Chapter III

Exploring Similarities across High-Dimensional Datasets

Karlton Sequeira, Rensselaer Polytechnic Institute, USA

Mohammed Zaki, Rensselaer Polytechnic Institute, USA

Abstract

Very often, related data may be collected by a number of sources, which may be unable to share their entire datasets for reasons like confidentiality agreements, dataset size, and so forth. However, these sources may be willing to share a condensed model of their datasets. If some substructures of the condensed models of such datasets, from different sources, are found to be unusually similar, policies successfully applied to one may be successfully applied to the others. In this chapter, we propose a framework for constructing condensed models of datasets and algorithms to find similar substructure in pairs of such models. The algorithms are based on the tensor product. We test our framework on pairs of synthetic datasets and compare our algorithms with an existing one. Finally, we apply it to basketball player statistics for two National Basketball Association (NBA) seasons, and to breast cancer datasets. The results are statistically more interesting than results obtained from independent analysis of the datasets.

Introduction

Often, data may be collected by a number of sources. These sources may be geographically far apart. There are a number of disadvantages in transferring the datasets from their source to a central location for processing. These include less reliability, security, higher computational and storage requirements, and so forth. It may be preferable to share condensed models of the datasets. Similarly, for reasons like confidentiality agreements, it may be required to use condensed models of datasets, which obfuscate individual details while conveying structural information about the datasets. Lastly, the datasets may have slightly different dimensionality or transformations like rotations, with respect to each other. This may preclude simply appending the datasets to each other and processing them.

If *unusually similar* substructure can be detected from the condensed models of some of the datasets, then policies successfully applied to one may be successfully applied to the others. For example, two consumer markets (*A* and *B*) differing in geography, economy, political orientation, or some other way may have some unusually similar consumer profiles. This may prompt sales managers in B to use successful sales strategies employed by sales managers in A for consumer profiles in which they are unusually similar. Also, profiles which are *unusually dissimilar* to any of those in the other graph are particularly interesting. The latter is analogous to the problem of finding contrast sets (Bay & Pazzani, 2001). Additionally, determining similarities and dissimilarities between snapshots of a dataset taken over multiple time intervals can help in identifying how the dataset characteristics evolve over time (Ganti, Gehrke, Ramakrishnan, & Loh , 1999).

A dataset may be a set of points drawn in possibly different proportions, from a mixture of unknown, multivariate, and perhaps non-parametric distributions. A significant number of the points may be noisy. There may be missing values as well. We currently assume that the dataset may belong to non-identical attribute spaces, which are mixtures of nominal and continuous variables. The datasets may be subject to translational, rotational, and scaling transformations as well. High-dimensional datasets are inherently sparse. It has been shown that under certain reasonable assumptions on the data distribution, the ratio of the distances of the nearest and farthest neighbors to a given target is almost 1 for a variety of distance functions and data distributions (Beyer, Goldstein, Ramakrishnan, & Shaft, 1999). Hence, traditional distance metrics which treat every dimension with equal importance have little meaning. Algorithms using such dissimilarity measures as a building block for application to high-dimensional datasets may produce meaningless results due to this lack of contrast.

In this chapter, we explore similarities across datasets using a two-step solution:

1. *Constructing a* ***condensed model*** *of the dataset.* This involves finding the components of the model, and relationships between these components. In our case, the components are subspaces. The condensed model is a weighted graph where the vertices correspond to subspaces and the weighted edges to relationships between the subspaces. A condensed model allows: (a) sharing of dataset summaries, (b) noise and outlier removal, and (c) normalization and dataset scaling.
2. *Identifying similarities between the condensed models.* In our solution, this reduces to finding structurally similar **subgraphs** in the two models and matching vertices between the structurally similar subgraphs.

In previous work (Sequeira & Zaki, 2004), we have shown algorithms to find components of the model. In this chapter, we make the following contributions:

1. We propose two kinds of **similarity measures** for subspaces (components). The first kind is projection based—that is, it uses the similarity of the projections of the subspaces. The other is support based—it uses the number of points shared by the subspaces.
2. We provide algorithms for identifying *unusually similar* substructure from the condensed models corresponding to pairs of datasets with possibly differing dimensionality.
3. We test our framework with synthetic datasets and apply it to finding similar substructure in models constructed from basketball player statistics and breast cancer datasets. Inferences from the similar substructure are found to be logically meaningful. Further, they reveal information, which remains unknown under independent dataset analysis.

Preliminaries

Consider dataset D_A having d_A dimensions. If $S_{A,i}$ is the domain of the i^{th} dimension, then $S_A=S_{A,1} \times S_{A,2} \times \ldots \times S_{A,d_A}$ is the high-dimensional space for D_A, where $D_A=\{x_i | i\in\{l,m,\} x_i \in S_A\}$. Similarly, $D_B=\{y_i | i\in\{l,n,\} y_i \in S_B\}$. If the range $S_{A,i}$ of each dimension is divided into ξ equi-width intervals, then S_A has a grid superimposed over it. Accordingly, we have the following definition: a **subspace** is a grid-aligned hyper-rectangle $[l_1, h_1]\times[l_2, h_2]\times\ldots\times[l_d, h_d]$, $\forall i\in[l, d,]$ $[l_i, h_i]\subseteq S_{A,i}$. Here for a given interval $[l_i, h_i]$, we have $l_i = (aS_{A,i})/\xi$ and hi $=(bs_{A,i})/\xi$, where a,b are non-negative integers, and $a<b\leq\xi$.

If $[l_i, h_i]\subset S_{A,i}$, the **subspace** is said to be *constrained* in dimension *i*—that is, the subspace does not span the entire domain of the dimension *i*, A subspace that is constrained in all the dimensions to a single interval—that is, $b-a=1$ is referred to as a *grid cell.*

If our algorithm to find components in a dataset finds $|V_A|$ components/subspaces internal to *DA*, the relationships between these subspaces are expressed by a $|V_A|\times|V_A|$ matrix $w_A : S_A \times S_A \rightarrow \Re$.

We also use the following notations for the rest of this chapter: let $A=(a_{i,j})_{1\leq i,j\leq m,n}$ and $B=(b_{kl})_{1\leq k,l\leq p,q}$ be two matrices. If $m=n$, $Tr[A]=\Sigma_{i\in[1,m]}a_{i,i}$ is the trace of A. A^T refers to the transpose of A.

$$\| A \|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} | a_{i,j} |^2\right)^{1/2},$$

is the Frobenius norm of A. *ones*(m,n) returns a m×n matrix containing all ones. The tensor product of A and B is a $mp\times nq$ matrix, and is defined as:

$$A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{2,n}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,n}B \\ a_{m,1}B & a_{m,2}B & \dots & a_{m,n}B \end{pmatrix}$$

An $n\times n$ matrix X is called *normal*, if it can be written as $X=U_X D_X U^T_X$. U_X is a unitary matrix containing the eigenvectors of X, and D_X is a diagonal matrix containing the eigenvalues of X.$\lambda_{X,i}$ denotes the i^{th} eigenvalue, where $\forall i\in[1,n-1]$, $\lambda_{X_i}\geq\lambda_{X,i+1}$, and $U_{X,i}$ denotes the eigenvector corresponding toλ_{X_i}. If $\lambda_{X1}>\lambda_{X2}$, $\lambda_{X,1}$ and $U_{X,1}$are called the dominant eigenvalue and dominant eigenvector respectively.

If $S=[s_1,s_2...]$ where $s_1,s_2...$ are column vectors, then $vec(S)$ creates a column vector by stacking its column vectors one below the other, so that $vec(S) = [s_1^T\ s_2^T \ldots]^T$.

Let V_A and V_B be the components (subspaces) of datasets D_A and D_B, respectively. Let P be the function, which takes as argument a mapping f:$V_A\rightarrow V_B$, and returns a permutation matrix (typically, a permutation matrix is a square matrix)—that is, a $|V_A|\times|V_B|$ matrix, such that

$$P_f(u,v) = \begin{cases} 1\, if & f(u) = v \\ 0 & otherwise \end{cases} \tag{1}$$

If *f* is a one-to-one mapping, then if $|V_A| \leq |V_B|$ ($|V_A| > |V_B|$), the rows (columns) of *P* are orthogonal to each other and $PP^T = I$ ($P^T P = I$). As in Van Wyk and Van Wyk (2003), we want *f* which minimizes the associated error function *err*, which we define as:

$$err(f \mid w_A, w_B) = \| w_A - P_f w_B P_f^T \|_F \qquad (2)$$

A mapping *f* from a subset of subspaces corresponding to w_A to a subset corresponding to w_B is *unusually similar,* if the probability of finding another mapping f' between these subsets, by MonteCarlo sampling as later in this chapter, such that $err(f \mid w_A, w_B) > err(f' \mid w_A, w_B)$is very low.

