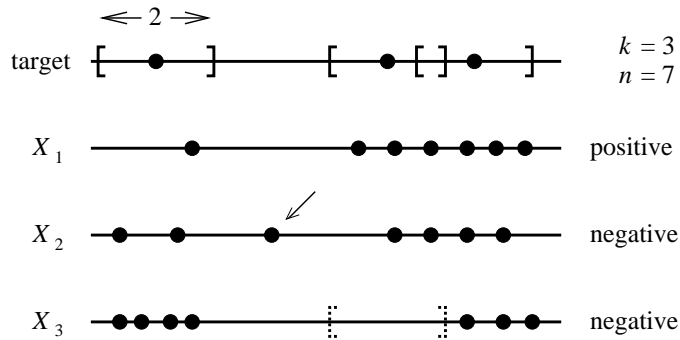


Figure 2.1: An example target concept of size  $k = 3$  points, with three bags  $X_1$ ,  $X_2$ , and  $X_3$ .

points unrelated to the model. Second, rather than taking the max over the distances from the model to the pattern, instead the  $s$ th max is taken, to improve noise tolerance. The final result can be stated as [14, 15, 30]:

$$\max_{\vec{q} \in Q}^s \min_{\vec{p} \in P} \{dist(\vec{p}, \vec{q})\} , \quad (2.2)$$

where  $\max^s$  denotes the  $s$ th max,  $P$  represents the pattern, and  $Q$  is the model (i.e. the “ideal” set of points representing the target concept). Again, if Equation (2.2) uses a weighted infinity norm with variable weights for each point in  $Q$ , a target concept  $C$  is a set of at most  $k$  axis-parallel boxes and a bag  $B$  is positive if and only if strictly fewer than  $s$  boxes of  $C$  do not contain a point from  $B$ . This is equivalent to Scott et al.’s [29] generalized MIL model with  $r = k - s$  (sans repulsion points). Adding a set  $\bar{Q}$  of repulsion points is easy as well. In addition to checking if Equation (2.2) evaluates to at most a constant  $\gamma$ , we check if the following equation evaluates to at least another constant  $\gamma'$ :

$$\min_{\vec{q} \in \bar{Q}}^{s'} \min_{\vec{p} \in P} \{dist(\vec{p}, \vec{q})\} . \quad (2.3)$$

So now in addition to requiring that a bag  $B$  misses fewer than  $s$  boxes from  $C$ , we also require that  $B$  hits at most  $s'$  boxes from  $\bar{C}$ .

### 2.3 The First Algorithm for GMIL

The first GMIL algorithm (GMIL-1) [29] (adapted from Goldman et al. [10]) assumes that each bag is a multiset of at most  $n$  points from the finite, discretized space  $X = \{1, \dots, s\}^d$ . This algorithm enumerates all the possible  $N = s(s+1)/2^d$  boxes in the space 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 point from  $B$  lies in  $b$  and  $a_b = 0$  otherwise. Then  $\bar{a}_b = 1 - a_b$ . These  $2N$  attributes are then given to Winnow [19], which learns a linear threshold unit. Winnow is very similar to the Perceptron algorithm. 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, especially in cases like ours when most inputs are irrelevant. Winnow associates a weight  $W_i$  with each boolean attribute, if the  $W_{tsum} \geq \theta$  (where  $W_{tsum}$  is the sum of  $W_i$  over all attributes,  $\theta$  is threshold of Winnow), the bag is predicted 1, otherwise predicted 0. Initially all weights  $W_i$  are set to 1. On a false negative prediction, for each attribute that is 1, Winnow promotes the weight  $W_i$  by multiplying  $W_i$  by some constant update factor  $\alpha$ . On a false positive prediction, for each attribute that is 1, Winnow demotes the weight  $W_i$  by dividing it by  $\alpha$ .

Using the above remapping of bags to boolean attributes, the complement of a target concept of geometric patterns can be represented as a monotone disjunction over the attributes, i.e. a disjunction with no negated variables. To see this, recall from Section 2.2 the two criteria that must be satisfied for a bag  $B$  to be positive: (1) each point in  $B$  must lie in some box in the target concept, and (2) each box in the target concept must contain a point from  $B$ . Let  $C = \{c_1, \dots, c_k\}$  be the boxes in the target concept and let  $C' = \{c'_1, \dots, c'_{k_{comp}}\}$  be a set of boxes whose union is exactly the complement of the union of the boxes of  $C$ . Then Criterion 1 is violated iff some box  $c'_i \in C'$  contains a point from

$B$ , i.e. if  $a_{c'_i} = 1$ . Criterion 2 is violated iff some box  $c_j \in C$  is empty, i.e. if  $\bar{a}_{c_j} = 1$ . Thus for a bag  $B$ , the following disjunction is true iff  $B$  is negative:

$$\bigvee_{c' \in C'} a_{c'} \vee \bigvee_{c \in C} \bar{a}_c ,$$

which is monotone since both  $a$  and  $\bar{a}$  are provided as original inputs to Winnow. Therefore by inverting the bags' labels and applying the remapping, the problem of learning geometric patterns is reduced to learning a monotone disjunction.

Applying a well-known result of Winnow's mistake bounds for learning monotone disjunctions allows us to conclude that Goldman et al.'s mistake bound for this problem is  $O(((k + k_{comp})d \log s))$ . However, its time complexity is  $\Omega(s^{2d})$ , which is exponential in both  $\log s$  and  $d$  (though they assumed  $d$  was constant). As an example of why this is a problem even for small  $d$ , consider the case where each point can only take on 20 values in each of 4 dimensions. Then the number of attributes to Winnow in this case is  $2 \cdot 210^4 > 3.8 \times 10^9$ . To address this issue, Goldman et al. partitioned the set of  $N$  boxes into **groups** such that it is guaranteed that for each box  $b$  in a group  $G$ , all attributes  $a_b$  have the same weight in Winnow, as do all attributes  $\bar{a}_b$ . Thus their algorithm maintains only one representative box per group  $G$  and exactly computes Winnow's weighted sum by summing the products of each group's weight and its size:

$$\sum_{i=1}^N a_i w_{a_i} + \bar{a}_i w_{\bar{a}_i} = \sum_{G \in \mathcal{G}} |G| (a_G w_{a_G} + \bar{a}_G w_{\bar{a}_G}),$$

where  $|G|$  is the number of boxes in group  $G$ ,  $a_G$  is the attribute for group  $G$ 's representative box, and  $\mathcal{G}$  is the set of all groups. The set of groups is built by using the points from all bags to rectilinearly partition  $X$  (see Figure 2.2). The result is a set of **regions** such that any pair of boxes  $b_i$  and  $b_j$  are in the same group if both have their "lower left" corners in region  $R_w$  and their "upper right" corners in region  $R_z$  (they also defined groups that had both their lower left and upper right corners in the same region). It is easy to see that when the groups are constructed this way, for each pair of boxes  $b$  and  $b'$  in the same group

$G$ ,  $b$  and  $b'$  contain the same subset of points from all bags. Thus  $a_b = a_{b'}$  and  $\bar{a}_b = \bar{a}_{b'}$  for all bags, and all the Winnow weight updates are the same for attributes  $a_b$  and  $a_{b'}$  as well as  $\bar{a}_b$  and  $\bar{a}_{b'}$ . Using this construction, at most  $O(m^{2d+1})$  groups are built, where  $m$  is the number of points used to partition the space. Thus the exponential dependence on  $\log s$  is removed, but the exponential dependence on  $d$  still remains.

