On Demand Phenotype Ranking through Subspace Clustering

Xiang Zhang, Wei Wang
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599, USA
{xiang, weiwang}@cs.unc.edu

Jun Huan
Department of Electrical Engineering
and Computer Science
University of Kansas
Lawrence, KS 66047, USA
jhuan@eecs.ku.edu

Abstract

High throughput biotechnologies have enabled scientists to collect a large number of genetic and phenotypic attributes for a large collection of samples. Computational methods are in need to analyze these data for discovering genotype-phenotype associations and inferring possible phenotypes from genotypic attributes. In this paper, we study the problem of on demand phenotype ranking. Given a query sample, for which only its genetic information is available, we want to predict the possible phenotypes it may have, ranked in descending order of their likelihood. This problem is challenging since genotype-phenotype databases are updated often and explicitly mine and maintain all patterns is impractical. We propose an on-demand ranking algorithm that uses a modified pattern-based subspace clustering algorithm to effectively identify the subspaces where these relevant clusters may reside. Using this algorithm, we can compute the clusters and their prediction significance for any phenotypes on the fly. Our experiments demonstrate the efficiency and effectiveness of our algorithm.

1 Introduction

Current high-throughput techniques in biological and biomedical research generate massive heterogenous genetic and phenotypic data rapidly. The identification of genotype-phenotype associations is essential in biological research for understanding complex biological systems. An important problem studied in this paper is to predict the phenotypes of a new individual from its genetic information [1, 10, 5, 2]. Traditional classification methods [3, 4, 8] focus on building a model for a single target attribute (phenotype). However, in real applications, there are usually a large number of phenotypes [7, 9], among which we want to identify those that are likely to be positive for an individual. Due to the large number of phenotypes, it is impractical for the domain experts to explicitly exam all possible phenotypes for a new sample. A reasonable way to solve this problem is to rank the phenotypes in descending order of which the new sample is likely to have. Then the domain experts can prioritize their effort guided by the ranking. Since the traditional classification methods focus on building a model for a single phenotype, they are not readily applicable to this problem. In this paper, we address the problem of ranking phenotypes of a new sample.

Goal: Given the genetic information of a new sample (such as an undiagnosed patient), our goal is to rank the possible phenotypes (such as a predefined pool of diseases) according to the likelihood of each individual phenotype to appear in the sample. We call this problem *on demand* phenotype ranking.

A brute forth approach to rank the phenotypes of a query sample would consist of two steps. In the first step, it mines the complete set of prediction rules and calculate their prediction significance for all phenotypes (rule generation step). In the second step, whenever a new query sample comes in, it matches the new sample with the discovered rules to rank the phenotypes according to the significance of the rules predicting them (rule matching step). Our experimental results in Section 4 show that this brute forth method is intractable in practice.

Contributions: We propose the problem of on demand phenotype ranking. We employ the concept of bi-clusters [11, 12] to model expression patterns shared by samples in the database. Cluster based prediction rules will be used to rank phenotypes for new query samples. We develop an efficient algorithm for this problem. Our algorithm focuses on patterns exhibited by the query sample and the corresponding clusters. Our algorithm also incorporates effective strategies to further prune the search space. The experimental results demonstrate that our algorithm is efficient and effective.

2 Problem Definition

Let $S = \{s_1, s_2, \cdots, s_N\}$ be a set of N samples (e.g., each sample may be a patient), $G = \{g_1, g_2, \cdots, g_M\}$ be a set of M genes and $P = \{p_1, p_2, \cdots, p_I\}$ be a set of I phenotypes. The dataset consists of two matrices M_G (containing genetic information) and M_P (containing phenotypic information). $M_G = S \times G = \{v_{nm}\}$ $(1 \leq n \leq N, 1 \leq m \leq M)$ is an $N \times M$ matrix of real numbers, where v_{nm} is the expression value of gene g_m in sample s_n. $M_P = S \times P = \{d_{ni}\}$ $(1 \leq n \leq N, 1 \leq i \leq I)$ is an $N \times I$ matrix of binary numbers, where $d_{ni} = 1$ if s_n is positive in phenotype p_i, and $d_{ni} = 0$ otherwise. Table 1 shows an example dataset.