Example

Let D_A and D_B be two datasets as shown in Table 1, with domains [0,1000) for each dimension. $D_A(p_1,d_1)$ refers to row p_1, column d_1 of dataset D_A. If we discretize the domain of each dimension into 10 intervals (i.e., ξ=10), then the grid cells surrounding the points in D_A, D_B yield the datasets D'_A, D'_B in Table 2. For example, $D_A(p_1,d_1)$=915. Therefore,

$$D'_A(g_1, d_1) = \lfloor \frac{915}{1000} \times \xi \rfloor = 9.$$

Thus, p_1 is constrained to the last interval in dimension d_1—that is, [900,1000). We then run a subspace mining algorithm (e.g., SCHISM (Bay & Pazzani, 2001), CLIQUE (Ganti et al., 1999)) on each of the discretized datasets independently and find two sets of subspaces *S* and *S*' corresponding to D_A and D_B respectively, as shown in Table 3. Here -1 implies that the dimension is unconstrained. $S(c_1,d_2)$=5 means that the subspace c_1 in the set S of subspaces is constrained to interval 5 in dimension d_2—that is, [500,600). Subspaces may be constrained to more than one interval in a dimension.

Typically, **subspace** mining algorithms also partition the dataset based on the subspaces it finds. Let us assume that the subspace mining algorithm assigns p_1, p_2 to c_1, p_3, p_4, p_5 to c_2, p_6, p_7, p_8 to c_3 and labels p_9 as noise. Similarly, it assigns p'_1, p'_2, p'_3, p'_4 to c'_1; p'_5, p'_6, p'_7 to c'_2 vand labels p'_8 as noise.

Given such subspaces and the points assigned to them, we wish to construct **condensed models** of the datasets, which can be used to discover structurally similar subspaces across the two datasets without having access to the datasets or their schema. For example, in Table 3, if d_2 corresponds to d'_4 and d_4 corresponds to d'_2, then c_1 and c'_1 are both constrained in the same dimension and to the same

Table 1. Original data

D_A	d_1	d_2	d_3	d_4	d_5
p_1	915	561	866	657	661
p_2	965	575	534	860	365
p_3	217	506	121	452	303
p_4	758	512	357	423	289
p_5	276	531	327	418	335
p_6	268	520	351	348	454
p_7	239	514	369	301	451
p_8	237	510	377	650	472
p_9	33	118	144	388	280

D_B	d'_1	d'_2	d'_3	d'_4	d'_5
p'_1	889	710	591	564	679
p'_2	854	189	641	564	666
p'_3	553	869	449	612	199
p'_4	779	690	203	598	872
p'_5	88	453	965	541	324
p'_6	391	436	193	578	301
p'_7	574	450	220	588	270
p'_8	805	60	803	525	152

Table 2. Discretized data

D'_A	d_1	d_2	d_3	d_4	d_5
g_1	9	5	8	6	6
g_2	9	5	5	8	3
g_3	2	5	1	4	3
g_4	7	5	3	4	2
g_5	2	5	3	4	3
g_6	2	5	3	3	4
g_7	2	5	3	3	4
g_8	2	5	3	6	4
g_9	3	1	1	3	2

D'_B	d'_1	d'_2	d'_3	d'_4	d'_5
g'_1	8	7	5	5	6
g'_2	8	1	6	5	6
g'_3	5	8	4	6	1
g'_4	7	6	2	5	8
g'_5	8	4	9	5	3
g'_6	3	4	1	5	3
g'_7	5	4	2	5	2
g'_8	8	6	8	5	1

Table 3. Two sets of subspaces

S	d_1	d_2	d_3	d_4	d_5
c_1	-1	5	-1	-1	-1
c_2	-1	5	-1	4	3
c_3	2	5	3	-1	4

S'	d'_1	d'_2	d'_3	d'_4	d'_5
c'_1	-1	-1	-1	5	-1
c'_2	-1	4	-1	5	2

interval—that is, [500,600). Also, c_2 and c'_2 are constrained in the same dimensions to similar intervals. Hence, $c_1 \approx c'_1$ and $c_2 \approx c'_2$. Thus, we wish to recover the mapping between c_1 and c'_1, and c_2 and c'_2.

Related Work

Our two-step solution to finding *unusually similar* substructure across datasets involves:

1. Constructing a condensed model of the dataset; this involves two sub-steps:
   a. finding components in the dataset
   b. constructing a condensed model from the components
2. Identifying similarities between the condensed model

Finding Components in the Dataset

We find components in the dataset using a subspace mining algorithm called SCHISM (Bay & Pazzani, 2001), which finds sets of possibly overlapping subspaces, for example, set S from dataset D_A in the example above. It partitions the points in the datasets using these subspaces. *Note that any other hyper-rectangular subspace mining algorithm—for example, MAFIA (Beyer et al., 1999), CLIQUE (Sequeira & Zaki, 2004), and so forth—may be used to find the subspaces and partition the dataset.* Hence, we do not delve into the details of the SCHISM algorithm.

Constructing Condensed Models from the Components

We condense the dataset using a weighted graph, where the vertices correspond to subspaces and the weights on the edges to similarities between the subspaces. While we are unaware of much related work on similarities between subspaces, it is noteworthy that subspaces are also clusters. Accordingly, we review some of the existing similarity measures used for comparing clusterings.

Clusterings may be compared based on the number of point pairs, in which the two clusterings C,C' agree or disagree. Each pair of dataset points is assigned to one of four categories N_{00}, N_{01}, N_{10} , and N_{11}. Pairs of points in N_{00} are assigned to distinct clusters in both C and C', those in N_{01} are assigned to the same cluster in both C and C', those in N_{01} are assigned to the same cluster in C but to distinct clusters in

C', and so on. If the dataset has n points, $N_{00}+N_{01}+N_{10}+N_{11}=n(n-1)/2$. Accordingly there exists the **Rand index**,

$$Rand(C,C') = \frac{N_{11}+N_{00}}{N_{11}+N_{10}+N_{01}+N_{00}}$$

and the **Jaccard index**,

$$Jaccard(C,C') = \frac{N_{11}}{N_{11}+N_{10}+N_{01}}$$

to compare the clusterings. Further Meila (2003) proposes the VI (variance of information) metric to compare clusterings: $VI(C,C') = H(C)+H(C')-2I(C,C')$, where $H(C)=\sum_{i=1}^{|C|} -p_i log(p_i)$, $p_i = \frac{n_i}{n}$ and where $I(C,C')=\sum_{i=1}^{|C|}\sum_{j=1}^{|C'|} p_{i,j} log(\frac{p_{i,j}}{p_i p_j})$,

$$p_{i,j} = \frac{|C_i \cap C'_j|}{n},$$

with n_i being the number of points in C_i, the i^{th} cluster in C. This implies that p_i and $p_{i,j}$ are simply the *support* of clusters C_i and $C_i \cap C_j$ respectively, according to the traditional definition of support in the data mining literature (Bay & Pazzani, 2001).

Thus, these clustering (dis)similarity measures use (dis)similarity in support overlap to express cluster similarity.

Identifying Similarities Between Condensed Models of Different Datasets

Ganti et al. (1999) compare datasets by comparing their respective models. The datasets share a common schema. A dataset may be typically modeled by a decision tree, a set of clusters, or a set of frequent itemsets. The model consists of a set of pairs. Each pair consists of an "interesting region" in the dataset (called the *structural component*) and the fraction of the dataset (called the *measure component*) it accounts for. They then partition the attribute space using hyperplanes, which (as per the type of model chosen) define the leaves, clusters or frequent itemsets, induced by the models of the two datasets. Using a single scan of each dataset, they can compute the fraction of each dataset in each distinct hyperspace, resulting from the superposition of the two models of the datasets. They then compare these fractions, corresponding to different datasets but the same hyperspace, using a "difference" function and combine the resulting "deviation" using an "aggregation" function which returns a measure of the similarity of the datasets. This method does not leverage the structure present in the data and hence is susceptible to translational transformations.

Much of the existing work in the database community (Bay & Pazzani, 2001) assumes the datasets have identical schema and that access to both datasets simultaneously is possible. By utilizing the underlying structure in the datasets, we avoid making such assumptions.

Li, Ogihara, and Zhu (2002) use a variant of the mutual information between datasets D_A and D_B, modeled by sets of maximal frequent itemsets (MFIs) F_A and F_B, which is defined as:

$$I(F_A, F_B) = \sum_{i \in F_A, j \in F_B} \frac{|i \cap j|}{|i \cup j|} log(1 + \frac{|i \cap j|}{|i \cup j|}) * min(|i|, |j|).$$

They assume an identical schema for two datasets and define the similarity between the datasets as:

$$\frac{I(F_A, F_B) * 2}{I(F_A, F_A) + I(F_B, F_B)}.$$

To test for significance of similarity, they propose bootstrapping-based approaches in which disjoint pairs of subsets of the attributes are drawn at random from samples of the given datasets. The similarity between the pairs of samples is used to estimate the distribution of similarity between the two datasets. They then generalize their approach to heterogeneous datasets, of which matchings between some of the attributes of the two datasets are known. These matchings are used to identify matchings of at least ξ attributes of one dataset with those of the other.