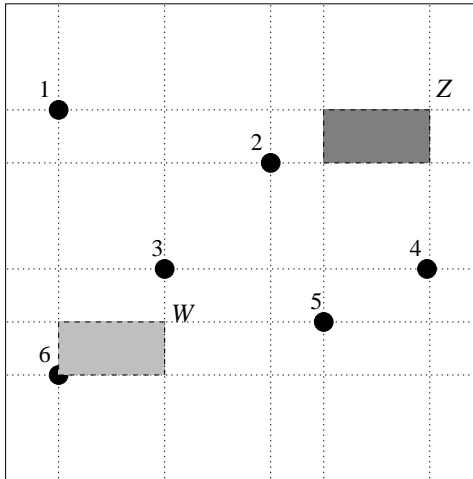


Figure 2.2: An example of how GMIL-1 constructs its groups.

The results of Goldman et al. [10] are purely theoretical. When implementing it for application to generalized multiple-instance learning problems, Scott et al. [29] made certain changes. First, they only used the attributes  $\bar{a}_b$  (which = 1 iff box  $b$  does not contain a point from the current bag), then they accommodate the half-Hausdorff metric of Equation (2.2) that generally tolerates more noise. They referred to this algorithm as “Half” and the version that uses both  $a_b$  and  $\bar{a}_b$  as “Full”, which also handles repulsion points. Second, they noted that Winnow can learn  $r$ -of- $k$  threshold functions rather than simple disjunctions with a factor of  $r$  increase in the mistake bound. Thus Half automatically handles ranked half-Hausdorff and Full automatically handles a natural definition of ranked full-Hausdorff. Third, in some experiments they used clustering on the training set to preprocess the data

to significantly reduce the number of points used to build the groups, hence speeding up the algorithm. Fourth, other heuristic modifications were made to speed up training, employing approximate methods to do bookkeeping in order to reduce training time. Finally, when training is finished, only the attributes with high weight in Winnow (“heavy groups”) are presented as the final hypothesis. This yields a more compact, interpretable hypothesis. Such a representation gives immediate information about what portions of the instance space are most relevant to classifying the bags. Pruning incrementally removes a fraction of the remaining groups each round until the the new (pruned) classifier’s false negative error rate increases unacceptably.

Scott et al. applied GMIL-1 on four different application areas: robot vision, content-based image retrieval, protein sequence identification, and drug discovery [29]. The experiments show that GMIL-1 is competitive with (and often better than) DD [21] and EMDD [36], which are two of the better algorithms in the conventional MIL model. But in these experiments GMIL-1 built around 4–7 million groups, requiring 500–700 Mb of memory and around 30 hours to train. It was also found that after training was completed, most of the groups could be thrown out with no impact on prediction error. Indeed, after training at least 80% (typically more than 97%) of the groups with the lowest weight in Winnow could be discarded and have a hypothesis that predicts nearly as well as the original. This suggests that only a relatively very small subset of groups is needed to learn these concepts well, which motivates our design of GMIL-2 described in Chapter 3.

## Chapter 3

### Our Algorithm GMIL-2

The basic learning algorithm for the GMIL model (GMIL-1) is inherently inefficient, requiring hours of computation time and hundreds of megabytes of memory to train. In this chapter a much faster and more memory-efficient algorithm (GMIL-2) is presented. GMIL-2 can handle the requirements of real pattern recognition systems.

#### 3.1 The Algorithm GMIL-2

GMIL-2 is similar to GMIL-1 in that it groups boxes together, assigns boolean attributes to these groups, and gives these attributes to Winnow to learn an  $r$ -of- $k$  threshold function over these attributes. The key difference from GMIL-1 is how GMIL-2 builds the groups. First, rather than using the union of points from all training bags, GMIL-2 uses a set  $\Psi$  of representative points that capture the distribution of the training bags. Second, rather than basing the construction of the groups on a rectilinear partition of  $X$ , GMIL-2 directly considers all subsets of  $\Psi$ .

##### 3.1.1 The Basic Algorithm

Recall that the reason we can group boxes together in GMIL-1 is that the boxes in each group contain the same set of points. For example, in Figure 2.2, the group defined by regions  $W$  and  $Z$  is a set of boxes that contains and only contains points 2, 3 and 5. Instead

of representing this group as a box, we can represent it as a set  $S = \{2, 3, 5\}$ . Any box that contains points 2, 3 and 5 and does not contain points 1, 4 and 6 belongs to this group, such as boxes  $b1$  and  $b2$  in Figure 3.1. So in this representation, any subset of  $\{1, 2, 3, 4, 5, 6\}$  is a potential group. But not all of these subsets are valid groups, e.g.  $\{4, 6\}$  since any box containing points 4 and 6 must also contain points 3 and 5. We can easily check if  $S$  is valid by finding the smallest box containing  $S$  and comparing  $S$  with the point set contained by the box. Also, if an unseen point  $p$  lies in a box of the group represented by  $S$ , the smallest box containing  $\{p\} \cup S$  must belong to this group. Another interesting fact we notice is that  $b1$  and  $b2$  are not in the same group of GMIL-1. The group defined by regions  $R_W$  and  $R_Z$  does not include  $b1$ . So we might yield a more compact group set for the same set of points. All these facts lead us to define groups in a new way.

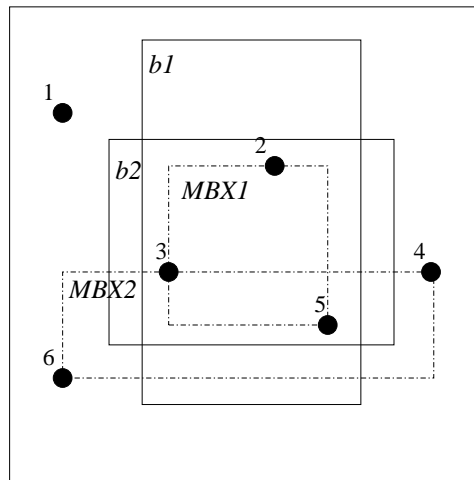


Figure 3.1: An example of how GMIL-2 constructs its groups.