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
q	8	0	3	1	5	4	2	16	20	13

(a) Expression values of a query sample

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
s_1	9	1	4	2	6	3	11	10	1	19
s_2	2	3	6	4	8	7	22	5	0	1
s_3	12	2	5	3	7	41	4	17	22	52
s_4	33	1	4	2	6	74	58	11	89	91
s_5	26	5	8	22	19	61	37	58	84	50
s_6	78	47	4	36	6	35	71	12	19	35
s_7	22	1	53	83	6	45	92	39	31	57
s_8	35	4	7	5	9	78	50	25	51	38
s_9	30	7	19	8	72	74	29	34	2	-5
s_{10}	56	76	36	61	79	-9	43	3	7	0
s_{11}	86	42	39	78	-7	-8	38	4	8	1
s_{12}	50	59	36	17	52	67	68	8	12	5
s_{13}	-4	49	31	22	31	57	79	4	8	1
s_{14}	76	52	33	28	23	72	-8	6	10	3

(b) Dataset of genetic information

	p_1	p_2	p_3
s_1	1	1	0
s_2	1	1	0
s_3	1	1	1
s_4	1	0	0
s_5	0	0	0
s_6	0	0	0
s_7	0	1	0
s_8	0	0	0
s_9	0	1	0
s_{10}	1	1	1
s_{11}	0	1	1
s_{12}	0	1	1
s_{13}	0	0	0
s_{14}	1	0	0

(c) Dataset of phenotypic information

Table 1: Query sample and running dataset

We use the cluster based prediction rules to rank phenotypes for a new (query) sample in descending order of their likelihood of presence. In this paper, we adopt the cluster definition from [11, 12, 13].

Definition 1. *(Cluster) A cluster C is a sub-matrix of M_G: $C = X \times Y$, where $X \subseteq S$ and $Y \subseteq G$, such that for any 2×2 sub-matrix $\begin{pmatrix} v_{ai} & v_{aj} \\ v_{bi} & v_{bj} \end{pmatrix}$ of C, we have $||(v_{ai} - v_{bi}) - (v_{aj} - v_{bj})|| \leq \epsilon$, where ϵ is the user-specified cluster threshold.* [1]

[1]To make the presentation clear, in this paper, we focus on shifting patterns [11]. Other pattern definitions, such as scaling [13] or shifting-and-scaling patterns [12], can be handled in a similar way.

In the remainder of the paper, we refer to the subset of genes Y as the *subspace* of cluster C. A cluster represents an expression pattern shared by the set of samples X in subspace Y.

For a specific phenotype $p_i \in P_I$, let S_i^+ (S_i^-) be the set of positive (negative) samples. For a cluster C, let C_i^+ (C_i^-) be the set of samples of C that are positive (negative) in p_i. We define the significance of a cluster as follows.

Definition 2. *(Significance) For a specific phenotype p_i, the probability that a positive sample is in cluster C is $prob_i^+(C) = |C_i^+| / |S_i^+|$. The probability that a negative sample is in C is $prob_i^-(C) = |C_i^-| / |S_i^-|$. The significance of cluster C in p_i is $sig_i(C) = prob_i^+(C)/prob_i^-(C)$ if $prob_i^-(C) \neq 0$; otherwise $sig_i(C) = \infty$.*

The intuition behind the significance definition is that, for a phenotype p_i, if the probability that a positive sample is in C is much higher than the probability that a negative sample is in it, then C would be a strong evidence to support that samples showing similar expression patterns as samples in C are likely to have positive p_i. The reason we use the probability instead of the absolute number of samples is that most bio-medical datasets are highly skewed. There are usually more negative samples than positive samples for each phenotype.

For a phenotype, we can define a complete order among all clusters based on their significance and the probability that a positive sample is in it.

