2010

Mining Shared Decision Trees between Datasets

Qian Han
Wright State University

Mining Shared Decision Trees between Datasets

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering

by

Qian Han
B.S., Department of Computer Engineering
Wuhan Polytechnic University, 2004

2010
Wright State University

Wright State University
SCHOOL OF GRADUATE STUDIES

March 10, 2010

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Qian Han ENTITLED Mining Shared Decision Trees between Datasets BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Computer Engineering.

Guozhu Dong, Ph.D.
Thesis Director

Thomas Sudkamp, Ph.D.
Department Chair

Committee on
Final Examination

Guozhu Dong, Ph.D.

Keke Chen, Ph.D.

Pascal Hitzler, Ph.D.

John A. Bantle, Ph.D.
Vice President for Research and
Graduate Studies and Interim Dean
of Graduate Studies

ABSTRACT

Han, Qian. M.S.C.E., B.S., Department of Computer Engineering, Wright State University, 2010. *Mining Shared Decision Trees between Datasets.*

This thesis studies the problem of mining models, patterns and structures (MPS) shared by two datasets (applications), a well understood dataset, denoted as WD, and a poorly understood one, denoted as PD. Combined with users' familiarity with WD, the shared MPS can help users better understand PD, since they capture similarities between WD and PD. Moreover, the knowledge on such similarities can enable the users to focus attention on analyzing the unique behavior of PD. Technically, this thesis focuses on the shared decision tree mining problem. In order to provide a view on the similarities between WD and PD, this thesis proposes to mine a high quality shared decision tree satisfying the properties: the tree has (1) highly similar data distribution and (2) high classification accuracy in the datasets. This thesis proposes an algorithm, namely SDT-Miner, for mining such shared decision tree. This algorithm is significantly different from traditional decision tree mining, since it addresses the challenges caused by the presence of two datasets, by the data distribution similarity requirement and by the tree accuracy requirement. The effectiveness of the algorithm is verified by experiments.

Contents

List of Figures

List of Tables

Introduction

This thesis studies the problem of mining models, patterns and structures (MPS) shared by two datasets, for the purposes of (1) understanding between the datasets and (2) gaining understanding of less understood datasets quickly.

We assume that we are given two datasets, one of the datasets, WD, is well understood, and the other dataset, PD, is poorly understood. The shared MPS can help users quickly gain useful insight on PD by leveraging their understanding and familiarity of WD, since the MPS capture similarities between WD and PD. Gaining such insight on PD quickly from the shared MPS can help the users to focus their main effort on analyzing the unique behavior of PD (see Figure 1.1), and to gain better overall understanding of PD quickly.

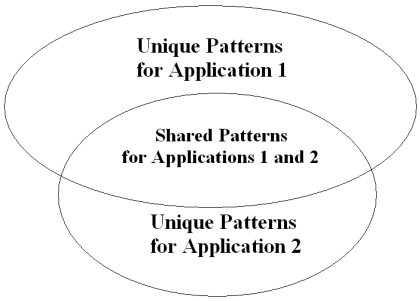

Figure 1.1: Shared and unique knowledge/patterns between two applications

The usefulness of this study has been previously recognized in many application domains. For example, in education and learning, the cross-domain analogy method has been recognized as an effective learning method [1][2]. In business and economics, a

country/company that lacks prior experience on economic/business development can adopt winning practices successfully used by countries/companies with similar characteristics [3][4]. In scientific investigations, researchers rely on cross-species similarities (homologies) between a well understood bacteria and a newly discovered bacteria, to help them to identify biological structures (such as transcription sites and pathways) in the newly discovered bacteria [5][6][7].

Despite its importance, previous studies have not systematically studied this problem, to the best of our knowledge. The references given above are only concerned with the use of shared similarity in applications. The learning transfer problem[1] (e.g. [8][9]), concerned with how to adapt and modify classifiers constructed from another dataset, is quilt different from our problem since we focus on mining shared models, patterns and structures.

For the sake of concreteness, the algorithmic part of this thesis will focus on mining of shared decision trees. Other forms of shared knowledge can be considered, including correlation/association patterns, graph-like interaction patterns, hidden Markov models, clusterings, and so on.

Specifically, this thesis proposes algorithms to mine high quality decision tree shared by two given datasets (WD and PD). A high quality shared decision tree is a decision tree that (1) has high classification accuracy on both WD and PD, and (2), to ensure that the tree captures similar knowledge structure in WD and PD, (the nodes of) the tree should partition WD and PD in a similar manner.