Suppose a set of points  $\Psi$  is given as representatives of a distribution of points in a  $d$ -dimensional feature space  $X$ . Let  $S$  be a subset of  $\Psi$ . We call the smallest box that contains  $S$  the minimum bounding box (MBX) of  $S$ . A subset  $S \subseteq \Psi$  is *valid* if there is no point  $p \in \Psi \setminus S$  that is contained by the MBX of  $S$ , otherwise *invalid*. Each valid subset  $S$

represents a group  $G_S$  of boxes that contain and only contain  $S$  in  $\Psi$ . GMIL-2 enumerates all such groups, that is, all valid subsets of  $\Psi$ . For each group  $G_S$ , we define two attributes:  $\ell_S$  and  $\bar{\ell}_S$ . Given a bag  $B \in X^n$ , the algorithm sets  $\ell_S = 1$  and  $\bar{\ell}_S = 0$  if for some point  $p$  from  $B$ , the MBX of  $\{p\} \cup S$  does not contain any point from  $\Psi \setminus S$ . Otherwise it sets  $\ell_S = 0$  and  $\bar{\ell}_S = 1$ . These attributes are then given to Winnow, which learns some  $r$ -of- $k$  threshold function over the attributes.

### 3.1.2 Building the Group Set

GMIL-2 needs to build the group set  $\Gamma = \{G_S \mid S \subseteq \Psi \text{ and } S \text{ is valid.}\}$ . A naive algorithm scans all  $2^{|\Psi|}$  subsets of  $\Psi$  and checks if they are valid. But such an algorithm has time complexity exponential in the number of representatives.

We now describe our algorithm to build the set of groups  $\Gamma = \{G_S \mid S \subseteq \Psi \text{ and } S \text{ is valid.}\}$ . Obviously, brute-force enumeration and testing of all  $2^{|\Psi|}$  subsets of  $\Psi$  is impractical for even moderately sized  $\Psi$ . Thus our algorithm for building the set of groups exploits the geometric property that many of these  $2^{|\Psi|}$  subsets will be invalid. Our algorithm is similar to breadth-first searching of the set of all possible groups. This BFS algorithm also leads to a heuristic for choosing  $\Psi$  in such a way that allows  $|\Psi|$  to be quite large while keeping  $|\Gamma|$  small.

Our BFS algorithm first puts into  $\Gamma$  the empty set and all singleton subsets  $\{p\}$  for each  $p \in \Psi$  because all such subsets are valid. After that, it identifies all valid subsets with size 2 and adds them to  $\Gamma$ , then valid subsets with size 3, and so on up to size  $|\Psi|$ . To find all valid subsets with size  $k + 1$ , BFS looks at all size- $k$  subsets. For each size- $k$  subset  $S'$ , BFS considers each point  $p \in \Psi \setminus S'$ . Each point  $p$  is added to  $S'$  to form  $S'_p = S' \cup \{p\}$ . Then BFS builds the MBX  $b_p$  for each  $S'_p$  and adds to  $\Psi$  the largest set  $S''_p \subseteq \Psi$  contained by  $b_p$ . Note that we might have  $|S''_p| > k + 1$ . That set is still added to  $\Psi$ . For example, in Figure 3.1, adding 2 into  $\{3, 5\}$  yields a valid subset  $S'_2 = S''_2 = \{2, 3, 5\}$ . But adding 4

into  $\{3, 6\}$  yields  $S'_4 = \{3, 4, 6\}$  and  $S''_4 = \{3, 4, 5, 6\}$ . Then  $S''_4$  is added to  $\Psi$ . It tries to add a point  $p$  to each valid size- $k$  subset  $S_k$  where  $p \in \Psi \setminus S_k$ . Then it builds the MBX  $b$  for  $\{p\} \cup S_k$  and puts the largest set  $S' \subseteq \Psi$  contained by  $b$  into  $\Gamma$ . Since any size- $(k+1)$  valid subset can be built by adding one or more points to a valid subset with size less than  $k+1$ , according to Theorem 1, the algorithm is guaranteed to find all valid size- $(k+1)$  subsets.

**Theorem 1** For any valid non-empty subset  $S \subseteq \Psi$ ,  $\exists S' : S' \subset S$  and  $S'$  is valid and  $\exists p : p \in S \setminus S', MBX_S = MBX_{S' \cup \{p\}}$ .

**Proof:** Since  $S$  is not empty, there is a point  $p_1 \in S$ . Let  $S_1 = \{p_1\}$ , then  $S_1 \subset S$  and  $S$  is valid. Then we pick a point  $p_2$  from  $S \setminus S_1$  and let  $S_2 = S_1 \cup \{p_2\}$ .  $MBX_{S_2}$  must lie in  $MBX_S$ . Otherwise  $MBX_S$  is not the smallest box that contains  $S$  since  $S_2 \subset S$ . Then we build a new subset  $S'_2$  that is the set of points from  $\Psi$  that is contained by  $MBX_{S_2}$ . Since  $MBX_S$  contains  $MBX_{S_2}$ ,  $MBX_S$  must contain  $S'_2$ . So  $S'_2 \subset S$  and  $S'_2$  is bigger than  $S'_1 = S_1$  and also valid.

By repeating the above procedure, we can get a series of  $S'_i$ . At each time  $S'_i$  gets bigger and  $S \setminus S'_i$  gets smaller. So for at most  $|S|$  times, we will get an  $S'_i$  such that  $S'_{i+1} = S$  is built by adding a point  $p_{i+1}$  into  $S'_i$  unless  $S$  is not valid. Then  $S'_i$  and  $p_{i+1}$  are what we claimed in Theorem 1.

The time complexity of our BFS building algorithm is  $O(\sum_{k=1}^{|\Psi|} \rho_k |\Psi|) = O(|\Psi||\Gamma|)$ , where  $\rho$  is the number of valid size- $k$  subsets. So the computation time is decided by the total number of groups. Since  $|\Gamma| \leq \min\{2^{|\Psi|}, |\Psi|^{2d}\}$ , at worst case, the time complexity of the BFS algorithm is  $|\Psi|$  times of either that of the scanning algorithm or GMIL-1's building algorithm. That means the speed of our BFS algorithm is relatively close to other two algorithms even in the worst case if  $\Psi$  is small. If  $\Psi$  is big, the scanning algorithm and GMIL-1 both need a great amount of time. However usually not all subsets can be valid, especially when  $\Psi$  is big. Thus  $|\Gamma|$  is often much less than  $\min\{2^{|\Psi|}, |\Psi|^{2d}\}$ .

### 3.1.3 Selecting Representatives

Obviously the more representatives that are used, more accurately they represent the true distribution of the points in the bags. This tempts one to let  $\Psi$  be the union of the points in all the training bags. However this could make  $|\Gamma|$  prohibitively large. Thus we need to reduce  $|\Psi|$ . One way is clustering all the points in the training bags and setting  $\Psi$  to the set of point representatives of these clusters.