Definition 3. *(Order of Clusters) For a specific phenotype p_i, cluster $C1 \prec C2$, if $sig_i(C1) < sig_i(C2)$, or $(sig_i(C1) = sig_i(C2)) \wedge (prob_i^+(C1) < prob_i^+(C2))$.*

For a query sample, only its genetic information (expression values) is available. For example, Table 1(a) shows the expression values of a query sample q. We are interested in those clusters that share similar expression patterns with q in their respective subspaces. These clusters are the *matching clusters* of q and are used to rank q's phenotypes.

Definition 4. *(Matching) A query sample q matches a cluster $C = X \times Y$ (or equivalently, C matches q) if $C' = (X \cup \{q\}) \times Y$ is also a cluster.*

For a particular phenotype p_i, a user may be interested in the clusters that have at least certain numbers of positive samples min_{s+} and genes min_g. A cluster is a ***valid cluster*** if it satisfies these constraints. Please note that we only specify min_{s+}, instead of setting a threshold for the minimum number of samples. The reason is that in real applications, we are interested in predicting positive phenotypes (such as positive diabetes patients). Therefore a cluster that can make strong prediction should have as few negative samples as possible. Setting the min_{s+} threshold also makes the model robust to noises. To keep the presentation clear, in this paper we assume a uniform threshold for the minimum number of positive samples required in a cluster for all phenotypes. We rank the phenotypes of a query sample in the following way.

Definition 5. *(Score of Phenotypes) Let H_i denote the complete set of valid clusters for a specific phenotype p_i. For a query sample q, the* ***score*** *of p_i, $score_q(p_i)$, is the highest significance of the clusters in H_i which match q, that is,*

$$score_q(p_i) = \max_{\mathrm{C \in H_i,\ C\ matches\ q}} sig_i(C).$$

For example, let's consider the running dataset shown in Table 1. Let $min_{s^+} = 3$, $min_g = 3$, and $\epsilon = 0$. Suppose that, for phenotype p_2, we have the complete set of valid clusters $H_2 = \{C1, C2\}$, where $C1 = (\{s_1, s_2, s_3, s_4, s_8\} \times \{g_2, g_3, g_4, g_5\})$ and $C2 = (\{s_{10}, s_{11}, s_{12}, s_{13}\} \times \{g_8, g_9, g_{10}\})$. The query sample q matches both clusters. After calculating their significance, we get $sig_2(C1) = 1.125$ and $sig_2(C2) = 2.25$. So $C2$ is the cluster that matches query sample q, and has highest significance. Therefore, for query q, the score of p_2 is $score_q(p_2) = sig_2(C2) = 2.25$.

Given the genetic information of a query sample, our goal is to rank the phenotypes in descending order of their scores for this new sample.

3 The Algorithm

Our algorithm systematically searches through clusters that matches the query sample for those ones that give highest significance score for each phenotype. Note that the set of clusters we look at is a small subset of the set of all clusters that produced by an unsupervised subspace clustering algorithms [11, 12, 13]. In addition, our algorithm incorporates effective pruning strategies by utilizing phenotype information.

3.1 Mining Process

3.1.1 Step 0: Finding Gene Pair MDSs as a Preprocessing Step In this step, the algorithm generates gene pair MDS for each pair of genes in the database[2]. This step can be done as a pre-processing step since it does not depend on the query sample. Table 2 shows all gene pair MDSs of the dataset in Table 1 when $\epsilon = 0$ and $\min_{s^+} = 3$. The gene pair MDSs will be used for finding clusters in some subspaces and calculating their significance, which will be discussed in following steps. To facilitate the discussion, we also show the positive samples in the gene pair MDSs for each phenotype in Table 2.