Besides motivating and defining the problem of mining shared models between applications, this thesis proposes an algorithm, namely SDT-Miner, for mining a decision tree shared by two datasets. The SDT-Miner algorithm addresses the challenges caused by the presence of two datasets, by the data distribution similarity requirement and by the tree accuracy requirement. We measure the quality of a mined shared decision tree using a weighted harmonic mean of average data distribution similarity, tree accuracy. Based on

[1] Learning transfer often assumes that the class label of data samples is unknown in the target dataset, this paper assumes that the class labels are known for the target datasets so that shared knowledge can be mined.

the above, it is clear that SDT-Miner is significantly different from traditional decision tree algorithms. The effectiveness of the algorithm is verified by experiments on synthetic and real world datasets. It should be noted that both the shared decision tree mining problem and *SDT-Miner* can be generalized to three or more datasets.

The rest of the paper is organized as follows: Section 1.1 gives a small illustrating example. Section II discusses related works and Section III provides the preliminaries. Section IV defines the general shared decision tree mining problem and the specific problem of mining a shared decision tree. Sections V presents the shared decision tree mining algorithm, namely SDT-Miner. An experimental analysis is given in Section VI. Section VII gives the conclusion of the thesis and lists some future research topics.

1.1 An Illustrating Example

To illustrate, consider the small example containing two datasets D_1 (as the WD) and D_2 (as the PD), shown in Table 1.1 and Table 1.2.

Figure 1.2 contains a decision tree, T, shared by D_1 and D_2. T has high classification accuracy (of 100%) in both D_1 and D_2, and has highly similar distributions at the tree nodes on data from D_1 and from D_2. (That is, for each tree node V, the class distribution of the subset of the data in D_1 meeting the condition of V is highly similar to that of the data in D_2 meeting that condition.) T is a decision tree shared by D_1 and D_2 of fairly high quality.

Table 1.1: Dataset D_1

TID	A_1	A_2	A_3	A_4	A_5	Class
1	3	6	2	3	4	C_1
2	2	2	9	5	6	C_1
3	7	5	8	8	12	C_2
4	4	8	15	6	9	C_2

Table 1.2: Dataset D_2

TID	A_1	A_2	A_3	A_4	A_5	Class
1	5	4	8	3	5	C_1
2	10	6	4	2	1	C_1
3	9	3	5	7	8	C_1
4	12	7	2	4	6	C_1
5	1	5	17	9	10	C_2
6	8	9	9	5	14	C_2

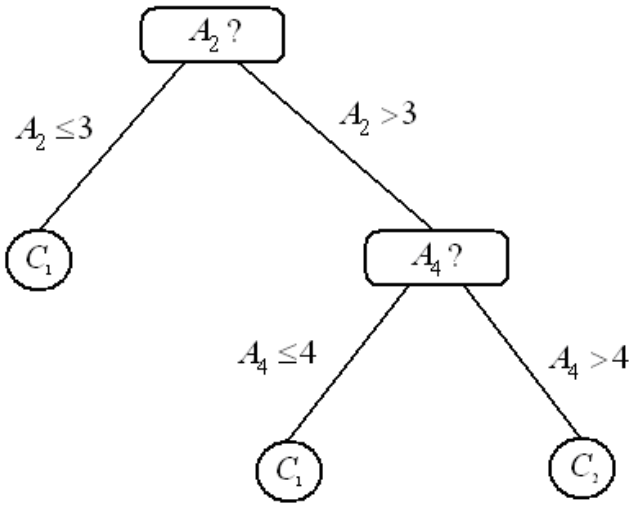

Figure 1.2: Shared decision tree T between D_1 and D_2

Related Work

Previous studies related to our work can be divided into two main groups.

Learning Transfer: The first group of related works consists of studies on learning transfer, which is mainly concerned with how to adapt/modify a classifier constructed from a source context for use in a target context. Reference [8] considers adapting EM-based Naive Bayes classifiers, for classifying text, built from a given context for use in a new context. Reference [9] proposes to combine multiple classifiers built from one or more source datasets, using a locally weighted ensemble framework, in order to build a new classifier for a target dataset.

Decision Tree: The second group of related works consists of studies on decision trees. This thesis studies the problem of mining models, patterns and structures (MPS) shared by two datasets. For the sake of concreteness, the algorithmic of this thesis will focus on mining of shared decision trees. But our shared decision tree mining problem is significantly different from traditional decision tree algorithms, it addresses the challenges caused by the presence of two datasets, by the data distribution similarity requirement and by the tree accuracy requirement.

Our study is also related to studying regarding cross-platform and cross-laboratory concordance of the microarray technology. For example, [10] studied how to use the transferability of discriminative genes and their associated classifiers to measure such concordance. However, to the best of our knowledge, no previous studies considered the use of shared decision trees to measure such concordance.

In summary, our aim is to mine shared patterns among multiple contexts, in order to help users see the similarity between multiple contexts, help them understand the new application using the similarity, and thereby help them focus their attention on unique knowledge patterns in the new applications.