Intuitively, the larger  $\Psi$  is, the better our resolution will be of the instance space and the better our algorithm will learn. However, because point representatives of clusters will typically lie in general position (i.e. not lie on the same low-dimensional hyperplanes), increasing  $|\Psi|$  using only cluster representatives will dramatically increase  $|\Gamma|$ . To see this, note that if most subsets of the points in Figure 3.1 were collinear instead of in general position, the number of distinct valid groups would decrease significantly. This motivates our method of building  $\Psi$ : first we set  $\Psi$  to be the point representatives of a small number of clusters (e.g. 5). Then we add many random points (e.g. 100) to  $\Psi$  that are collinear with these point representatives.

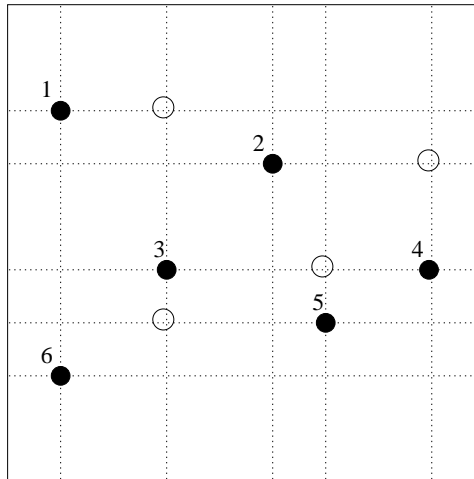


Figure 3.2: An example of how GMIL-2 adds additional points.

Specifically, first we cluster all points in training bags into  $N$  clusters. Then we build a grid  $\mathcal{G}$  such that the intervals on each dimension are defined by the projections of  $N$  clusters on this dimension. Let  $\Omega_{\mathcal{G}}$  be the set of all crossing points on  $\mathcal{G}$ . We choose the  $N$  clusters and  $K$  additional random points from  $\Omega_{\mathcal{G}}$  as our representatives. For example, in Figure 3.2, we add 4 additional points that are white. A valid subset that contains points 3 and 5 must also contain another two white points. In addition, the values of these additional points come from  $N$  clusters instead of any other random values. The heuristic is that these values should have more valuable information than any other arbitrary values if our clusters are meaningful.

### 3.2 Contrasting GMIL-1 with GMIL-2

Obviously the only difference in the running of both learning algorithms is the makeup of the group set  $\Gamma$ . Here we present some more detailed comparisons between the group sets under various special cases of the representative sets.

First consider when  $\Psi$  for GMIL-2 is the union of all points in all training bags. Let  $S$  be a valid subset of  $\Psi$  and let  $\Gamma_1^S$  be the union of all groups for GMIL-1 whose boxes contain exactly  $S$ . Then the GMIL-2 group represented by  $S$  is exactly equal to  $\Gamma_1^S$ ,  $\ell_S = a_G$  and  $\bar{\ell}_S = \bar{a}_G$  for all training bags, and the two algorithms are equivalent. The only difference is that the set of groups (i.e. the set of attributes given to Winnow) is smaller for GMIL-2.

One special case of GMIL-2 is putting all points in training bags into  $\Psi$ . Then each valid subset  $S$  is equal to a group  $G$  of GMIL-1 whose representative box is the MBX of  $S$ . Both groups contain the boxes that contain the same set of points of training bags. That means  $\ell_S = a_G$  and  $\bar{\ell}_S = \bar{a}_G$  for all training bags. We see that the set of groups built in GMIL-2 is identical to those built in GMIL-1, when taken with respect to training since Winnow can learn the same threshold function for both algorithms. However the number of attributes of GMIL-2 is often less than GMIL-1 because GMIL-2 can put all

boxes containing the same set of points into one group, which GMIL-1 cannot guarantee. For example, in Figure 3.1,  $b1$  and  $b2$  are not in the same group of GMIL-1 although they contains the exactly same set of points. Thus GMIL-2 would get a more compact group set than GMIL-1 in this case.

Now consider the case when GMIL-2 builds its grid on a set of representative points rather than the union of all points in all bags. Now since the points in Figure 3.1 are only representatives, it is possible that there is e.g. a point  $x$  from the training bags that lies in box  $b1$  but not in  $b2$ . From its use of the grid to build its group set, GMIL-1's groups can recognize this fact, but GMIL-2's subset-based construction of  $\Gamma$  prevents this. Thus GMIL-2 operates in a space with lower resolution than GMIL-1, and the only way it can match GMIL-1's resolution is to set  $\Psi = \Omega_G$ . If this is the case, then the two algorithms are again equivalent.

Thus for special cases, the two algorithms build equivalent group sets. If only a subset of all training points or grid points is used as representative points, GMIL-2 can be treated as an approximation to GMIL-1. The time complexity of GMIL-2 is polynomial in  $d$  but in the worst case exponential in  $|\Psi|$ . Thus we can now scale up our algorithm to high dimensions without much of a time complexity increase. However, we obviously cannot use all points from the training bags, which means we learn on a reduced resolution. It has been confirmed by experiments on smaller-dimensional data that GMIL-2 is significantly faster than GMIL-1 [32]. Tao et al.'s results show that GMIL-1 requires 500-700 Mb of memory and ran for around 30 hours to train on a single data set. In contrast, in the same experiments GMIL-2 used less than 50 Mb of memory and finished training in less than one hour on average with no statistically significant change in generalization error. We now test to see if GMIL-2 is also suitable for high-dimensional data. We applied GMIL-2 on protein super family identification problem in which case  $d = 8$ . In the following chapter I will discuss the protein super family identification problem and present the experiment results.

## Chapter 4

### Applying GMIL-2 to Protein Superfamily Identification

#### 4.1 Protein Superfamily Identification Problem

Protein families generally consist of proteins with similar function, and superfamilies are collections of related families. Generally, proteins are added to databases much faster than they can be tested to determine to which superfamily they belong. Thus a fundamental problem in computational biology is searching protein databases to find candidate members of a specific superfamily, which can then be tested in the lab to verify membership.