There have been a number of graph matching algorithms, stemming from work in the field of computer vision, regarding applications like image registration, object recognition, and so forth. Many of the past approaches involve matching between labeled or discrete-attributed graphs (Bunke, 1999; Carcassoni, 2002; Kalviainen & Oja, 1990; Van Wyk & Van Wyk , 2003). Like the solutions to many other NP-hard problems, graph matching algorithms may be *enumerative* (Bunke, 1999; Shapiro & Haralick, 1985) or *optimization based* (Carcassoni, 2002; Van Wyk & Van Wyk, 2003). Most of these algorithms assume the graphs lie in the same space, which is usually low dimensional (i.e., two or three dimensions).

The concept "two vertices are similar, if vertices they are related to are similar" allows recursive definition of inter-vertex similarity. This idea is used explicitly or implicitly by a number of propagation-based algorithms (Melnik, Garcia-Molina, &

Rahm, 2002) for a range of applications. The recursive definition causes similarity to flow from one vertex to the other.

Blondel, Gajardo, Heymans, Senellart, and Van Dooren (2004) show that given w_A and w_B (the similarity among the subspaces with the two datasets), $|V_A|\times|V_B|$ similarity matrix S, whose real entry $s_{i,j}$ represents the similarity between vertex i of G_A and j of G_B, can be obtained as the limit of the normalized even iterates of $S_{k+1} = w_B S_k w_A^T + w_B^T S_k w_A$. Note that this model does not assume that w_A and w_B are symmetric. This algorithm has time complexity of matrix multiplication, which is currently $O(=n^{2.376})$. We compare our algorithms with Blondel's algorithm.

Gionis, Mannila, and Tsaparas (2005) examine the problem of finding a clustering that agrees as much as possible with a set of given clusterings on a given dataset of objects. They provide an array of algorithms seeking to find either: (i) the clustering that minimizes the aggregate number of disagreements with the given set of clusterings (clustering aggregations), or (ii) a partition of the objects into two groups, such that the sum of aggregate dissimilarities between objects in the same group and aggregate similarities between objects in different groups is minimized (correlation clustering). Here the (dis)similarities between objects are defined using the given clusterings. This differs from our work, in that the same dataset is used to produce each clustering.

Constructing A Condensed Model Of The Dataset

We represent each dataset D_A by a weighted graph $G_A(V_A, E_A, w_A)$, where V_A is the set of subspaces found by the subspace mining algorithm, $E_A \subseteq V_A \times V_A$ is the set of edges between the subspaces in the dataset, and $w_A : S_A \times S_A \rightarrow \Re$ is the adjacency matrix/set of weights on the edges of the graph G_A, indicating similarity between components/subspaces in the condensed model/graph of G_A. Depending on whether we use support or similarity of projections as the basis for comparing subspaces, we prescribe the following subspace similarity measures.

Support-Based Subspace Similarity

Each subspace $u \in V_A$ partitions the space S_A into a clustering containing two clusters—that is, u and $S_A \backslash u$. Accordingly, if $C_u, C_{u'}$ are the clusterings yielded by subspaces $u, u' \in V_A$, we can define w_Av(u,u') using $Jaccard(C_u, C_{u'})$ and $Rand(C_u, C_{u'})$. Additionally, we experiment with using the VI measure of Meila (2003): $w_A(u,u') = exp(-VI(C_u, C_{u'}))$.

Projection-Based Subspace Similarity

Consider the case where the datasets being modeled are sets of points sampled in different proportions with respect to each other from the same mixture of multivariate distributions. Then, correctly matching these distributions using support-based subspace similarity measures is unlikely. Accordingly, we seek similarity measures which use similarity of the projections of the subspaces.

We define the similarity between subspace $R \in V_A$ and a grid cell Q surrounding a point $r \in D_A$ using the Jaccard-coefficient as:

$$\rho(r \in Q, R \in V_A) = \frac{1}{d_A} \sum_{i=1}^{d_A} \frac{|Q_i \cap R_i|}{|Q_i \cup R_i|} \qquad (3)$$

Here, Q_i, R_i refer to the set of intervals spanned by subspaces Q, R respectively, in dimension i. If dimension i of R is unconstrained, then $|R_i|=\xi$.

For example, using our running example,

$$\rho(p_1 \in g_1, c_1) = \frac{1}{d_A}\left(\frac{1}{\xi} + \frac{1}{1} + \frac{1}{\xi} + \frac{1}{\xi} + \frac{1}{\xi}\right) = 0.28.$$

Based on the subspaces found by the subspace mining algorithm, it is possible, for example using nearest neighbors, to assign points in the dataset to subspaces. Using the assignment of points to subspaces, we have devised two similarity measures: AVGSIM and HIST.

AVGSIM

Each subspace may be thought to be more accurately approximated by the points assigned to it. As we know the similarity between the grid cell around each point and every subspace found by the subspace mining algorithm using ρ() from Equation 3, the similarity between two subspaces $u \in V_A, u' \in V_A$ can be defined as:

$$w_A(u,u') = \frac{\sum_{r \in u} \rho(r,u')}{|u|} + \frac{\sum_{r \in u'} \rho(r,u)}{|u'|} \qquad (4)$$

From our running example,

$$\rho(p_1 \in g_1, c_2) = 0.24, \rho(p_2 \in g_2, c_2) = 0.44, \rho(p_3 \in g_3, c_1) \\ = \rho(p_4 \in g_4, c_1) = \rho(p_5 \in g_5, c_1) = 0.28 \quad .$$

Then, $w_A(c_1, c_2) = \frac{0.24 + 0.44}{2} + \frac{0.28 + 0.28 + 0.28}{3} = 0.62.$

To ensure that $\forall u \in V_A, w_A(u,u) = 1$, we normalize by setting

$$w_A(u,u') = \frac{w_A(u,u')}{\sqrt{w_A(u,u) \times w_A(u',u')}} .$$

HIST

Based on the coordinates of points assigned to each subspace in V_A, we estimate discrete p.d.f.s. for each dimension for each subspace. If each dimension of the d_A-dimensional dataset is discretized into ξ equi-width intervals, then $u(i,j)$ corresponds to the fraction of points assigned to vertex/subspace u, which are discretized to the j^{th} interval in the i^{th} dimension. Using our running example, there are two points p_1,p_2 assigned to subspace c_1. Both of them are discretized to the interval 5 in the dimension d_2—that is, [500,600). Therefore, $c_1(2,5) = \frac{2}{2} = 1$. Accordingly,

$$w_A(u,u') = \frac{1}{d_A \xi} \sum_{i=1}^{d_A} \sum_{j=1}^{\xi} sim(u(i,j), u'(i,j)) \tag{5}$$

where $sim : [0,1] \times [0,1] \rightarrow [0,1]$ is a similarity function.

Note that this $w_A()$ requires no normalization if we use the Gaussian or increasing weighted $sim()$ function shown below. Otherwise, normalization is required. We have tested a number of symmetric similarity functions:

- **Dot product:** $sim(a,b) = a \times b$
- **Gaussian weighted:** $sim(a,b) = exp(\frac{-(a-b)^2}{2s^2})$
- **Increasing weighted:** $sim(a,b) = \frac{1}{1 + \frac{|a-b|}{s}}$

where s is a user-defined parameter controlling the spread of sim. Note that our similarity measures are both symmetric and independent of the number of points assigned to each subspace.

Identifying Similarities Between Condensed Models

Once we have the internal similarity among the subspaces within each dataset in the form of weighted graphs, to find similarities between the dataset graphs, we test three algorithms. One (**OLGA**) uses the tensor product, the next (**EigenMatch**) uses ideas from the first and Blondel's algorithm, and the last uses MonteCarlo sampling.

OLGA

We combine the graphs G_A and G_B into a single bipartite graph $G = (V_A \cup V_B, E \subseteq V_A \times V_B, \Pi)$. Π is a $|V_A| \times |V_B|$ matrix of pairwise vertex similarities.

To find Π, we construct the product graph (see function **ProductGraph** in Figure 1 $G' = (V_A \times V_B, E' \subseteq (V_A \times V_B) \times (V_A \times V_B), w_{A,B})$, where $w_{A,B} : E' \to \Re$ is the adjacency matrix, indicating similarity between vertices corresponding to pairs of subspaces from underlying graphs of G'. Let $\beta(A,B) = sim(w_A(u,u'), w_B(v,v'))$, then

$$w_{A,B}((u,v),(u',v')) = \begin{cases} \beta(A,B) & if \quad \beta(A,B) > \tau \\ 0 & otherwise \end{cases} \tag{6}$$

where τ is a user-specified threshold, used to minimize noise and limit space complexity of the algorithm. As $w_A(u,u'), w_B(v,v')$ depend on G_A, G_B respectively, the weight of an edge in product graph G' is high, if the weights on the corresponding edges in the underlying graphs are similar. Thus, we do not explicitly compare dimensions of vertices in the two graphs, thereby making no assumptions on identical schema. Let $S = vec(\Pi)$ (as defined above) and $l = |V_A||V_B|$ length column vector. Using the concept, "two vertices are similar, if vertices they are related to, are similar," then similarity between $u \in V_A$ and $v \in V_B$ is a function of all the vertices in V_A and V_B, and the relationships that u and v have with them, respectively. If S_i denotes S at iteration i, we can write this as (with $u' \in V_A, v' \in V_B$):

$$\begin{aligned} S_i((u,v)) &= \sum w_{A,B}((u,v),(u',v'))S_{i-1}((u',v')) \\ &= w_{A,B}((u,v),:) \cdot S_{i-1} \\ Then, S_i &= w_{A,B} \cdot S_{i-1} \end{aligned}$$

where $w_{A,B}((u,v),:)$ returns the $(u,v)^{th}$ row of $w_{A,B}$. As shown in Figure 1, we set the initial similarities—that is, all entries in S_0—to 1.0 (line 6). We then iterate using

Equation 7 (line 8). We determine convergence by checking to see if the Frobenius norm of the residual at the end of each iteration is less than a user-specified threshold ε (line 9).