[2]Please refer to [11] for the details of finding gene (or sample) pair MDSs.

	p_1	p_2	p_3
$\{g_2g_3\}$:$\{s_1s_2s_3s_4s_5s_8\}$	$s_1s_2s_3s_4$	$s_1s_2s_3$	$\emptyset$
$\{g_2g_4\}$:$\{s_1s_2s_3s_4s_8s_9\}$	$s_1s_2s_3s_4$	$s_1s_2s_3s_9$	$\emptyset$
$\{g_2g_5\}$:$\{s_1s_2s_3s_4s_7s_8\}$	$s_1s_2s_3s_4$	$s_1s_2s_3s_7$	$\emptyset$
$\{g_3g_4\}$:$\{s_1s_2s_3s_4s_8\}$	$s_1s_2s_3s_4$	$s_1s_2s_3$	$\emptyset$
$\{g_3g_5\}$:$\{s_1s_2s_3s_4s_6s_8\}$	$s_1s_2s_3s_4$	$s_1s_2s_3$	$\emptyset$
$\{g_4g_5\}$:$\{s_1s_2s_3s_4s_8\}$	$s_1s_2s_3s_4$	$s_1s_2s_3$	$\emptyset$
$\{g_8g_9\}$:$\{s_6s_{10}s_{11}s_{12}s_{13}s_{14}\}$	$\emptyset$	$s_{10}s_{11}s_{12}$	$s_{10}s_{11}s_{12}$
$\{g_8g_{10}\}$:$\{s_{10}s_{11}s_{12}s_{13}s_{14}\}$	$\emptyset$	$s_{10}s_{11}s_{12}$	$s_{10}s_{11}s_{12}$
$\{g_9g_{10}\}$:$\{s_{10}s_{11}s_{12}s_{13}s_{14}\}$	$\emptyset$	$s_{10}s_{11}s_{12}$	$s_{10}s_{11}s_{12}$

Table 2: Gene pair MDSs and their positive samples

	MDSs	p_1	p_2	p_3
s_1	$\{g_1g_2g_3g_4g_5\}$	1	1	0
s_2	$\{g_2g_3g_4g_5g_6\}$	1	1	0
s_3	$\{g_2g_3g_4g_5g_7\}$	1	1	1
s_4	$\{g_2g_3g_4g_5g_8\}$	1	0	0
s_8	$\{g_2g_3g_4g_5\}$	0	0	0
s_{10}	$\{g_6g_8g_9g_{10}\}$	1	1	1
s_{11}	$\{g_5g_6g_8g_9g_{10}\}$	0	1	1
s_{12}	$\{g_8g_9g_{10}\}$	0	1	1
s_{13}	$\{g_1g_8g_9g_{10}\}$	0	0	0
s_{14}	$\{g_7g_8g_9g_{10}\}$	1	0	0

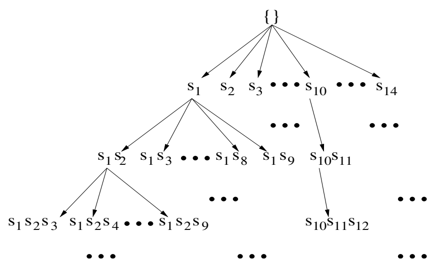
Table 3: Sample MDSs and corresponding phenotypic information

3.1.2 Step 1: Finding MDSs of Query Sample and Database Samples In this step, we pair the query sample q with each sample in the database and compute the sample pair MDSs. Using the dataset in Table 1, the sample pair MDSs of the query sample and database samples are shown in Table 3 when $\epsilon = 0$ and $min_g = 3$. Since one sample in each sample pair is always the query sample, we only show the other sample s_i in the pair in Table 3. We use the term "*sample MDS* of s_i" as the abbreviation of "the sample pair MDS of the query sample and s_i" in the remainder of this paper. We also show the phenotype information of each sample in Table 3 for convenience.

The following proposition shows the completeness and correctness of our algorithm. The proof is straightforward and thus omitted here.

Proposition 3.1. *Any cluster generated from combinations of the sample MDSs in step 1 matches the query sample. And all clusters that match the query sample can be generated by combinations of these MDSs.*

3.1.3 Step 2: Generating Significant Clusters for each Phenotype by Enumerating Combinations of Sample MDSs In this step, the algorithm starts to enumerate combinations of the sample MDSs (as shown in Table 3) to generate clusters. Figure 1(a) shows the search space of the com-



(a) Search space of combinations of sample MDSs

p_1	$\{s_2, s_3, s_4, s_{10}, s_{14}\}$
p_2	$\{s_2, s_3, s_7, s_9, s_{10}, s_{11}, s_{12}\}$
p_3	$\emptyset$

(b) $F(s_1)$

p_1	$\{s_4, s_{10}, s_{14}\}$
p_2	$\{s_7, s_9, s_{10}, s_{11}, s_{12}\}$
p_3	$\emptyset$

(c) $F(s_1 s_3)$

p_1	$\{s_{10}, s_{14}\}$
p_2	$\emptyset$
p_3	$\emptyset$

(d) $F(s_1 s_4)$

Figure 1: Enumerating the combinations of sample MDSs

binations of sample MDSs for the dataset in Table 1. We explore the search space in a depth first manner.