Preliminaries

In this section we briefly review some concepts, and define several notations, regarding decision tree and information gain.

3.1 Decision Tree

A decision tree is a tree; each internal node of the tree denotes a test on an attribute (the splitting attribute of the node), each branch represents an outcome of the test, and each leaf node has a class label. Figure 1.2 give an example. The test of an internal node partitions the data of the node into a number of subsets, one for each branch of the node. A decision tree is built from a given training dataset, which consists of tuples with class labels. In this thesis we focus on binary decision trees to simplify the discussion, although our approach and results can be easily generalized.

We will use the following two notations below. Given a node V of a decision tree T and a dataset D, let $SC(V)$ denote the set of conditions on the edges in the path from the root of T to V, and $SD_D(V)$ the subset of D for V is defined by $SD_D(V) = \{t \in D \mid t$ satisfies all tests in $SC(V)\}$.

3.2 Information Gain

The information gain measure is often used to select the splitting attributes for internal nodes in the decision tree building process. For each internal node of the tree under construction, the attribute with the highest information gain is chosen as the splitting attribute.

The concept of information gain is based on expected information. Suppose V is an internal node of a tree and D_V is the set of data associated with V. Suppose the classes of the data are C_1, ..., and C_m. The expected information needed to classify a tuple in D_V is given by

$$Info(D_V) = -\sum_{i=1}^{m} p_i \log_2(p_i), \tag{3.1}$$

where p_i is the probability that an arbitrary tuple belongs to class C_i.

For binary trees, a splitting attribute A for V partitions D_V using 2 tests on A. The tests have the form $A \leq a$ and $A > a$, if A is a numerical attribute with many values and a is a selected split value. These tests split D_V into 2 subsets, D_1, D_2, where D_j = $\{\ t \in D_V \mid t[A]$ satisfies the j^{th} test for $V\}$. The information of this partition is given by

$$Info(A, a) = \sum_{j=1}^{2} \frac{|D_j|}{|D_V|} \times Info(D_j). \tag{3.2}$$

For each attribute A, let a_V denote the split value of A that yields the best (smallest) value of $Info(A, a)$ among all possible split values a of A. The information gain of A for node V is defined by

$$IG(A,\ a_V) = Info(D_V) - Info(A,\ a_V). \tag{3.3}$$

Problem Definition: Mining Shared Decision Tree

Roughly speaking, our aim is to mine a high quality decision tree shared by two datasets, which provides high classification accuracy and highly similar data distributions.

Before defining this problem, we first need to describe the input data for our problem, and introduce several concepts, including what is a shared decision tree, what is a high quality shared decision tree.

To mine decision tree shared by two datasets, we need two input datasets D_1 and D_2. D_1 and D_2 are assumed to share an identical set of attributes. For the case that they contain different sets of attributes, the user will need to determine equivalence between attributes of D_1 and attributes of D_2, and then map the attributes of D_1 and D_2 to an identical set of attributes using the equivalence relation and eliminate those attributes of D_i that have no equivalent attributes in D_j, $j \neq i$.

A shared decision tree is a decision tree, that can be used to accurately classify data in dataset D_1 and accurately classify data in dataset D_2.

A high quality shared decision tree is a decision tree that has high data distribution similarity, and has high shared tree accuracy in both datasets D_1 and D_2.

The concepts of data distribution similarity and shared tree accuracy are defined next.

4.1 Data Distribution Similarity

Data distribution similarity (DS) captures cross-dataset distribution similarity of a tree (DST). DST measures the similarity between the distributions of the classes of data in the two datasets in the nodes of the tree. It is based on the concepts of class distribution vector (CDV) and distribution similarity of a node (DSN).

We use the class distribution vector (CDV) for a node V of a tree T to describe the distribution of the classes of a dataset D_i at V, that is:

$$CDV_{\mathrm{i}}(V) = (Cnt(C_1, SD_i(V)),\ Cnt(C_2, SD_i(V))), \tag{4.1}$$

where Cnt (C_j, $SD_i(V)$)= $\mid \{t \in SD_i(V) \mid t's$ class is $C_j\}\mid$.

The distribution similarity (DSN) at a node V is measured by the similarity between the class distributions for the two datasets at V. It is defined as the normalized inner product of two CDV vectors for D_1 and D_2:

$$DSN(V) = \frac{CDV_1(V) \cdot CDV_2(V)}{\|CDV_1(V)\| \cdot \|CDV_2(V)\|}. \tag{4.2}$$

where $||CDV_i(V)||$ presents the norm of the vector $CDV_i(V)$, and $CDV_1(V) \cdot CDV_2(V)$ means the inner product of two vectors $CDV_1(V)$ and $CDV_2(V)$.