As we are looking for a matching between vertices from G_A to G_B, we may unstack the vector S and use the resulting $|V_A|\times|V_B|$ matrix as the adjacency matrix of the bipartite graph G (i.e.,$\prod$).

Ideally, $\prod$ is a permutation matrix which minimizes $err(f|w_A,w_B)$ (Equation 2). Typically however, $\prod$ is a real matrix. Hence, we need to *round* $\prod$ to a permutation matrix. We use the **Match** function to do the same. **Match** returns f: $V_A \rightarrow V_B$. There are a number of matching algorithms, for example, stable matching, the Kuhn-Munkres algorithm (Kuhn, 1955), perfectionist egalitarian polygamy (Melnik et al., 2002), and so forth. We can formulate the *rounding* as finding a matching which maximizes the sum of the weights on the edges of the matching. Finding such a **matching** (also called an *alignment*) is called *bipartite weighted matching,* which has earlier been optimally solved by the Hungarian algorithm (Kuhn, 1955). This algorithm has complexity $O(\max\{|V_A|,|V_B|\})^3)$. This is equivalent to partitioning G into a number of clusters such that no cluster contains two vertices from the same graph, and the total of the similarity among the vertices within each cluster is maximized. **Match,** unless otherwise mentioned, refers to the Hungarian algorithm. There are other approximate matching algorithms of lower complexity. We do not take into

Figure 1. Matching two graphs

```
ProductGraph($G, G_A, G_B$):
1. $\forall(u,v) \in (V_A \times V_B)$ create vertex $(u,v)$
2. $\forall(u,u') \in (V_A \times V_A)$
3.     $\forall(v,v') \in (V_B \times V_B)$
4.         add edge $((u,v),(u',v'))$ using Eq. 6

OLGA($G_A, G_B, \tau, k$):
5. ProductGraph($G, G_A, G_B$)
6. $S_0 = ones(|V_A|,|V_B|)$
7. for i=1:k
8.     $S_i = \frac{w_{A,B} \cdot S_{i-1}}{\| w_{A,B} \cdot S_{i-1} \|_F}$
9.     if $\| S_i - S_{i-1} \|_F < \varepsilon$ break
10. return Match($S_k$)

FastOLGA($G_A, G_B$):
11. Find $U_{A,1}, \lambda_{A,1}, \lambda_{A,2}$
12. Find $U_{B,1}, \lambda_{B,1}, \lambda_{B,2}$
13. if $\lambda_{A,1} \neq \lambda_{A,2}$ and $\lambda_{B,1} \neq \lambda_{B,2}$
14.     $S = U_{A,1} \otimes U_{B,1}$
15.     return Match($S$)
```

account the complexity of **Match** while stating complexity of the algorithms, as it is a parameter. This idea is similar to similarity propagation in Melnik et al. (2002). However, they use directed, labeled graphs.

If $w_{A,B}$ is normal, it is diagonalizable. If it has a dominant eigenvalue,

$$S_i = \frac{w_{A,B} \cdot S_{i-1}}{\| w_{A,B} \cdot S_{i-1} \|_F} \; Then,\, S' = \lim_{i \to \infty} S_i = \frac{w_{A,B} \cdot S'}{\| w_{A,B} \cdot S' \|_F} \tag{7}$$

Rearranging, $(w_{A,B} - \| w_{A,B} \cdot S' \|_F \cdot I)S' = 0$, where I is the $l \times l$ identity matrix. Note, this is the characteristic equation for $w_{A,B}$. Then, $w_{A,B}$ has a dominant eigenvalue $\lambda_1 = \| w_{A,B} \cdot S' \|_F$ and dominating eigenvector S'. The rate of convergence is determined by the ratio

$\frac{\lambda_2}{\lambda_1}$ (Golub & Van Loan, 1996).

If *sim* returns the scalar product of its inputs and τ=0, then $w_{A,B}((u,v),(u',v')) = w(u,u')w(v,v')$ and $w_{A,B} = w_A \otimes w_B$, as defined above. If $w_{A,B}$ corresponds to the tensor product, further improvements in the time and space complexity of the algorithm are possible. Accordingly, we have **FastOLGA** algorithm in Figure 1.

It is known (West, 1996) that the set of eigenvalues of the tensor product of two matrices is the set of values in the tensor product of the eigenvalues of these matrices, that is,

$$w_{A,B} = w_A \otimes w_B \Rightarrow 1 \le i, j \le |V_A|, |V_B|, \lambda_{w_A,i}\lambda_{w_B,j}$$

is an eigenvalue of $w_{A,B}$. Hence, the dominant eigenvalue of the tensor product of $w_{A,B}$ (if it exists) is the product of the dominant eigenvalues of the w_A and w_B. This implies that convergence is achieved if both w_A and w_B have dominant eigenvalues (line 13). Similarly, the set of eigenvectors of the tensor product of two matrices is the set of values in the tensor product of the eigenvectors of these matrices. This implies that $S' = U_{A,1} \otimes U_{B,1}$, using notation from above for dominant eigenvectors. Finding a **maximal matching** in the tensor product of the dominant eigenvectors corresponds to projecting the longer eigenvector onto the space of the smaller eigenvector and permuting the dimensions of the former, such that their cosine similarity is maximized (i.e., aligning them).

The dominant eigenvector of an $n \times n$ matrix can be determined in $O(n^2)$time (lines 11,12) using QR factorization (Golub & Van Loan, 1996), and the tensor product of $|V_A|$ and $|V_B|$ length vectors is computed in $|V_A||V_B|$ steps (line 14). This allows computation of S′ in $O(max(|V_A|^2, |V_B|^2))$ time (i.e., faster than the Blondel algorithm).

EigenMatch

The main result of the **OLGA** algorithm is that it approximately reduces graph matching to the problem of aligning the dominant eigenvectors of the two graphs to be matched. This raises the question: why not try to align more than just the dominant eigenvectors? Accordingly, we analyze the optimization function *err* in 2. As $Tr[ww^T] = \| w \|_F^2$,

$$\begin{aligned}
& \min_P \| w_A - Pw_BP^T \|_F^2 \\
= \; & \min_P Tr[(w_A - Pw_BP^T)(w_A - Pw_BP^T)^T] \\
= \; & \min_P Tr[w_Aw_A^T + Pw_BP^TPw_B^TP^T \\
& \quad - w_APw_BP^T - Pw_BP^Tw_A] \\
= \; & \| w_A \|_F^2 + \min_P \| Pw_BP^T \|_F^2 \\
& \quad - Tr[w_APw_BP^T + Pw_BP^Tw_A]
\end{aligned}$$

As the trace of the product of two square matrices is independent of the order of multiplication, $Tr[w_A(Pw_BP^T)] = Tr[(Pw_BP^T)w_A]$. Also, $\| w_A \|_F^2, \| Pw_BP^T \|_F^2$ are terms related to the magnitude of the matched subgraphs, while the latter two terms pertain to the structure of the matching. Hence the problem reduces to $\max_P Tr[w_APw_BP^T]$. If w_A, w_B are normal matrices, then using eigen decomposition,

$$\begin{aligned}
& \max_P Tr[w_APw_BP^T] \\
= \; & \max_P Tr[U_AD_AU_A^TPU_BD_BU_B^TP^T] \\
= \; & \max_P Tr[(D_AU_A^TPU_BD_BU_B^TP^T)U_A] \\
= \; & \max_P Tr[D_A(U_A^TPU_B)D_B(U_B^TP^TU_A)] \\
= \; & \max_W Tr[D_AWD_BW^T] \; where \; W = U_A^TPU_B
\end{aligned}$$

Blondel et al. (2004) use normalized even iterates of $S_{k+1} = w_AS_kw_B$ to find similarities between normal matrices w_A, w_B. We adopt this idea, so that $W_{k+1} = D_AW_kD_B$. We drop the normalization as it is a constant for a single iteration. However, instead of an iterative algorithm, we choose a good seed and utilize just one iteratio'—$W_1 = D_A W_0 D_B$. For the seed, we use the **FastOLGA** algorithm (line 2), which aligns the dominant eigenvectors. Substituting in W, we get $D_A W_0 D_B = U_A^T P U_B$. Rearranging, we get

$$P = U_A D_A W_0 D_B U_B^T, \ where \ W_0 = FastOLGA(w_A, w_B) \tag{8}$$

$U_X D_X = [U_{X,1}\lambda_{X,1} \ \ U_{X,2}\lambda_{X,2} \cdots]$. Thus, each eigenvector of w_A and w_B will then be weighted by its eigenvalue. Then during rounding of P, the matching algorithm will be fully cognizant of the smaller eigenvalues as well.