In our algorithm, we only need enumerate the combinations of positive sample MDSs. We will formally prove correctness of this search strategy in Theorem 3.2. Using only positive samples greatly improves the efficiency because, in bio-medical dataset, the number of positive samples (e.g., disease patients) is usually much smaller than the number of negative samples (e.g., healthy people).

Attachable Sample Sets: Our algorithm systematically explores subspaces (of genes) in which, for some phenotype, positive samples fall into a cluster. In principle, these subspaces can be identified by examining combinations of positive samples for each phenotype. In practice, many samples may have multiple positive phenotypes and hence a sample combination may need to be visited multiple times, once for each phenotype. To make this process more efficient, in our algorithm, we construct a single tree structure that processes all phenotypes simultaneously when examining a sample combination. Please note that we use the tree as guide for the search without materializing the whole tree.

In order to ensure no duplicate generation of sample combination, we assume all samples are naturally ordered by their IDs $s_1, s_2, \ldots, s_N$. As illustrated in Figure 1(a), the root node corresponds to the empty set and each descendant node corresponds to a sample set. Each node n_a also has a table $F(n_a)$ storing the set of positive samples yet to be considered for each phenotype. We refer to this sample set as the *attachable sample set* (denoted by $A_i(n_a)$) for the corresponding phenotype p_i. At the root node, the attachable sample set for phenotype p_i is the set S_i^+ of all positive samples for p_i. The remaining nodes are generated in a depth first order. A child node n_b corresponds to a sample set that has exact one more sample than its parent n_a. This additional sample (denoted by s_x) is from the attachable sample sets $F(n_a)$. The number of children of node n_a is equal to the number of distinct samples in $F(n_a)$. The attachable sample set for phenotype p_i at the child node n_b is a subset of the attachable sample set at the parent n_a by excluding samples whose IDs are smaller than or equal to x. There are two scenarios where $A_i(n_b) = \emptyset$: (1) $A_i(n_a)$ does not contain s_x; and (2) s_x is the sample of the highest ID in $A_i(n_a)$. For example, Figures 1(b), 1(c) and 1(d) show the attachable sample sets of each phenotype at nodes (s_1), $(s_1 s_3)$ and $(s_1 s_4)$ respectively using the dataset in Table 1. The attachable sample sets shrink monotonically at descendant nodes. At a leaf node, the attachable sample set is empty for every phenotype.

Cluster Generation and Significance Calculation: Suppose that the current node n_a corresponds to sample set $(s_{a_1} s_{a_2} \cdots s_{a_m})$. First, we compute the intersection of the m sample MDSs of $s_{a_1}, s_{a_2}, \cdots s_{a_m}$ to get the subspace Y. For example, using the sample MDSs shown in Table 3, for node $(s_1 s_2)$, we get the subspace $\{g_2, g_3, g_4, g_5\}$ after taking the intersection of the two sample MDSs of s_1 and s_2. There are $\binom{|Y|}{2}$ gene pairs that are subsets of Y. We take the intersection of their gene pair MDSs to get a set of samples X. $X \times Y$ is a cluster defined in Definition 1, since each pair of samples in X form a cluster in subspace Y. Continuing with the previous example, for the 6 gene pairs that are subsets of $\{g_2, g_3, g_4, g_5\}$, we intersect their MDSs in Table 2. The resulting cluster is $C = (\{s_1, s_2, s_3, s_4, s_8\} \times \{g_2, g_3, g_4, g_5\})$. Finally, we calculate the significance of the cluster in each phenotype according to Definition 2.