For example, suppose $SD_1(V)$ contains 50 tuples of Class C_1 and 10 tuples of Class C_2, and $SD_2(V)$ contains 10 tuples of Class C_1 and 5 tuples of Class C_2. Then $CDN_1(V)$=(50, 10), $CDN_2(V)$=(10, 5), and $DSN(V)$=0.88.

4.1.1 Cross-Dataset Distribution Similarity of Tree (DST)

Now we turn to define DST. There are different methods to define the cross-dataset distribution similarity of a shared tree according to the class distribution vector and distribution

similarity. These methods can be classified as follows:

A. All Node Measure

Firstly, all node measure, as implied by the name, considers the data distribution at all non-root nodes. This method pays attention to all non-root nodes since the data distribution similarity at the root is identical after the Equalizing Class Ratios process (Section VI). For this measurement, we can use either weight based method or vectors based method.

(1) Weight Based Method

Since different nodes may have different importance, we can use weights of nodes in defining DST to distinguish the difference. Formally, given a tree T, the DST is given as:

$$DST(T) = \frac{\sum_{i=1}^{n} DSN(V_i)w(V_i)}{\sum_{i=1}^{n} w(V_i)}, \tag{4.3}$$

where V_1, V_2, ...,V_n are all non-root nodes in tree T, $w(V_i)$ is the weight for node V_i.

About the weight assignment, two different methods can be applied to each node. One method is to assign equal weight to each node. The other method is to assign level based weight to each node. Specifically, level based weight means to assign higher weight to the nodes near the root of the tree, since the nodes near the tree root may be more important than the nodes near the leaves. These two weight assignment methods are also be described as:

(a) Equal weight: the weight vector $w(V_i) = 1$ in equation 4.3 for all non-root nodes V_i.

(b) Level based weight: weight vector $w(V_i) = 2^{-Lvl(V_i)}$ in equation 4.3 for all non-root nodes V_i.

(2) Vectors Based Method

Besides the weight based method, we can also define DST regarding vectors. For

each node V, we measure the associated class distributions vector, namely [c_{11}, c_{21}] and [c_{12}, c_{22}], where c_{i1} is the number of tuples from the first dataset D_1 in class C_i that satisfy all of the conditions on the path from the root to V, and similarly c_{i2} is that number for the second dataset D_2.

Each dataset D_i has a CDV for each node. In that way, for each specific dataset, all CDVs for every node could comprise of the class distribution vector for this dataset, namely CDV_i. In formula, CDV_i can be expressed:

$$CDV_i = (CDV_i(V_1),\ CDV_i(V_2),\ \ldots,\ CDV_i(V_n)), \tag{4.4}$$

where V_1, V_2, ...,V_n presents all non-root nodes in the tree.

Then the DST can be measured by the similarity between the vectors CDV_1 and CDV_2. It is defined as the normalized inner product of two CDV vectors for D_1 and D_2:

$$DST(T) = \frac{CDV_1 \cdot CDV_2}{\|CDV_1\| \cdot \|CDV_2\|}. \tag{4.5}$$

B. Leaf Node Measure

The other measurement of DST is leaf node measure, which only focuses on analyzing the data distribution similarity of all leaf nodes, and ignores the effect of the internal nodes. This method is under the assumption that leaf nodes are more important than other internal nodes in the classification of the shared decision trees. In this measurement method, equal weight based method and vectors based method are also applied.

(1) Equal Weight Based Method

This method applies the same idea compared to the equal weight method in "All Node Measure". The only difference is that leaf node measure is averaging the DSNs of all leaf nodes, instead of all non-root nodes.

(2) Vector Based Method

Compared to the same method in "All Node Measure", the CDV_i vectors are composed of CDVs of all leaf nodes. In formula, we have:

$$CDV_i = (CDV_i(V_1),\ CDV_i(V_2),\ \ldots,\ CDV_i(V_n)), \tag{4.6}$$

where V_1, V_2, ...,V_n present all leaf nodes. Then the DST can be measured by the same equation 4.5 in all node measure.

We now use an example to illustrate the above DST measures and analyze each method. The following Figure 4.1 presents a decision tree shared by datasets D_1 and D_2. For each node V_i, [c_{11}, c_{21}] is shown on the left of the node, and [c_{12}, c_{22}] is shown on the right. Table 4.1 and 4.2 list the weight based CDVs for datasets D_1 and D_2, respectively, and table 4.3 lists the vector based CDVs for datasets D_1 and D_2 and the corresponding DST.