Accordingly, we have the algorithm **EigenMatch** as shown in Figure 2. This algorithm has the same time complexity as eigen decomposition—that is, $O(n^3)$ (Golub & Van Loan, 1996).

Matching Using MonteCarlo Sampling

One way of estimating the unusualness of matchings produced by our algorithms involves generating random **matchings** and comparing the *err* value of the best of these, with that produced by our algorithms. Accordingly, if $|V_A|$>=}$|V_B|$, we generate a random permutation of the numbers [1,$|V_A|$] and map the first $|V_A|$ numbers of this permutation to the vertices numbered [1,$|V_B|$] of G_B. Otherwise, we swap the graphs and get the mapping in the same way. We call this ***MonteCarlo sampling.***

We repeat this sampling a number of times, evaluate them using the *Zscore* described further on, and keep the one with the best *Zscore*. The number of such samples generated is controlled by the time taken to run **OLGA**. This ensures that **OLGA** and **MonteCarlo sampling** have the same amount of time to find the matching.

Figure 2. Matching all eigenvectors

```
EigenMatch ($G_A, G_B$):
1. $w_A = U_A D_A U_A^T, w_B = U_B D_B U_B^T$
2. $W_0 =$ FastOLGA($w_A, w_B$)
3. $P = U_A D_A W_0 D_B U_B^T$
4. return Match($P$)
```

Experiments

In evaluating the performance of the algorithms, we pay attention to the following measures:

- **Execution time**
- **Number of matches (#(matches)):** It is the number of D_B's matchable components that are correctly matched. A component in D_B is *matchable* if there exists a known, unusually similar component in D_A.
- ***Zscore*:** We estimate the distribution of $err(f|w_A,w_B)$ (Equation 2) by generating a number of **matchings** using **MonteCarlo sampling** and computing the *err*. Using this distribution, the mean and standard deviation can be determined, and the scores corresponding to the mapping found by an algorithm are normalized to get the *Zscore*. Thus, the *Zscore* is the number of standard deviations from the mean. Very negative *Zscore* implies that the corresponding matching is very unlikely to have happened by MonteCarlo sampling, and such a matching is said to have found *unusually similar* substructure.

Experiments on **OLGA**, Blondel's algorithm, and MonteCarlo sampling were carried out on a SUN Sparc 650 MHz machine running on Solaris O/S with 256 MB RAM in C++. Blondel's algorithm, **EigenMatch, FastOLGA,** and MonteCarlo sampling were also implemented on a Pentium 2 GHz machine running on Windows XP with 256MB RAM in Matlab.

Synthetic Datasets

We use synthetic datasets to test the performance of our algorithms and similarity measures, as dataset and algorithm parameters are varied. By generating the datasets ourselves, we can verify the correctness. Our program for generating synthetic datasets is based on that previously described in Sequeira and Zaki (2004). It has the following set of parameters:

1. Average number of dimensions (d)
2. Average number of points in a dataset (n)
3. Average number of embedded subspaces (k)
4. Average probability that a subspace is constrained in a dimension (c)
5. Average probability that a subspace is constrained in the same dimension as the previous subspace (o)

6. Amount of perturbation (p)
7. Type of transformation

First, $1.5k$ subspaces are generated one after the other. They are by default multivariate normal, with means in each dimension $\mu(j \in [1,d])$, chosen from U[0,1000), where U[l,h) implies a uniform distribution over the interval [l,h). The standard deviation in each dimension $\sigma(j \in [1,d])$ is by default set to 20. A dimension is constrained with probability c. Two serially generated subspaces are constrained in the same dimension with probability o. Their means are constrained to be within 2 standard deviations of each other, to allow overlapping of subspaces. Unconstrained dimensions have means chosen from U[0,1000).

For $i \in \{1,2\}$, for dataset D_i, n_i, k_i are chosen uniformly from U($5n$,$1.5n$) and U($5k$,$1.5k$) respectively. The first k_i subspaces are embedded in D_i after perturbing their parameters using a transformation. There are three types of transformations:

- **Noisy:** $\forall j \in [1,d], \mu(j) = \mu(j) + U(-p,p)*1000$
  $\sigma(j) = \sigma(j)*(1+U(-p,p))$
- **Translation:** $\mu(j) = \mu(j) + i*p*1000$
- **Scaling:** $\sigma(j) = \sigma(j)*(1+ip/5)$

where p is the perturbation parameter. Each embedded subspace accounts for at least 1% of the total number of points. The actual number of points corresponding to a subspace is a function of the imbalance factor,

$$a, a = \frac{\max_l \alpha_l}{\min_l \alpha_l}$$

where α_l is the fraction of D_i generated using parameters of the l^{th} subspace embedded in D_i. Noisy points, which account for 5% of the points in D_i, are multivariate uniform—that is, each coordinate is chosen from U[0,1000).

In experiments shown below, we assume that the subspace mining algorithm finds the embedded subspaces correctly, so as to isolate the contributions of this chapter. Thus, we test only the graph creation and matching algorithms described in this chapter. We tested the algorithms by matching synthetic datasets having embedded subspaces. As we serially insert subspaces, for every pair of datasets, we ensure that the dataset with the larger number of embedded subspaces includes all subspaces embedded in the other dataset. The datasets have, on average, n=1000 points and d=50 dimensions and k=25 embedded subspaces, except those with k>40 subspaces, which have n=10000 points. Unless otherwise stated, c=o=0.5, p=0.03, a=4.0, we use the noisy transformation and Gaussian weighted *sim*()function. By default, we try to map a 27-vertex graph to a 34-vertex one using **OLGA** and the HIST similarity measure. For **OLGA,** we set τ=.925, *k*=30. We evaluate the algorithms based on the #(matches) and its *Zscore*, as some parameter in the dataset is varied or the subspace similarity function is varied.

Comparison of Similarity Functions

We first tested the similarity functions by attempting to match each graph to itself. As expected, we found that while the linear algebra-based algorithms succeed in doing so, the MonteCarlo sampling often does not. We have not shown these results, due to space constraints.

In Figures 3, 5, and 7 we compare **OLGA**'s performance in terms of #(matches) as some parameter, namely, *p*,*o* and *c*, used in generating the embedded subspaces is varied. Note that #(matches) is virtually the same for both measures, except parameter *p*, where HIST performs better at p>0.05. It also outperforms AVGSIM in terms of *Zscore* as seen in Figures 4, 6, and 8. This suggests that HIST favors a more global solution as compared to AVGSIM. This occurs because it develops a profile for the entire subspace, including dimensions for which the subspace is not constrained, whereas AVGSIM takes more of a discretized approach. Also, there exist some values of these parameters, for which HIST's *Zscore* drops below that of the optimal matching, in spite of having a significantly lower #(matches). This happens for extreme settings of the parameters. It suggests that *Zscore* and hence *err* quality measures are best suited to matching datasets having a low amount of perturbation.

In Figures 9 and 10, we compare the effect that different transformations on the dataset have on similarity measures. We notice that AVGSIM is consistently outperformed by HIST in the *Zscore* category, emphasizing the robustness of the latter. In terms of #(matches), HIST is outperformed for the noisy transformation because again it tries to optimize globally. Thus, in general HIST outperforms AVGSIM.

Comparison of Support-Based Similarity Functions

Early in this chapter, we discussed three functions used to compare clusterings. We then showed how to use them to find support-based subspace similarity. From Figure 11, Jaccard Index outperforms Rand Index and VI Metric.

Comparison of Mapping Algorithms

Firstly, we found experimentally that **FastOLGA** and Blondel's algorithm always arrive at the identical matching, suggesting that the similarity transformation found by Blondel's algorithm is basically the tensor product of the dominant eigenvectors. Note however that our algorithm is theoretically faster than Blondel's. In view of this result, we show their results combined except for the timing results.

In Figures 12, 13, and 14, we compare the performance of **OLGA** and **EigenMatch** with that of Blondel's algorithm (Blondel et al., 2004) and the best matching produced in terms of *Zscore* by MonteCarlo sampling. Note that for Figure 12, the log scale is used for the y-axis. Although **OLGA** is the most consistent performer, the best matching produced by MonteCarlo sampling (denoted in the figures as ``best MonteCarlo") performs well for matching small graphs, as it has a smaller state space to search. In Figure 13, **EigenMatch** outperforms the others in minimizing the *Zscore* more often than not. However, **EigenMatch** is unreliable in terms of #(matches) it produces. It sometimes produces no matches, while for k =75, it perfectly matches 13 vertices. This is because it attempts global optimization in trying to align all the eigenvectors. **OLGA,** by virtue of using the *sim* function, prunes the graph and hence tries to find unusually similar matches. Hence, it typically outperforms Blondel's algorithm. Also note that while Blondel's algorithm converges faster than the other algorithms, it is provably slower than **FastOLGA** and produces the same results. We have verified this using our Matlab simulation, but have not shown it in the graph, as efficient simulation of **OLGA** in Matlab is non-trivial.

All the algorithms have running time independent of n and d. Hence, results for these are not shown.