Now we formally prove that for any phenotype, only the positive sample MDSs are necessary for the generation of significant clusters.

Theorem 3.2. *Suppose that, for some phenotype p_i, $C1 = T \times Z = (\{s_{a_1}, s_{a_2}, \cdots, s_{a_x}, s_{b_1}, s_{b_2}, \cdots, s_{b_y}\} \times Z)$ is a cluster which includes both positive and negative samples. $C1_i^+ = \{s_{a_1}, s_{a_2}, \cdots, s_{a_x}\}$, $C1_i^- = \{s_{b_1}, s_{b_2}, \cdots, s_{b_y}\}$, and Z is the subspace of $C1$. Then there exists a cluster $C2$ in the subspace obtained by intersecting sample MDSs of only positive samples $\{s_{a_1}, s_{a_2} \cdots s_{a_x}\}$ such that $sig_i(C2) \geq sig_i(C1)$.*

Proof. Suppose by intersecting the sample MDSs of $s_{a_1}, s_{a_2}, \cdots$, and s_{a_x}, we get subspace Z'. It is easy to see that $Z \subseteq Z'$. Thus by intersecting $\binom{|Z'|}{2}$ gene pairs MDSs of Z', we will generate a cluster $C2 = T' \times Z'$, with $T' \subseteq T$. Moreover, we have $C2_i^+ \supseteq C1_i^+$ since $C2$ contains at least $\{s_{a_1}, s_{a_2}, \cdots, s_{a_x}\}$ as positive samples. So we have

$C2_i^- \subseteq C1_i^-$ since $C2_i^+ \cup C2_i^- = T' \subseteq T = C1_i^+ \cup C1_i^-$. Therefore, $sig_i(C2) \geq sig_i(C1)$. □

Updating Phenotype Ranking: In the process of searching, we maintain the current ranking of phenotypes and their corresponding clusters in a list L_p. At any time, if we find that for any phenotype p_i, a newly generated cluster has higher order (as defined in Definition 3) than the cluster in L_p, then we update L_p.

The algorithm is outlined in Algorithm 1. The procedure of enumerating the clusters is detailed in Procedure ClusEnum.

Algorithm 1: Rank phenotypes of query sample

```
Input: Query sample $q$, dataset $M_G$ and $M_P$, cluster
       thresholds $\epsilon$, $min_{s^+}$, and $min_g$
Output: A list $L_p$ of ranked phenotypes for $q$
1  $GPMDS \leftarrow$ gene pair MDSs;
2  Initialize the list of ranked phenotypes: $L_p \leftarrow \emptyset$;
3  $SMDS \leftarrow$ sample MDSs;
4  for each child node $n_a$ of the root, do
5      Compute $F(n_a)$;
6      $ClusEnum(n_a, F(n_a), SMDS, GPMDS, L_p)$;
7  end
8  Return $L_p$.
```

Procedure `ClusEnum`

```
Input: current node $n_a = (s_{a_1} s_{a_2} \cdots s_{a_m})$, its attachable
       sample sets $F(n_a)$, sample MDSs $SMDS$, Gene
       pair MDSs $GPMDS$, list of ranked phenotypes $L_p$
1  Find subspace $Y$ of $n_a$ by intersecting the sample MDSs of
   $s_{a_1}, s_{a_2}, \cdots$, and $s_{a_m}$;
2  if $|Y| < min_g$ then return;
3  For all the $\binom{|Y|}{2}$ gene pairs that are subsets of $Y$, intersect
   their gene pair MDSs to get cluster $C$;
4  Calculate $sig_i(C)$ for each phenotype $p_i$;
5  Update $L_p$;
6  Prune $F(n_a)$;
7  if $\cup A_i(n_a) \neq \emptyset$ (for $A_i(n_a) \in F(n_a)$) then
8      for each $s_x \in \cup A_i(n_a)$ do
9          Generate a child node $n_b = (s_{a_1} s_{a_2} \cdots s_{a_m} s_x)$;
10         Compute $F(n_b)$;
11         $ClusEnum(n_b, F(n_b), SMDS, GPMDS, L_p)$
12     end
13 end
14 Return $L_p$.
```

3.2 Pruning Strategies Various pruning strategies are developed to improve the efficiency of the algorithm. Pruning strategies 1 and 3 are extensions of that for building association rule based classifier for a single phenotype [6].