Many successful search methods are based on hidden Markov models (HMMs, e.g. [8]) built on the amino acid sequences of known members of the superfamily. This method is useful if the superfamily exhibits primary sequence conservation, i.e. if the sequences are similar to each other in their chains of amino acids (represented by letters from a 20-character alphabet). However, some superfamilies of proteins, such as thioredoxin fold proteins (Trx), lack this property, i.e. they have low primary sequence conservation and are only similar in the higher-order structures that they “fold” into. For example, in Figure 4.1, segments of five Trx-fold proteins are shown with each sequence identified by its ID from the Protein Data Bank (<http://www.rcsb.org/pdb/>). Only the two cysteines (C, marked by asterisks) are fully conserved in the alignment, and very little else is even partially conserved. These two cysteines form a redox motif designated by the CxxC motif. This is typical of Trx-fold proteins.

```

* *
1A8L:  . . .KLIVFVRKDHQCQCDQLKQLVQEL. . .
1BED:  . . .PVVSEFFSFYCPHCNTFEPITIAQL. . .
1QK8:A  . . .LVFFYFSASWCPPCRGFTPQLIEF. . .
1F9M:A  . . .PVVLDMFTQWCGPCKAMAPKYEKL. . .
1MEK:  . . .YLLVEFYAPWCGHCKALAPEYAKA. . .

```

Figure 4.1: Alignment of segments of five Trx-fold proteins, indexed by PDB ID.

The low conservation of primary sequence in protein superfamilies such as Trx-fold makes conventional methods difficult to use. Therefore, in this work we use structural properties to build our 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.

## 4.2 Property-Based Sequence Analysis

Recently, Kim et al. [17] studied G Protein-Coupled Receptors in the following way. They processed each candidate protein, computing for each amino acid seven properties: GES hydropathy index [9, 11], solubility [3], polarity, pI, Kyte-Doolittle index [18],  $\alpha$  helix index [5], and molecular weight. For each property, one value is computed per amino acid, so the sequence of  $n$  amino acids is transformed to a sequence of  $n$  numbers. Kim et al. then 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. Their results were good for GPCRs, but Trx has been much more difficult to learn in a single-instance model [28], because mapping the sequences to their summary statistics loses significant information that reveals fundamental properties of the proteins (e.g. that in all the Trx-fold proteins, positions  $k$  through  $\ell$  have high solubility). Such information is not only useful in identifying new members of the family, but it also may reveal to a biochemist in more detail why certain proteins belong to a family and others

do not. This information might also be useful in other applications such as predicting the function of a specific protein by measuring similarities of a candidate protein’s signature to the signatures of proteins with known function.

When mapping the property sequences to multiple-instance examples, the first coordinate of point  $p$  in bag  $P$  relates 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. Since the lengths of the sequences vary significantly (even among only the positive sequences), correctly aligning them is a challenge. This problem is compounded by the low primary sequence similarity, which prevents us from using conventional multiple sequence alignment algorithms.

To answer the problem of setting the first coordinate, we used a naïve and simple mechanism to align the sequences in our experiments. First we aligned the two conserved cysteines (marked by asterisks in Figure 4.1) in all sequences, since this CxxC motif (or a similar one) appears in all Trx-fold proteins. Then (since it is known that all Trx-fold proteins extend at most 180 amino acids beyond the motif) we used the next 180 symbols of each sequence, discarding everything else that lay beyond that point. If a sequence was not long enough to go 180 symbols past the CxxC, it was linearly rescaled so that the last symbol was in position 180. Finally, since it is also known that Trx-fold proteins extend at most 20 positions upstream of the motif, we also used these 20 positions, yielding a sequence of length at most (and often strictly less than) 204, mapped to a space that spans  $[1, 204]$ . Examples are shown in Figure 4.2.

To keep the 7 properties of each amino acid and its position information, the primary

```

* *
123456789012345678901234567890123456789012345679
1  HDEVVLWKHDFIVVEFYAPWCGHC-----K-----.....
2  FKNVVLESSVPVLVDFWAPWCGPC--R-I--I-A--P-V--V.....
3  -----AQTIVIFGRSGCPYC--V--R-A--K-D--L-A.....
4  NFEQALAAHRHLLVEFYAPWCGHCKALAPEYAKAAAQLKAEG.....
5  MESVRSLVEDKPVVIFSKSSCCMC--H-S-I--Q-T-L--I-.....

```

Figure 4.2: Simple alignment of segments of five Trx-fold proteins. Pproteins 1, 2, 3 and 5 are rescaled.

sequence of  $n$  amino acids is transformed to a sequence of  $n * 8$  numbers. Each amino acid is mapped to 8 numbers, the first number is its position in the sequence, and the remaining 7 numbers are its 7 properties. Therefore each amino acid is mapped to a point in an 8 dimensional space, the entire sequence is a set of multiple points in that 8-dimensional space. This nicely fits the generalized multi-instance learning model. Then we smoothed these property sequences with a size-16 Gaussian kernel to filter out noise before we use these profiles sequences as inputs to the GMIL-2. The “kernel” for smoothing defines the shape of the function that is used to take the average of the neighbouring 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 raw profile is smoothed by applying the Sliding Window Recognizer [33], which transforms the profile 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, in our experiments  $s = 16$ .

### 4.3 Experimental Results

We ran two tests of our algorithm on protein data. In both tests, we filtered our data to reduce primary sequence similarities, since the goal of this application is to identify new

Trx sequences that are highly dissimilar to known ones.

### 4.3.1 Random Data Sets

As a first test of GMIL-2, we applied it on large, random data sets, constructed as follows. First we filtered our positive and negative sets such that no two sequences were more than 80% similar [34], finally yielding 183 positives and 197 negatives. We then split our data into three sets of approximately equal sizes:  $A$ ,  $B$ , and  $C$ . Kim et al.'s [17] seven properties (listed in Section 4.2) were computed for each amino acid in each sequence in each set, the position information of each amino was extracted from the aligned sequences, and added into the sequence property files. Then we ran 3 experiments: training on each pair of sets from  $\{A, B, C\}$  and testing on the third. Our results were a true positive rate around 0.74 and true negative rates around 0.88.

In contrast, we summarize Wang et al.'s results [34], who applied other approaches to this problem. They used the hidden Markov model tool SAM on the original (primary) sequences (a common approach in biological sequence analysis) and on predicted and true secondary structures of the sequences. A protein's secondary structure is a linear sequence of these higher-order structures. When a protein folds on itself in three-dimensional space, amino acids that are near each other together form various higher-order structures, such as  $\alpha$  helices and  $\beta$  sheets [31]. Also, similar to Kim et al., they computed summary statistics of the properties (e.g. mean and variance of the derivatives) to map the multiple-instance data that we used to single-instance data, which they gave to a support vector machine. HMM trained on the primary sequences had true positive and true negative rates around 0.99, HMM trained on predicted secondary structure had both true positive and true negative rates around 0.82, HMM trained on true secondary structure had both true positive and true negative rates around 0.70, and SVM had both true positive rate around 0.81 and 0.85 for true the negative rate. The sequences are so similar that standard HMM-based

approaches perform slightly better. This implies that in fact 80% inter-sequence similarity is high enough for HMM on primary sequence to be the preferred modeling technique. Since the goal of our work was to identify new, highly-dissimilar sequences (i.e. ones that cannot be detected by HMM on primary sequence), we conducted another test. In the next section we describe our jack-knife test on highly dissimilar sequences.