Figure 3. #(matches) v/s p

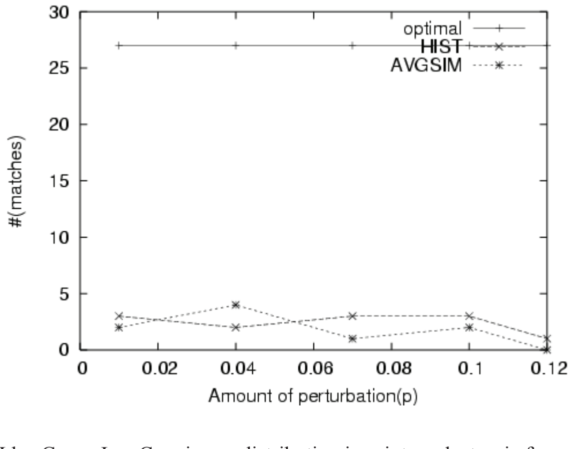


Figure 4. Zscore v/s p

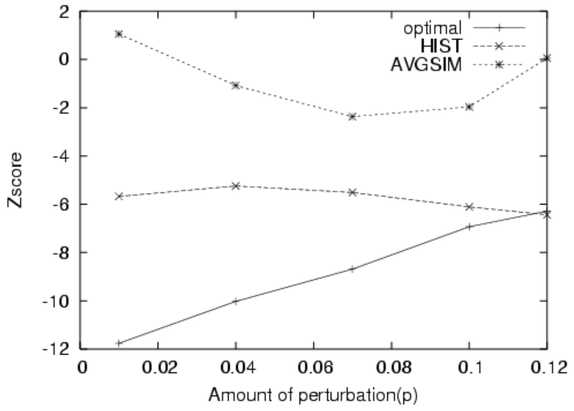


Figure 5. #(matches) v/s o

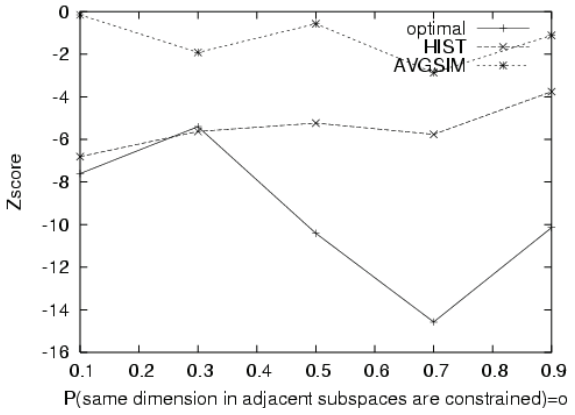


Figure 6. Zscore v/s o

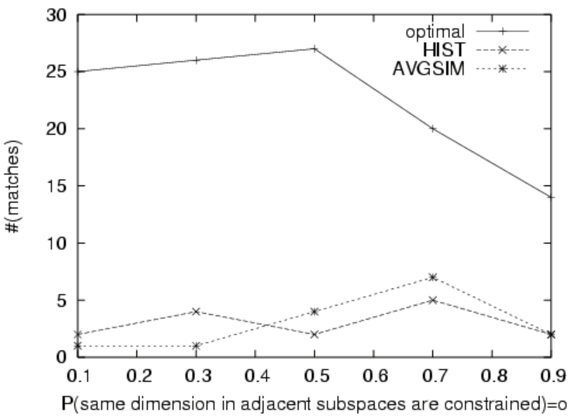


Figure 7. #(matches) v/s c

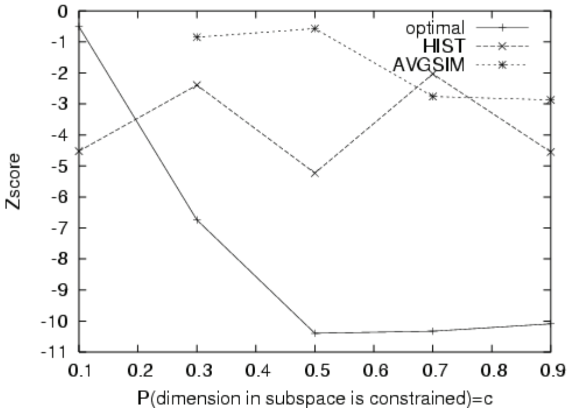


Figure 8. Zscore v/s c

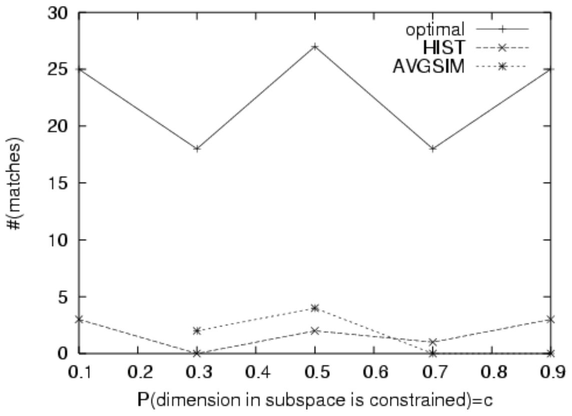


Figure 9. #(matches) v/s transformation

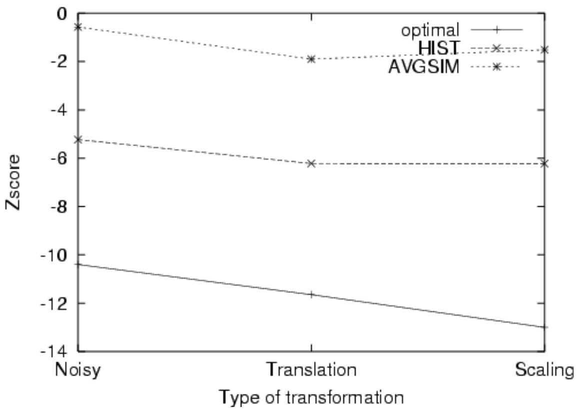


Figure 10. Zscore v/s transformation

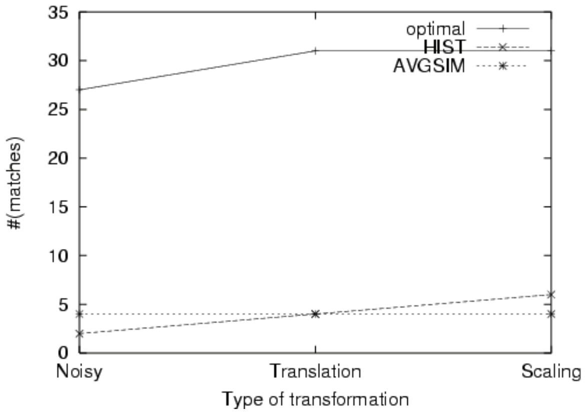


Figure 11. Clustering Comparison Functions

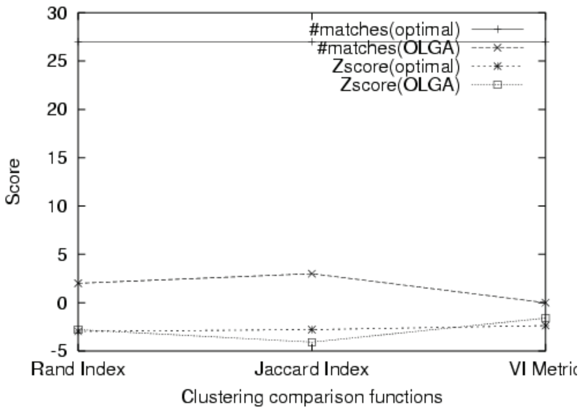


Figure 12. #(matches) v/s k

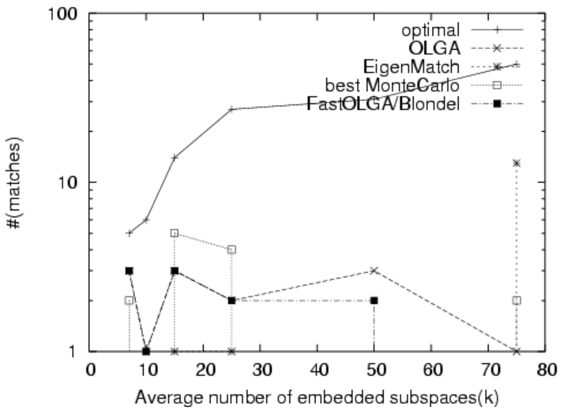


Figure 13. Zscore v/s k

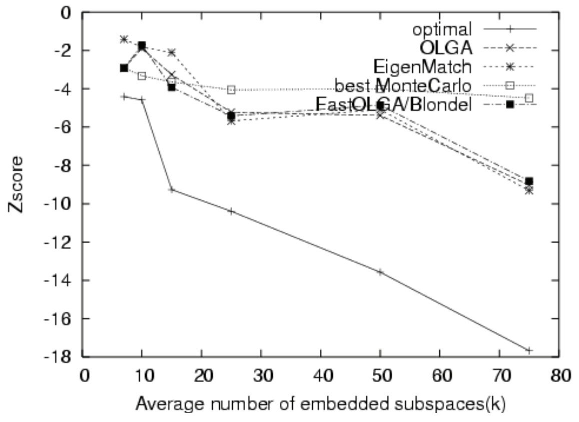
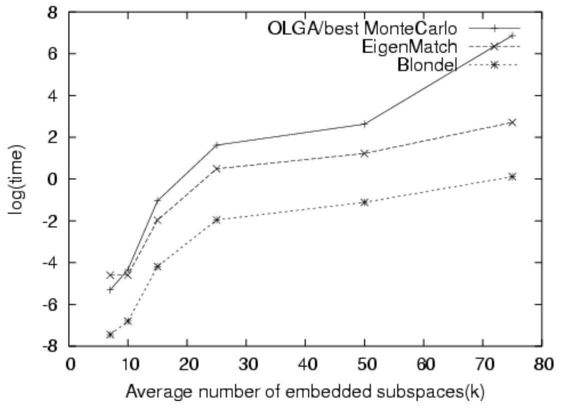


Figure 14. Time v/s k



Applications

As stated in the introduction to this chapter, our framework may be applied to monitoring evolution of datasets over time. Rather than use dataset snapshots, we use the statistics of players from the NBA, averaged annually from two consecutive basketball seasons, namely, 2003-04 (dataset A) and 2004-05 (dataset B). They are accessible from *http://sports.yahoo.com/nba/stats/*.