3.2.1 Pruning Strategy 1 As discussed in Section 3.1, at each node, we take the intersection of the gene pair MDSs to find the cluster and calculate its significance. This search strategy allows us to skip some descendant nodes of the current node.

Suppose that at node $n_a = (s_{a_1} s_{a_2} \cdots s_{a_m})$, we find a cluster $C = X' \times Y$, with $\{n_a\} \subset X'$. Then all samples in $X' - \{n_a\}$ can be removed from the attachable sample sets $F(n_a)$. This is because including samples in $X' - \{n_a\}$ would reach the same subspace Y as that of n_a. As a result, the same set of gene pair MDSs are used which generate the same cluster.

3.2.2 Pruning Strategy 2 In the process of searching, not every sample in the attachable sample sets is eligible to create a child node. The following proposition shows how to decide if it is necessary to create a child node for a sample in the attachable sample sets.

Proposition 3.3. *Suppose that, at node $n_a = (s_{a_1} s_{a_2} \cdots s_{a_m})$, Y is the subspace of n_a. There are in total $\binom{|Y|}{2}$ gene pairs that are subsets of Y. Let Z represents the set of all $\binom{|Y|}{2}$ gene pair MDSs. A sample s_x in attachable sample sets is eligible to generate a child node of n_a if s_x occurs in at least $\binom{min_g}{2}$ gene pair MDSs in Z.*

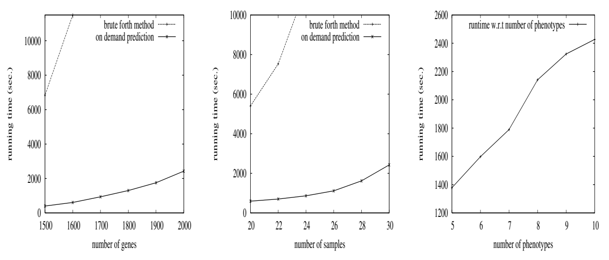
3.2.3 Pruning Strategy 3 Suppose that, at node $n_a = (s_{a_1} s_{a_2} \cdots s_{a_m})$, the attachable sample set of phenotype p_i is $A_i(n_a) = \{s_{x_1}, s_{x_2} \cdots s_{x_n}\}$ and the cluster is $C = X \times Y$. Then for phenotype p_i, any node in the subtree of n_a could generate clusters of at most $|\{n_a\}| + |A_i(n_a)| = m + n$ positive samples and at least $|C_i^-|$ negative samples. Hence, for phenotype p_i, the significance of any such cluster is at most $\frac{(|\{n_a\}|+|A_i(n_a)|)/|S_i^+|}{|C_i^-|/|S_i^-|}$. Therefore, we can prune the samples from the attachable sample sets if this significance upperbound is lower than the current significance score in L_p for phenotype p_i.

4 Experiments

To evaluate our algorithm, we perform experiments on both synthetic dataset and real-life genetic-phenotypic data. The experiments were performed on a 2.4 GHz PC with 1G memory running WindowsXP system.

4.1 Efficiency In our experimental study, we compare our algorithm to the following brute forth method. Given a dataset and thresholds min_{s^+}, min_g, and ϵ, in the pre-processing step, we compute all valid clusters in the dataset and store them. In the matching step, when a query sample comes in, we search for the best matching cluster in the set of clusters we obtain from the pre-processing step.