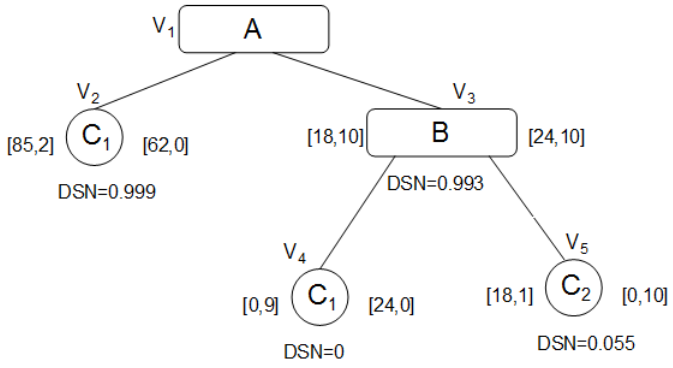

Figure 4.1: A shared decision tree

Now we analyze each method based on the results of different DST methods.

(1) For the equal weight method of all node measure, the DST is 0.51. This method considers all non-root nodes equally, and every node has an impact on the DST.

(2) For the level based weight method of all node measure, the DST is 0.67. This

Table 4.1: CDV_1

$CDV_1(V_2)$	[85,2]
$CDV_1(V_3)$	[18,10]
$CDV_1(V_4)$	[0,9]
$CDV_1(V_5)$	[18,1]

Table 4.2: CDV_2

$CDV_2(V_2)$	[62,0]
$CDV_2(V_3)$	[24,10]
$CDV_2(V_4)$	[24,0]
$CDV_2(V_5)$	[0,10]

method pays more attention to the nodes near the root, and pays less attention to the leaf nodes. From this example, it is observed that the DST even reaches to 0.67 although two leaf nodes obviously do not have any similarity.

(3) For the vector based method of all node measure, the DST is 0.898. The DST is affected by the node that has more tuples in it.

(4) For the equal weight method of leaf node measure, the DST is 0.35. This method only focuses on the leaf nodes and does not consider the influence of the internal nodes. From this example, it is observed that the DST reduces to 0.35 although there is a node with DSN$= 1$.

(5) For the vector based method of leaf node measure, the DST is 0.899. This method can not give the comprehensive view of the whole tree since it only observes the leaf node.

To present the data similarity between two datasets more accurately, we select the equal weight method of all node measure to calculate the DST in all of our following experiments.

Table 4.3: Vector Based Method

	All Node Measure	Leaf Node Measure
CDV_1	[85,2,18,10,0,9,18,1]	[85,2,0,9,18,1]
CDV_2	[62,0,24,10,24,0,0,10]	[62,0,24,0,0,10]
DST	0.898	0.899

4.2 Tree Accuracy

We define shared decision accuracy of a tree T as the minimum of the two tree accuracies of T on the two given datasets:

$$Acc_{D_1,D_2}(T) = \min(Acc_{D_1}(T), Acc_{D_2}(T)). \tag{4.7}$$

where $Acc_{D_j}(T)$ is the accuracy of T on dataset D_j.

$Acc_{D_j}(T)$ is defined by:

$$Acc_{D_j}(T) = 1 - \frac{|W|}{|D_j|} \tag{4.8}$$

where $\frac{|W|}{|D_j|}$ is the error rate for dataset D_j, W is the set of tuples classified wrongly in the leaf nodes of T, and D_j is the set tuples in dataset D_j.

Using this definition, a tree with high tree accuracy will have high classification accuracy on both datasets.

To obtain the set of tuples classified wrongly in the leaf nodes, it is crucial to determine the class for each leaf node. As we known, in traditional decision tree algorithms, the class of a leaf node is assigned by the majority class of that node. Leaf nodes of shared decision tree have data from two datasets, hence there is one majority class for each dataset. For one leaf node, when the majority classes of two datasets are the same, we simply pic that majority class. However, when the majority classes of two datasets are different, we

determine the class label of this leaf node in a way to minimize the overall error, considering both datasets.

The following two figures present two scenarios in which we need to figure out the classes of leaf nodes. The first scenario shown in figure 4.2 describes that one child from the parent is a leaf node and the other child could continue to split; while the second scenario in figure 4.3 shows that two children from the parent are both leaf nodes.

In figure 4.2, we only need to determine the class of left child since the right child is continued to split. There are two cases for the left leaf node class. When it is assigned by C_1, the number of tuples classified wrongly in dataset D_1 is 10 and in dataset D_2 is 15. Then the error rate in this leaf node of D_1 and D_2 is computed to be 0.59. On the other hand, when assigned by C_2, the number of tuples classified wrongly in dataset D_1 is 14 and in dataset D_2 is 0. Then the error rate of D_1 and D_2 becomes to 0.24. Comparing the error rate 0.59 with 0.24, the class of this leaf node should be assigned by C_2 to minimize the overall error rate.