### 4.3.2 Jack-Knife Tests

Within our data set, there are many similar sequences, which means that the experiments of Section 4.3.1 are inappropriate to evaluate our methods for the purpose they were designed: to identify new families that are highly dissimilar to known ones, i.e. identify sequences that primary sequence-based HMMs cannot. Since our goal is to identify new families, the sequences in our data set should be highly dissimilar to each other. Thus we constructed a new positive set of Trx-fold proteins such that primary sequence conservation between each pair of sequences was so low that SAM was unlikely to identify any one with a model built on the rest. We started by randomly selecting one sequence from a set  $S$  of 1100 known Trx-fold sequences, placing it in our positive set  $P$ , and built an HMM  $M$  on  $P$  using SAM. We then used  $M$  to score the other 1099 Trx-fold sequences from  $S \setminus P$  (“\” denotes set difference, i.e. those sequences that are in  $S$  but not in  $P$ ) and added to  $P$  the one with the highest E-value (i.e. the least similar one). We then iteratively built a new HMM on  $P$ , scored the remaining sequences in  $S \setminus P$ , and added to  $P$  the sequence with largest E-value until  $|S| = 25$ . We chose as our new positive data set the 20 most distinct sequences that this procedure found (see Table 4.1) also of very low similarity. Pairwise identity of sequences from the set of 20 ranged from 0.32 to 0.82, averaging 0.48, and a jack-knife test using HMM on primary structure only found one of these sequences. The set of non-Trx-fold proteins remained the same as Section 3.1. In these 20 sequences, 17 of them have the CxxC motif, 2 of them have the CxxS motif, and the final sequence has

neither CxxC nor CxxS motif. Due to the small number of Trx-fold proteins available in our data set, we performed a jack-knife (leave-one-out cross-validation) test. We held out one Trx-fold protein for use in testing and used the rest for training, repeating once for each of the 20 Trx-fold proteins in the data set. Since the multiple-instance learning algorithm requires both Trx-fold and non-Trx-fold proteins for training, we split the non-Trx-fold proteins set into 8 sets of approximately equal sizes (20 sequences in each set), then trained our algorithms on the 19 Trx-fold proteins plus one of the 8 sets of non-Trx-fold proteins, and tested on the held-out Trx-fold protein plus the remaining 7 sets of non-Trx-fold proteins. We repeated this for each of the 8 sets of non-Trx-fold proteins. Thus totally we ran  $20 \times 8 = 160$  experiments. So all error rates reported are only on sequences that were not used to build the models.

As we discussed in Chapter 3, if we use all points in all the training examples for our set  $\Psi$ , this could make the total number of groups  $|\Gamma|$  prohibitively large. Thus we clustered all the points in the training examples into  $N$  clusters, then we added  $K$  additional random points ( $N = 5$ ,  $K = 80$  in our experiments), using these points as our representatives  $\Psi$  to build group sets for GMIL-2.

The results are summarized in Table 4.2, true positive and true negative error rates are reported for each hold out negative set, the error rate is the average error rate over all 20 runs. The true negative rate is 75%. The true positive rates in the Table 4.2 are the fractions (out of 20) of the set of Trx-fold proteins that GMIL-2 correctly identified. Since each held out Trx-fold protein was used as a test example in 8 experiments (one for each negative set), we gave GMIL-2 credit for correctly classifying the held-out positive if that held-out positive example was successfully identified at least half the time.

Table 4.3 reports the performance of GMIL-2 on each of the 20 Trx-fold proteins from the jack-knife test. A “0” in an entry indicates the that protein was successfully identified from the testing set. Each Trx-fold protein was tested 8 times because GMIL-2 was trained

Table 4.1: Summary of the 20 positive sequences

accession number	motif	putative class
gi: 13400018	SxxC, CxxC	Thioredoxin
gi: 14602058	CxxC	Thioredoxin
gi: 19698793	CxxC	Thioredoxin
gi: 2194076	CxxC	Thioredoxin
gi: 7512732	CxxC	Thioredoxin
gi: 443281	CxxC	Thioredoxin
gi: 1076496	CxxC	PDI
gi: 840745	CxxC	Thioredoxin
gi: 1421133	CxxC	Gluteredoxin
gi: 129727	CxxC, CxxC	PDI
gi: 15229353	CxxS,	Gluteredoxin
gi: 14787802	KxxC,	PDI
gi: 14729415	CxxC, CxxC	PDI
gi: 13122603	CxxC,	Gluteredoxin
gi: 11494247	CxxC,	Thioredoxin
gi: 15150492	CxxC,	Gluteredoxin
gi: 13358154	CxxC,	Thioredoxin
gi: 24372070	CxxC,	Thioredoxin
gi: 23483739	CxxS, CxxS	PDI
gi: 16763418	CxxC,	PDI

Table 4.2: Summary of results on the jack-knife tests on the set of 20 Trx-fold proteins with motif-based alignment. “TP” is true positive rate, “TN” is true negative rate.

neg set	TP	TN
1	0.60	0.79
2	0.45	0.67
3	0.65	0.75
4	0.55	0.75
5	0.75	0.75
6	0.50	0.75
7	0.65	0.75
8	0.45	0.76

on the other 19 Trx-fold proteins and 8 different negative sets, one for each set. The last column of the table is the total number of misses of that protein in all experiments. It shows 15 Trx-fold proteins were successfully identified and 5 were not, so the overall true positive rate of GMIL-2 trained on property profiles of motif-based alignment sequences is 75%.

In contrast, we summarize results from Wang et al. [34], who applied other approaches to this problem. They used HMM, SVM on the primary sequences and on predicted secondary structures of the sequences. Their results are summarized in Table 4.4. The SVM model and the model built on predicted secondary structure (PSI-PRED [16], New Prior) were the overall best performers, correctly identifying 10/20 and 10/20 positives and over 0.885 and 0.81 of the negatives. Our algorithm could identify 15/20 positives, outperforming all their experiments.