The synthetic datasets are used in this study. We generate the genetic data matrix M_G and phenotype data



(a) Varying #genes (b) Varying #samples (c) Varying #phenotypes

Figure 2: Efficiency evaluation

matrix M_P in the following way. For M_G, we first generate a random matrix with size $N \times M$. We then embed a certain number of clusters in M_G. By default in our subsequent experimental study, M_G contains $N = 30$ samples and $M = 2000$ genes with 30 embedded clusters. On average, a cluster contains $20\% \times N = 6$ samples and $2.5\% \times N = 50$ genes. Similarly, the synthetic phenotypic data matrix M_P is generated randomly with a default of 10 phenotypes and 30% positive samples on average for each phenotype. Query samples are randomly generated and the default values for matching parameters are $min_{s^+} = 2$, $min_g = 3$, $\epsilon = 1$.

In Figures 2(a) to 2(c), we show the running time of our algorithm and the brute forth method as a function of various parameters. In contrast to our slow-growing running time as shown in Figures 2(a) and Figure 2(b), the brute forth method is intractable in practice due to the huge number of clusters in the dataset. Figure 2(c) shows a sub-linear relationship between the runtime of our algorithm and the number of phenotypes. If we explore the search space for each phenotype separately, the running time would be linear to the number of phenotypes. The sub-linear performance demonstrates the advantage of simultaneously enumerating the clusters for all phenotypes.

4.2 Effectiveness We apply our algorithm on real-life genetic-phenotypic data. The dataset is collected in the School of Public Health at UNC-Chapel Hill. Among all patients, there are in total 19 patients who have been diagnosed with either asthma or cardiovascular diseases or both. Microarray experiments are performed on all patients to measure the expression values of 5000 genes. For our method, we perform leave-one-out analysis: each time we take one sample as the query sample and the remaining samples as database samples. We set $min_g = 3$, $min_{s^+} = 3$, and $\epsilon = 0.2$. Our algorithm correctly ranks 95% of the query patients, that is, the positive disease is ranked higher than other diseases for the query patients.

For comparison, we also run the k-nearest-neighbor (KNN) method on the same dataset. For each patient, the k patients with the closest expression patterns for the 5000 genes are selected. The diseases are then ordered by the number of patients having them in the k neighbors. This KNN method performs the best when $k = 5$ in our experiment, where it correctly ranks the phenotypes for 73% of the query patients.

5 Conclusions

In this paper, we investigate the problem of on demand phenotype ranking. We utilize pattern based subspace clusters to construct prediction rules for phenotype ranking. Given a sample, our algorithm only examines samples that share similar patterns with the query sample. It utilizes sample MDSs to identify the subspaces where these patterns resides and uses gene-pair MDSs to generate clusters representing these patterns. Only the clusters with high prediction power of any phenotypes are used to rank the phenotypes. The experimental results demonstrate the efficiency and effectiveness of our algorithm and show infeasibility of the brute forth method.

References

[1] M. Allen and et al. Positional cloning of novel gene influencing asthma from chromosome 2q14. *Nature Genet.*, 35:258–263, 2003.

[2] D. Bandyopadhyay and et al. Structure-based function inference using protein family-specific fingerprints. *Protein Science*, 15:1537–1543, 2006.

[3] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. *Wadsworth*, 1984.

[4] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121 – 167, 1998.

[5] P. Carmona-Saez and et al. Integrated analysis of gene expression by associatoin rule discoery. *BMC Bioinformatics*, 7:54, 2006.

[6] G. Cong, K. Tan, K. Tung, and X. Xu. Mining top-k covering rule groups for gene expression data. *SIGMOD*, 2005.

[7] G. O. Consortium. The gene ontology (go) database and informatics resource. *Nucl. Acids. Res.*, 32(90001):D258–261, 2004.

[8] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. *KDD*, 1998.

[9] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[10] M. J. van de Vijver and et al. Gene-expression signature as a predictor of survival in breast cancer. *N. Engl. J. Med.*, 347:1999–2009, 2002.

[11] H. Wang, W. Wang, J. Yang, and Y. Yu. Clustering by pattern similarity in large data sets. *SIGMOD*, 2002.

[12] X. Xu, Y. Lu, A. Tung, and W. Wang. Mining shifting-and-scaling co-regulation patterns on gene expression profiles. *ICDE*, 2006.

[13] L. Zhang and M. Zaki. An effecitve algorithm for mining coherent clusters in 3d microarray data. *SIGMOD*, 2005.