Considering the second scenario in figure 4.3, we need to determine the classes of both leaf nodes. There are two cases for the classes. When the class of left leaf node is assigned by C_1 and the class of right node is assigned by C_2, the number of tuples classified wrongly in two leaf nodes of datasets D_1 and D_2 are 3, 17, respectively. Then the error rate of D_1 and D_2 is 0.23. On the other hand, when assigning C_2 for left leaf node and C_1 for right node, the number of tuples classified wrongly in two leaf nodes of datasets D_1 and D_2 are 17, 5, respectively. Then the error rate of both datasets becomes to 0.34. To minimize the overall error rate, the class of left leaf node should be assigned by C_1 and the class of right leaf node should be assigned by C_2.

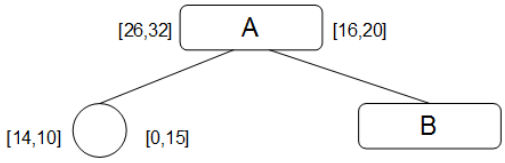

Figure 4.2: Tree T_1

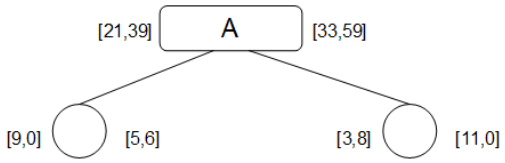

Figure 4.3: Tree T_2

4.3 Combining the Factors to Define Tree Quality

The quality of shared decision tree is influenced by two factors: data distribution similarity and shared tree accuracy. We need to combine these factors into one number, to facilitate the comparison of the quality values.

Several combination methods can be used, including the arithmetic mean (the average), the geometric mean (the root of the product), and the harmonic mean (detailed below).

We select the harmonic mean method to combine the factors for the following reasons.

The harmonic mean pays more attention to the smallest of the factors than the other two methods. Among the two factors we consider, the tree accuracy factor is the most important and it often has the smallest value among the factors.

We may also want to control the degree of importance of the factors in the harmonic mean. This can be done using the weighted harmonic mean. The weighted harmonic mean of n positive values $x_1,\ldots, x_n$ is defined by:

$$\sum_{i=1}^{n} w_i (\sum_{i=1}^{n} \frac{w_i}{x_i})^{-1}, \tag{4.9}$$

where w_i is weight assigned to x_i. We will discuss how to select the weights shortly.

Below we will use WHM as short-hand for weighted harmonic mean. Moreover, we will use w_{DS} to denote the weight assigned to data distribution similarity, w_{TA} the weight assigned to shared tree accuracy.

Definition 1. *($SDTQ_{WHM}$)*

The weighted harmonic mean based quality of a shared decision tree T is defined by:

$$SDTQ_{WHM}(T) = \frac{w_{DS} + w_{TA}}{\frac{w_{DS}}{DS_{(T)}} + \frac{w_{TA}}{TA_{(T)}}}. \tag{4.10}$$

To determine the weights on the factors, we evaluated how the SDTQ_{WHM} values respond to the different factor value combinations when different weight vectors are used. We found that (w_{DS},w_{TA})=(1,1) for $SDTQ_{\text{WHM}}$ is good choice; since this weight vector pays equal attention to the tree accuracy and distribution similarity factors. Therefore, in all of our experiments, we use this weight vectors to define SDTQ_{WHM}.

4.4 The Shared Decision Tree Mining Problem

We are now ready to define the shared decision tree mining problem that we will study in this thesis. Our goal is to mine high quality decision tree that exhibits patterns/models shared by two given datasets.

Definition 2. *(The SDT Mining Problem)*

Given two datasets D_1 and D_2 with an identical list of attributes, the shared decision tree mining problem is to mine one shared decision tree T such that SDTQ$_{WHM}(T)$ is high; that is T has highly similar data distribution and high tree accuracy in the two datasets.

Shared Decision Tree Miner

(*SDT-Miner*)

This section introduces the *Shared Decision Tree Miner* (*SDT-Miner*) algorithm, for mining a decision tree shared by two datasets.

Roughly speaking, to split a node, *SDT-Miner* uses a splitting attribute/value pair to maximize an objective function that combines data distribution similarity and information gain.

While *SDT-Miner* is similar to C4.5 in the tree building process[2], it differs from C4.5 (i) concerning purpose (mining a decision tree shared by two datasets vs mining a decision tree for a single dataset), and (ii) regarding two new ideas on how to select the splitting attribute (it selects attributes (a) with high data distribution similarity in two given datasets, and (b) with high information gain in two given datasets).

SDT-Miner (see Algorithm 1) has four input parameters: Two Datasets (D_1 and D_2), a set ($AttrSet$) of candidate attributes for use in shared decision trees, a dataset size threshold ($MinSize$) for split termination. *SDT-Miner* builds a shared decision tree by using the *SDTNode* function (see Function 1) recursively.