Another possible application of our framework is the mining of related but schematically differing datasets. We use two datasets pertaining to breast cancer (Beyer et al.,

1999) donated to the UCI ML repository at *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin,* obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

Finally, we apply our methodology on time series microarray datasets from the cell cycle of *S. Cerevisiae.*

NBA Data Analysis

Datasets A and B contain statistics for 443 and 464 players respectively. Each dataset has 16 columns:number of games played, average minutes played per game, average field goals, 3-pointers and free throws made and attempted, offensive and defensive rebounds, assists, turnovers, steals, blocks, personal fouls, and points per game. In this application, we seek to find groups of players having similar performance across the two seasons. If models of performance for the two seasons yield structurally similar clusters which overlap in their members, then these overlapping members are likely to have very similar performance. Consider the following scenario in which such knowledge may be employed: let players, say *E*, *F*, and *G*, currently belonging to distinct teams *P*, *Q*, and *R* respectively, all wish to leave their current team. If by our clustering models for the two seasons, it is known that players E and F belong to structurally similar clusters, then they show similar performance across the two seasons, prompting management at *P* and *Q* to consider "exchanging" them.

The statistics of each year yield a set of clusters/subspaces, 22 for dataset A and 15 for dataset B, for which we construct a graph using methods described above. We then structurally matched the two graphs using **OLGA.** For each pair of matched clusters, we report the intersection set of players. We found clusters as shown in Figure 4, preserved structurally, with respect to the rest of the dataset, across the two years. In basketball, there are primarily three positions at which the players play: 'center', 'forward', and 'guard'. Within these three positions there are further variants, like 'power forward', 'point guard', and so on. The position at which the NBA players played (i.e., *player position*) is not a column in our datasets. Examination of the cluster members revealed that the clusters primarily had members having the same player position.

For example, in the first cluster, out of six members, four—Curtis Borchardt, Ervin Johnson, Kendrick Perkins, and Stanislav Medvedenko—all play as 'centers'. Across both datasets, the probabilities of a randomly chosen player being either 'center', 'forward', or 'guard' are approximately the same and are given as p('center')=0.25, p('forward')=0.42, p('guard')=0.33. If the six players were drawn independently with replacement from this distribution, the probability that *k* of them are 'centers' is binomially distributed with parameters *n*=6 and *p*=0.25. Accordingly, the p-value of this cluster is bounded by the area of the tail of this distribution, to the right of *k*=4. Thus, p-value=

$$\sum_{k=4}^{n}\binom{n}{k}p^{k}(1-p)^{n-k}=\sum_{k=4}^{5}\binom{5}{k}(0.25)^{k}(0.75)^{5-k}=0.0562,$$

which may be considered to be statistically significant. As player position was not a part of the dataset, this analysis has provided us with a new insight. Also, it was found that all the players in the clusters do not start the game and come off the bench. As the players in the same cluster, as found by our algorithm, are typically in the same position, exchanging them seems very reasonable.

The cluster from dataset A corresponding to the first cluster in Table 4 has 50 players, of which 11, 25, and 14 are 'centers', 'forwards', and 'guards', respectively. These players are alike in that they belong in the lower third in terms of attempts at field goals, 3-pointers, and free throws; middle third for field goals made; and upper third for number of games played. Such a cluster has high entropy with respect to the player position. None of these categories singly yield statistically significant p-values. The same is true for the corresponding cluster from dataset *B* as well. The corresponding cluster in *B* has players belonging to the lower third in terms of attempts at field goals, 3-pointers, and free throws; number of field goals and free throws made; number of blocks and personal fouls; and average minutes per game. The six players reported in the table. Thus, structural alignment of the models for the datasets produces higher-entropy clusters with respect to those of the original models, with respect to the hidden variable (i.e., player position).

Table 4. Structurally similar clusters from two NBA seasons

Common Cluster Members	**Characteristic**	**p-value**
Curtis Borchardt, Ervin Johnson, KendrickPerkins, Stanislav Medvedenko, Walter McCarty, Lonny Baxter	4/6 are 'centers'	0.0562
Calbert Cheaney, Howard Eisley, Kendall Gill, Anfernee Hardaway, Jumaine Jones, Mickael Pietrus, James Posey, Kareem Rush, Theron Smith	7/9 are 'forwards'	0.062
Jeff Foster, Mark Madsen, Jamal Sampson	3/3 are 'centers'	0.0156
Brevin Knight, Tyronn Lee, Jeff McInnis, Latrell Sprewell, Maurice Williams	4/5 are 'guards'	0.0436

Table 5. Structurally similar clusters from schematically different breast cancer datasets and $p(M,u) \neq p(M,v)$*. Here* $v \in V_Y, u \in V_X$*, and* $\delta^2 = ((p(M,v) - p(M,u))^2$*.*

(v,u)	(0,31)	(2,23)	(4,13)	(8,3)	(11,24)	(14,34)	(16,25)	(17,28)	(19,35)	(20,2)
$p(M,v)$	1.0	0.021	0.933	0.5	0.97	1.0	0.833	0.833	1.0	0.8
$p(M,u)$	0.6	0.027	1.0	1.0	1.0	0.56	0.77	1.0	0.28	1.0
δ^2	0.16	0.0000036	0.0044	0.25	0.0009	0.193	0.0044	0.027	0.50	0.04

Breast Cancer Data Analysis

The first dataset (X) has nine dimensions/columns having integral values between 1 and 10 for clump thickness, uniformity of cell shape and size, marginal adhesion, bare nuclei, and so forth, and 699 samples/rows. There are a few missing values as well. Thirty-five percent of the samples are malignant (M) and the rest are benign. The second dataset (Y) has 30 dimensions/columns corresponding to three statistics (mean, standard error, max) for each of 10 real-valued features (radius, symmetry, area, texture, etc.) of the cell nuclei drawn from 569 samples/rows. Thus, the schema for X and Y is different. In Y, 37.25% are malignant and the rest are benign.

Each sample in both X and Y is labeled as either malignant or benign. Our goal is to discover these labels using unsupervised, rather than supervised learning techniques. Using SCHISM (Sequeira & Zaki, 2004), we find 36 clusters in X and 21 in Y. After creating the graphs and matching the clusters structurally, we examine the labels of the samples in matched clusters. Let $p(M, u \in V_X)$, $p(M, v \in V_Y)$ denote the probability that a sample drawn uniformly at random from clusters u,v respectively, from graphs corresponding to datasets X,Y respectively, is labeled malignant. Then if our framework finds that $P_f(u,v)=1$, from Equation 1—that is, the cluster u of X is matched to cluster j of Y—we found that $p(M,u) \approx p(M,v)$ (i.e., we found a strong correlation between labels of elements of matched clusters). In Table 5, we report the probabilities of $p(M,u \in V_X)$ and $p(M,v \in V_Y)$ $\forall P_f(u,v)=1$ and $p(M,u) \neq p(M,v)$. The first column, interpreted as cluster 0 of the second dataset (Y), has all its elements labeled as malignant, while cluster 31 of the first dataset (X) has three of its five elements (i.e., 3/5=0.6) labeled as malignant. Such findings allow us to search for correlations between the two spaces corresponding to X and Y. Although, the clusters found in both datasets are predominantly malignant, our algorithm correctly matches the benign ones—that is, cluster 2 of Y with cluster 23 of X, and the higher entropy clusters 16 of Y with 25 of X. A few of the clusters matched do not have a high correlation, as we forcibly attempt to match every cluster in Y to some cluster in X. Blondel's algorithm produces a worse mapping, in that it matches a cluster of malignant samples with a cluster of predominantly benign samples. We compare the results from the algorithms by measuring the correlation between the matched clusters using

$$corr(f) = \frac{\sum_{v \in V_Y} exp(-(p(M,v) - p(M, f(v)))^2)}{\sum_{v \in V_Y} exp(-1)}$$

Accordingly, we find $corr(f_{OLGA}) = 2.586$ and $corr(f_{BLONDEL}) = 2.0636$, where f_{OLGA} and $f_{BLONDEL}$ are the mappings produced by **OLGA** and Blondel's algorithm, respectively. Thus, **OLGA** outperforms Blondel's algorithm for the breast cancer dataset.