Table 4.3: Summary of which sequences were found by each classifier in the 20-fold jack-knife test on motif-based alignment. “0” indicates a hit, “1” a miss. A Trx protein is considered correctly predicted if its total misses is  $\leq 4$ .

accession number	Negative Set								miss
	1	2	3	4	5	6	7	8	
gi:443281	1	1	0	0	0	0	0	1	<b>3</b>
gi:1076496	1	0	1	1	0	1	1	0	5
gi:840745	0	0	0	0	0	0	0	1	<b>1</b>
gi:1421133	1	1	1	1	1	1	1	0	7
gi:129727	0	0	0	0	0	1	0	0	<b>1</b>
gi:15229353	0	1	0	0	0	0	0	1	<b>2</b>
gi:14787802	1	0	1	1	1	1	1	0	6
gi:13400018	0	1	0	0	0	0	1	1	<b>3</b>
gi:14729415	0	1	1	0	0	1	0	1	<b>4</b>
gi:13122603	0	0	0	0	0	0	0	0	0
gi:11494247	1	0	1	0	1	1	0	0	<b>4</b>
gi:15150492	0	1	0	1	1	0	0	1	<b>4</b>
gi:13358154	0	0	0	1	1	0	1	0	<b>3</b>
gi:24372070	0	1	0	1	0	1	0	0	<b>3</b>
gi:23483739	1	0	0	0	0	0	1	1	<b>3</b>
gi:16763418	0	1	1	0	0	0	0	0	<b>2</b>
gi:14602058	0	1	1	1	0	1	0	1	5
gi:19698793	0	1	0	0	0	0	0	1	<b>2</b>
gi:2194076	1	1	0	1	0	1	1	1	6
gi:7512732	1	0	0	1	0	1	0	1	<b>4</b>
sum	8	11	7	9	5	10	7	11	

#### 4.4 Secondary Structure Alignment

In the experiments of Section 4.3, we extract the position information of each amino acid from the motif-based alignment of primary sequences, using this position information as the first coordinates of points of multi-instance learning examples. But the linear rescaling of the subsequence downstream of the motif seems inappropriate since the Trx-fold of each sequence is of a different length. I.e. our alignment mechanism probably does not align some critical subsequences very well. Since secondary structure of Trx-fold proteins is conserved, it is natural to try to use this information to align our sequences.

A protein's secondary structure is a linear sequence of higher-order structures. When a protein folds on itself in three-dimensional space, amino acids that are near each other together form various higher-order structures, such as  $\alpha$  helices and  $\beta$  sheets [31]. In addition to the conserved CxxC motif, for secondary structure, three  $\alpha$ -helices and four  $\beta$ -sheets are organized in a specific pattern (a  $\beta$ - $\alpha$ - $\beta$ - $\alpha$ - $\beta$ - $\beta$ - $\alpha$  motif). For most sequences, the CxxC motif is located between the first  $\beta$ -strand and the first  $\alpha$ -helix in the fold, so the entire motif is  $\beta$ -CxxC- $\alpha$ - $\beta$ - $\alpha$ - $\beta$ - $\beta$ - $\alpha$  [23, 13]. Therefore, even though the protein primary sequences are not conserved and thus primary sequence alignment could not get valuable position information, one can use the structural pattern to align these proteins. It should be noted, however, that some Trx-fold proteins allow insertions and deletions of secondary structures, which complicate matters. Imperfect secondary structure prediction brings more complication.

For each sequence, we first used PSI-PRED [16] to predict the secondary structure of each sequence. The secondary structure sequence of proteins were aligned using SAM [12]. Since each residue in secondary structure sequence of a protein corresponds to an amino acid in its primary sequence, for every amino acid of the truncated primary sequence obtained in the above experiments, we could find its position in the secondary structure sequence.

Table 4.4: Summary of which sequences were found by each classifier in the 20-fold jack-knife test. “H” indicates a hit, “M” a miss, and “NH” a near hit (E-value in (0.1, 1.0]). For SAM-based algorithms, Prim means primary structure, PS means predicted secondary structure, PSI means secondary structure is predicted by PSI-Pred, Pre means secondary structure is predicted by PREDATOR, U means using uniform prior, N means using new prior, M means hidden Markov model, SVM means using SVM method.

accession number	M Prim	M PS PSI N	M PS PSI U	M PS Pre N	M PS Pre U	SVM PS PSI	SVM PS Pre
gi: 13400018	M	M	M	M	M	M	H
gi: 14602058	M	NH	NH	H	M	M	M
gi: 19698793	M	H	H	H	M	M	M
gi: 2194076	M	H	H	M	M	M	M
gi: 7512732	M	M	H	H	H	H	H
gi: 443281	H	H	H	H	M	H	M
gi: 1076496	M	M	M	M	M	M	M
gi: 840745	M	H	H	NH	M	H	M
gi: 1421133	M	H	M	NH	M	H	M
gi: 129727	M	H	M	NH	M	M	H
gi: 15229353	M	M	M	M	M	M	M
gi: 14787802	M	M	M	M	M	H	M
gi: 14729415	M	M	M	M	M	H	M
gi: 13122603	M	M	M	M	M	H	H
gi: 11494247	M	H	M	H	M	H	M
gi: 15150492	M	M	M	M	M	H	H
gi: 13358154	M	H	H	M	M	M	M
gi: 24372070	M	M	M	NH	H	M	M
gi: 23483739	M	H	NH	M	M	M	H
gi: 16763418	M	H	M	NH	M	H	H

This position information was then added to the same 7 properties we used in Section 4.2 to replace the previous position information obtained from motif-based alignment.

If our classifier were used to search a sequence database, we obviously would not know which alignment (positive or negative) to align candidate sequence  $S$  to. So we would align  $S$  to both, getting bags  $S_+$  and  $S_-$ . Each would be labeled by our classifier, and the prediction that is most confident ( $|W_{tsum} - \theta|$ ,  $W_{tsum}$  is the sum of  $W_i$  over all attributes,  $\theta$  is the threshold of Winnow) is the final prediction. For example, if a candidate sequence is aligned to positive alignment, and is predicted “Trx” with  $W_{tsum} = 157602$ ,  $\theta = 174962$ , so  $|W_{tsum} - \theta| = 17360$ . Then we aligned it to negative alignment, it is predicted “non-Trx” with  $W_{tsum} = 190575$ ,  $|W_{tsum} - \theta| = 15613$ , so the final prediction is negative.

We realigned the secondary structure sequences of Trx-fold proteins and non-Trx-fold proteins. The secondary structure sequence of Trx-fold proteins were aligned based on the model of non-Trx-fold proteins and build non-Trx-fold proteins secondary structure sequence based on the model of Trx-fold proteins. Then we repeated the jack-knife test of Section 4.4. The results were summarized in Table 4.5 and Table 4.6. Predictions predicted positive by both classifiers were predicted Trx-fold proteins with high confidence. For those that were predicted non-Trx proteins by both classifiers, we predicted those non-Trx. For those that disagree between the two classifiers, we compared their confidences and went with the strongest prediction. The final results of our jack-knife test are summarized in Table 4.5 and Table 4.6. The average true negative rate is 65%, 14 out of 20 Trx-fold proteins were identified successfully at least half the time, so the true positive rate is 70%.

Compared with Wang et al.’s best results (SVM) whose TP is 50% and TN is 0.885% [34], our GMIL-2 has better performance on TP rate. Since our techniques are very general, they should be applicable to other superfamilies with low primary sequence conservation.