SDTNode splits the data of a tree node by picking the best splitting attribute for the data. To obtain shared decision tree with high data distribution similarity and high tree

[2]For each tree node, C4.5 finds the attribute that maximizes information gain as the splitting attribute.

Algorithm 1. SDT-Miner

Input: Two Datasets: D_1, D_2;
AttrSet: Set of candidate attributes for use in shared decision trees;
MinSize: Dataset size threshold for splitting termination;
Output: A shared decision tree for D_1 and D_2.
Method:
1. Create root node V;
2. Call SDTNode(V, D_1, D_2, AttrSet, Minsize);
3. Output the shared decision tree rooted at V.

accuracy, *SDTNode* uses a DI scoring function to help determine what the best splitting attribute is. Roughly speaking, DI is defined to be a sum of data distribution similarity and information gain. Specifically, let V be a node (to split), A a candidate splitting attribute for V, a_V a candidate split value for A. Then DI can be defined by:

$$DI(A, a_V) = DSN(A, a_V) + IG(A, a_V). \tag{5.1}$$

We now explain three things about the *SDTNode* function.

(1) The *SDTNode* function (line 1) uses another function called *ShouldTerminate* to determine if splitting should stop at a given node V . We designed *ShouldTerminate* to help avoid building "overfitting" trees, in addition to checking other normal termination conditions. Specifically, $ShouldTerminate(V, D'_1, D'_2, MinSize)$ returns "true" if (a) either D'_1 or D'_2 is pure[3], or (b) all available attributes have been used in the path from the root of the tree to V, or (c) either $|D'_1| \leq MinSize$ or $|D'_2| \leq MinSize$. Conditions (a) and (b) are normal splitting termination conditions for decision tree construction. Condition (c) is used to stop splitting the node if the datasets for the node are very small, which helps avoid overfitting.

(2) Selecting the attribute B and split value b_V to maximize DI (line 3) ensures that the split attribute/value lead to fairly high DS and IG. Experiments confirmed that this

[3]A dataset is pure if all of its tuples belong to a common class.

Function 1. SDTNode(V, D'_1,D'_2, AttrSet, MinSize)

1. If ShouldTerminate(V, D'_1, D'_2, MinSize) then assign
   // determine shared class label for both D_1 and D_2 at V
2. the majority class in D'_1 and D'_2 as class label of V and return;
3. Select the attribute B and split value b_V that maximize DI, that is
4. $DI(B,b_V)$=max$\{DI(A,a_V)|A \in AttrSet$ and A was,
5. not used on the path from the root of the tree to V,
6. a_V is a common candidate split value for A at $V\}$;
7. Let B and b_V be the splitting attribute/value for V;
   // compute the left subtree of V
8. Create leaf child node V_l for V;
9. Let "$B \leq b_V$" be the test on the left outgoing edge of V;
10. Let D'_{il} = $\{ t \in D'_i \mid$ t satisfies "$B \leq b_V$" $\}$ for i = 1, 2;
11. Call SDTNode(V_l, D'_{1l},D'_{2l}, $AttrSet$, $MinSize$);
    // compute the right subtree of V
12. Create right child node V_r for V;
13. Let "$B > b_V$" be the test on the right outgoing edge of V;
14. Let D'_{ir} = $\{ t \in D'_i \mid$ t satisfies "$B > b_V$" $\}$ for i = 1, 2;
15. Call SDTNode(V_r, D'_{1r},D'_{2r}, $AttrSet$, $MinSize$);

Function 2. ShouldTerminate(V, D'_1, D'_2, MinSize)

1. Return true if at least one of the following three conditions is true:
2. (a) D'_1 is pure or D'_2 is pure;
3. (b) All attributes in $AttrSet$ were used on the path from the root to V;
4. (c) $|D'_1| \leq MinSize$ or $|D'_2| \leq MinSize$; // the datasets of V are small
5. Return false. // none of the above three conditions is true.

heuristic can help us find high quality trees.

(3) The candidate common split values (*CCSV*) for A at V (line 6) are determined by considering the A values in both datasets for V. They are the mid points of consecutive such A values: If $v_1, v_2, \ldots, v_n$ are the distinct values of A in $D'_1 \cup D'_2$ in increasing order, then $(v_i, v_i + 1)/2$ is a candidate common split value for each i.

We now use the following example to illustrate how to find the *CCSV* between two datasets. Two datasets D_a and D_b are given in the following tables 5.1 and 5.2. Table 5.3 lists the *CCSV* for both datasets D_a and D_b.