Microarray Data

With a large number of noisy, high-dimensional gene expression datasets becoming available, there is a growing need to integrate information from heterogeneous sources. For example, different clustering algorithms, designed to serve the same purpose, may be run on a dataset, and we may wish to integrate output from the two algorithms. Alternatively, the same algorithm may be run on two datasets differing only slightly in experimental conditions. In the first example, the algorithm provides heterogeneity, while in the latter, it is the experimental conditions.

In our specific application, we look at three microarray datasets, called GDS38, GDS39, and GDS124, pertaining to the *Saccharomyces cerevisiae* (yeast) cell cycle (to access the datasets, visit *http://www.ncbi.nlm.nih.gov/projects/geo/gds/gds_browse.cgi*). The datasets contain expression values of the different genes of yeast sampled over its cell cycle. The cultures are synchronized by different mechanisms, namely, alpha factor block-release(*A*), centrifugal elutriation (*E*), and cdc15 block release (*C*). GDS38 (i.e., *A*) has 16 samples/columns taken at seven-minute intervals, while GDS39 (i.e., *E*) has 14 samples/columns taken at 30-minute intervals, and GDS124 (i.e., *C*) has 25 samples/columns taken from almost three full cell cycles. Datasets A and E have 7,680 genes/rows, while C has 8,832 rows. The entry in the *i*th row and *j*th column of the dataset corresponds to the gene expression value for the *j*th time sample of gene *i* during the cell cycle of yeast. Microarray datasets are known to be very noisy. Also, these datasets have a large number of missing values as well.

It is hypothesized that genes which exhibit similar expression patterns may be coregulated—that is, having similar regulation mechanisms. Hence, we are looking for subspaces having similar expression patterns. We use SCHISM (Sequeira & Zaki, 2004) to find these subspaces. We use ξ=3. This discretizes gene expression values

Table 6. GO-based interpretation of similar substructure

Gene Ontology(GO) Term	p-value	Genes
chromatin silencing at telomere	0.00068	SUM1, BRE1
telomeric heterochromatin formation	0.00068	
gene, chromatin silencing	0.00215	
regulation of metabolism	0.00561	BRE1, ADR1, SUM1
organelle organization and biogenesis	0.00918	BRE1, ADR1, SUM1, SPC110
ribosome biogenesis	4.13e-05	MAK16, SPB4, CGR1 ...
ribosome biogenesis and assembly	0.0001	... TSR2, RLP7, NOP4
dicarboxylic acid transporter activity	0.00059	SFC1, DIC1
recombinase activity	0.00059	KEM1, RAD52
cytoskeletal protein binding	0.00251	NUM1, BNR1, ASE1, MLC2
transcription cofactor activity	0.00885	SPT8, SWI6, ARG81
signal transduction	0.01704	COS111, BEM2
DNA replication	0.00194	ECO1, DPB2
DNA repair	0.00402	
response to DNA damage stimulus	0.00551	
response to endogenous stimulus	0.00551	
growth	0.0078	TEC1, BEM2

into three categories: under-expressed (first interval), normal (second interval), and over-expressed (third interval). Thus, the subspaces correspond to a subset of the genes/rows which are simultaneously either under-expressed or normal or over-expressed for some subset of the time samples/columns. SCHISM returns 13 and 25 subspaces for datasets GDS38 and GDS39 respectively. We then construct the graphs for each dataset and match the underlying subspaces/vertices using **OLGA.** We examined the genes in the intersection of the matched subspaces to verify the efficacy of our algorithms. We submitted the list of genes in the intersection of the matched subspaces to the SGD Gene Ontology (GO) Term Finder (for details, see *http://db.yeastgenome.org/cgi-bin/GO/goTermFinder*) tool. This tool searches for significant shared GO terms, or parents of the GO terms, used to describe the genes in the submitted list of genes to help discover what the genes may have in common. A small sample of their results is shown in Table 6.

The first row of Table 6 is interpreted as follows: Genes SUM1 and BRE1 are associated with the process of chromatin silencing at telomere. These genes actually belong to a cluster of seven genes, but out of 7,274 genes in yeast, there are

42 involved in this process. Using the right tail of the binomial distribution, GO TermFinder reports the p-value (measure of statistical significance) as 0.00068. Further, they are also associated with gene silencing, and the p-value is 0.00215. SUM1 and BRE1 belong to a subspace of 193 genes when SCHISM is applied to dataset GDS38. This results in a much lower p-value and is hence not reported as statistically significant. This is true for other clusters reported too. Thus, the condensed model technique yields smaller, more statistically interesting clusters, by leveraging information from multiple sources.

Conclusion

From the *Zscore* values obtained by the algorithms, it is obvious that the algorithms find *unusually similar* matchings with respect to MonteCarlo sampling. The p-values of the inferences from the application to the NBA datasets confirm this. It is evident that **OLGA** and **EigenMatch** succeed in finding similar subspaces based on the structure of the dataset alone, without sharing the datasets. The experiments on the breast cancer data suggest that correlations between clusters in related datasets of differing schema may also be inferred, using our framework.

As part of future work, we hope to extend our algorithms to finding common substructure across multiple datasets. Also, currently our similarity measures are best suited to finding similarities between hyperrectangular subspaces. Patterns in datasets may require less restrictive descriptions, for example, coherent patterns in NBA datasets, curves, and so forth. We hope to develop similarity measures for such patterns as well.

Acknowledgements

This work was supported in part by NSF CAREER Award IIS-0092978, DOE Career Award DE-FG02-02ER25538, and NSF grants EIA-0103708 and EMT-0432098. Karlton Sequeira is now at Amazon.com, but the work was done while he was at RPI.

References

Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD Conference on Management of Data.*

Bay, S., & Pazzani, M. (2001). Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery, 5*(3), 213-246.

Bennett, K.P., & Mangasarian, O.L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software, 1,* 23-34.

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is nearest neighbors meaningful? In *Proceedings of the International Conference on Database Theory.*

Blondel, V., Gajardo, A., Heymans, M., Senellart, P., & Van Dooren, P. (2004). A measure of similarity between graph vertices: Applications to synonym extraction and Web searching. *SIAM Review, 46*(4), 647-666.

Bunke, H. (1999). Error correcting graph matching: On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(9), 917-922.

Carcassoni, M., & Hancock, E. (2002). Alignment using spectral clusters. In *Proceedings of the British Machine Vision Conference.*

Ganti, V., Gehrke, J., Ramakrishnan, R., & Loh, W. (1999). A framework for measuring changes in data characteristics. In *Proceedings of the ACM Symposium on Principles of Database Systems.*

Gionis, A., Mannila, H., & Tsaparas, P. (2005). Clustering aggregation. In *Proceedings of the IEEE International Conference on Data Engineering.*

Nagesh, H., Goil, S., & Choudhary, A. (2001). Adaptive grids for clustering massive data sets. In *Proceedings of the SIAM Data Mining Conference.*

Golub, G., & Van Loan, C. (1996). *Matrix computations* (3rd ed.). Baltimore, MD: Johns Hopkins University Press.

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques.* San Francisco: Morgan Kaufmann.

Kalviainen, H., & Oja, E. (1990). Comparisons of attributed graph matching algorithms for computer vision. In *Proceedings of the Finnish Artificial Intelligence Symposium.*

Kuhn, H. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly, 2,* 83-97.

Li, T., Ogihara, M., & Zhu, S. (2002). *Similarity testing between heterogeneous datasets.* Technical Report UR-CS-TR781, Computer Science Department, University of Rochester, USA.

Meila, M. (2003). Comparing clusterings by the variation of information. In *Proceedings of the International Conference on Learning Theory.*

Melnik, S., Garcia-Molina, H., & Rahm, E. (2002). Similarity flooding: A versatile graph-matching algorithm. In *Proceedings of the IEEE International Conference on Data Engineering.*

Neuwald, A., Liu, J., & Lawrence, C. (1995). Gibbs motif sampling: Detection of bacterial outer membrane repeats. *Protein Science, 4,* 1618-1632.

Sequeira, K., & Zaki, M. (2004). SCHISM: A new approach to interesting subspace mining. In *Proceedings of the IEEE International Conference on Data Mining.*

Shapiro, L., & Haralick, M. (1985). A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 7*(1), 90-94.

Van Wyk, B., & Van Wyk, M. (2003). Orthonormal Kronecker product graph matching. *Lecture Notes in Computer Science, 2726,* 107-117. Berlin: Springer-Verlag.

West, D. (1996). *Introduction to graph theory.* Englewood Cliffs, NJ: Prentice-Hall.

Wolberg, W.H., Street, W.N., Heisey, D.M., & Mangasarian, O.L. (1995). Computer-derived nuclear features distinguish malignant from benign breast cytology. *Human Pathology, 26,* 792-796.

Section II

Patterns