We also did the similar test on secondary structure of random data sets, our results were a true positive rate around 0.16 and true negative rates around 0.89, while Wang et

Table 4.5: Summary of combined results on the jack-knife tests on the set of 20 Trx-fold proteins secondary structure sequences. “TP” is true positive rate, “TN” is true negative rate.

neg set	TN
1	0.8541
2	0.6473
3	0.5209
4	0.5503
5	0.6564
6	0.6659
7	0.5389
8	0.8004
average	0.654275

al.’s HMM on same data set could achieve 0.82 true positive and true negative rate. So for those proteins whose sequences are similar, our GMIL-2 can not outperform HMM, but have better performance on very dissimilar sequences.



## Chapter 5

### Conclusion

While the standard MIL model is powerful, there exist applications with natural target concepts that cannot be represented. This is what motivated the prior introduction of Scott et al.'s GMIL model, along with the algorithm GMIL-1 to learn geometric concepts in it. Here we presented GMIL-2, a much faster algorithm (in practice) than GMIL-1. GMIL-2 uses a compact group set based on a set of representative points. It has the same generalization ability as GMIL-1 and needs much less time and memory to train [32].

To test GMIL-2 with high-dimensional data, we applied it to protein super family identification problem. The Trx-fold super family is a very important set of proteins with very low similarity in primary sequence. In this work we have applied generalized multiple instance learning to this problem, focusing on structural information rather than primary sequence. The GMIL-2 model built on the properties profile using primary sequence position information could achieve the positive rate 75% and true negative rate 75%, while Want et al.'s HMM could achieve 98% true negative rate but only 5% true positive rate. The GMIL-2 model built on the properties profile using secondary structure sequence position information could achieve the positive rate 70% and true negative rate 65%, Want et al.'s SVM could achieve 88.5% true negative rate but only 50% true positive rate [34].

Because the experimental results are affected by the multiple alignment of proteins, in future we could try different alignment tools to achieve better results. The other future

work is choosing different points representatives. The selection of  $\psi$  will affect the resolution and the number of group sets. So far we are using the cluster points and its collinear random points, in the future we could try different selection of points representatives

## Bibliography

- [1] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15*, 2002.
- [2] P. Auer, P. M. Long, and A. Srinivasan. Approximating hyper-rectangles: Learning and pseudo-random sets. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 314–323. ACM, 1997.
- [3] T. Brown. *Molecular Biology Labfax*. Academic Press, second edition, 1998.
- [4] L. De Raedt. Attribute-value learning versus inductive logic programming: The missing links. In *Proc. 8th International Conference on Inductive Logic Programming*. Springer Verlag, 1998.
- [5] G. Deleage and B. Roux. An algorithm for protein secondary structure prediction based on class prediction. *Protein Engineering*, 1:289–294, 1987.
- [6] 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.
- [7] D. R. Dooly, Q. Zhang, S. A. Goldman, and R. A. Amar. Multiple-instance learning of real-valued data. *Journal of Machine Learning Research*, 3(Dec):651–678, 2002.
- [8] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

- [9] D. M. Engelman, T. A. Steitz, and A. Goldman. Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins. *Annual Review of Biophysics and Biophysical Chemistry*, 15:321–353, 1986.
- [10] 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.
- [11] G. Von Heijne. Membrane protein structure prediction: Hydrophobicity analysis and the positive-inside rule. *Journal of Molecular Biology*, 225:487–494, 1992.
- [12] R. Hghey and A. Krogh. Hidden markov models for sequence analysis: Extension and analysis of the basic method. *CABIOS*, 12:95–107, 1996.
- [13] A. Holmgren and M. Bjornstedt. Thioredoxin and thioredoxin reductase. *Methods in Enzymology*, 252:199–208, 1995.
- [14] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:850–863, September 1993.
- [15] D. P. Huttenlocher and W. J. Rucklidge. *A multi-resolution technique for comparing images using the Hausdorff distance*. Technical Report 92-1321, Dept. of Computer Science, Cornell University, 1992.
- [16] K. Bryson and L. J. McGuffin and D. T. Jones. 2000.
- [17] 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.
- [18] 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.

- [19] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [20] O. Maron. *Learning from Ambiguity*. PhD thesis, Dept. of Electrical Engineering and Computer Science, M.I.T., June 1998.
- [21] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems 10*, 1998.
- [22] O. Maron and A. L. Ratan. Multiple-instance learning for natural scene classification. In *Proc. 15th International Conf. on Machine Learning*, pages 341–349. Morgan Kaufmann, San Francisco, CA, 1998.
- [23] J. Martin. Thioredoxin—a fold for all reasons. *Structure*, 3:245–250, 1995.
- [24] J. Ramon and L. De Raedt. Multi instance neural networks. In *Proceedings of the ICML-2000 Workshop on Attribute-Value and Relational Learning*, 2000.
- [25] S. Ray and D. Page. Multiple instance regression. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 425–432, 2001.
- [26] G. Ruffo. *Learning single and multiple instance decision trees for computer security applications*. PhD thesis, Dept. of Computer Science, University of Turin, Torino, Italy, 2000.
- [27] Steven L. Salzberg. Book review: C4.5: Programs for machine learning by ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16:235–240, 1994.
- [28] S. D. Scott, H. Ji, P. Wen, D. E. Fomenko, and V. N. Gladyshev. *On modeling protein superfamilies with low primary sequence conservation*. Technical Report TR-UNL-CSE-2003-4, Department of Computer Science, University of Nebraska, January 2003.

- [29] 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.
- [30] D.-G. Sim, O.-K. Kwon, and R.-H. Park. Object matching algorithms using robust Hausdorff distance measures. *IEEE Transactions on Image Processing*, 8(3):425–429, 1999.
- [31] L. Stryer. *Biochemistry*. W. H. Freeman and Company, fourth edition, 1995.
- [32] Q. Tao and S. Scott. A fast algorithm for generalized multi-instance learning. *Submitted*, 2003.
- [33] J. L. Tukey. *Exploratory data analysis*. Addison Wesley, 1977.
- [34] C. Wang, S. D. Scott, J. Zhang, Q. Tao, D. E. Fomenko, and V. N. Gladyshev. On modeling protein superfamilies with low primary sequence conservation. *In Preparation for Submission*, 2003.
- [35] J. Wang and J. D. Zucker. Solving the multiple-instance problem: A lazy learning approach. In *Proc. 17th International Conf. on Machine Learning*, pages 1119–1125, 2000.
- [36] Q. Zhang and S. A. Goldman. EM-DD: An improved multiple-instance learning technique. In *Neural Information Processing Systems 14*, 2001.
- [37] Q. Zhang and S. A. Goldman. *EM-DD: An improved multiple-instance learning technique*. Technical Report WUCS-01-17, Washington University in St. Louis, July 2001.
- [38] 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.