Table 5.1: Dataset D_a

value of A_i	Class
1	C_1
3	C_2
5	C_1
5	C_1

Table 5.2: Dataset D_b

value of A_i	Class
2	C_2
4	C_1
8	C_2
10	C_1

Table 5.3: CCSV

CCSV
1.5
2.5
3.5
4.5
6.5
9

Experimental Evaluation

This section presents experiment results on some real-world and synthetic datasets to evaluate the performance of our algorithms and to illustrate the trees mined by our algorithms.

The experiments were conducted on a 2.20 GHz AMD Athlon with 3 GB memory running Windows XP, and the codes were implemented in Matlab.

6.1 The Datasets

6.1.1 Real Datasets

Our experiments on real world datasets used six microarray datasets for various diseases (mostly cancers). Table 6.1 lists the names of these datasets, and their sizes (number of tuples); more details are provided in Section 6.3.2. Each tuple in such a dataset is a microarray measurement of the gene expression levels in a patient sample; each column is the gene expression level for a gene of in that patient sample. We normalized columns (genes) so that each column of each dataset has a mean of zero and a standard deviation of one. This was done to make value changes of different genes more comparable.

To mine decision trees shared by pairs of microarray datasets, we need to identify equivalent attributes (genes) for each pair. We used the ArrayTrack software [11] to do that. Equivalent genes are different names in different gene name systems for the same gene. Table 6.2 lists the number of equivalent attributes for various dataset pairs.

Table 6.1: Datasets

Datasets	Number of tuples
Brease Cancer (BC)	19
Central Nervous System (CN)	60
DLBCL-Harvard (DH)	58
Lung Cancer-BAWH (LB)	32
Lung Cancer-Michigan (LM)	96
Prostate Cancer (PC)	21

Table 6.2: Number of equivalent attributes

Dataset pairs	Number of equivalent attributes
BC vs CN	5114
BC vs DH	5114
BC vs LB	8123
BC vs LM	5114
BC vs PC	8124
CN vs DH	5555
CN vs PC	5317
DH vs LB	5313
LB vs PC	9030
LM vs PC	5317

6.1.2 Equalizing Class Ratios

After identifying equivalent attributes (genes) from those dataset pairs, we noticed that the class distributions of two datasets may have huge difference. The big difference may can have a big impact on the quality value of the shared decision trees, making it difficult to compare quality values for shared trees mined from different dataset pairs. To solve this problem, we modify the datasets using the sampling with replacement method. More specifically, we replicate tuples of one class to the dataset that contains fewer tuples, to make the class distributions of two datasets nearly equal. The method is given in Algorithm 2.

Algorithm 2. Equalizing Class Ratios

Input: Two Datasets: D_1 and D_2;
Output: Two new generated datasets D_1^* and D_2^*.
// D_1^*, D_2^* are extensions of D_1, D_2, with near equal class distributions.
Method:

1. Let c_{ij} =$|t \in D_j|$t's class is $C_i|$, $R_j = \frac{C_{1j}}{C_{2j}}$, $D_1^* = D_1$, $D_2^* = D_2$;
2. Let D_s be the smaller dataset among D_1 and D_2,
   and D_t be the other dataset;
   // C_r denotes the class of D_s that we will add tuples to;
3. If $R_s > R_t$, then let C_r be C_2, and m=$\lceil \frac{c_{1s}}{R_t} - c_{2s} \rceil$;
4. Else let C_r be C_1, and m=$\lceil c_{2s} \cdot R_t - C_{1s} \rceil$;
5. Using sampling with replacement, we select m tuples from C_r of D_s,
   and then add these tuples to class C_r of D_s^*;

6.1.3 Synthetic Datasets

Since the real microarray datasets for cancers usually contain very few tuples, and the datasets from UCI often contain very few attributes, they cannot be directly used for scalability experiments. To solve this problem, we generate some synthetic datasets from those real microarray datasets using the method given in Algorithm 3.

Suppose we want to make a synthetic dataset with N tuples. We will make M=$\lceil \frac{N}{N_T} \rceil$ copies of each tuple t of D. Each tuple's attribute values will be changed randomly, and the magnitude is less or equal than P as a percentage of the corresponding old values. In order to get attributes with varying degrees of similarity to the old attributes, we also use an upper-bound of change, P_i, for each attribute A_i, which is smaller than P.

6.2 Performance Analysis Using Synthetic Datasets on Execution Time

We now use experiments to evaluate how computation time of SDT-Miner changes when the number of tuples increases. The experiments use synthetic datasets built using Algo-

Algorithm 3. Synthetic dataset generation

Input: A Datasets: D;
N: Desired number of tuples;
P: Maximal percentage of per-value change;
Output: A synthetic dataset D_{NP} generated from D.
Method:

1. For each attribute A_i, generate a random number $P_i \in$ [0,P];
   P_i is to be the maximal percentage of change for each attribute A_i;
2. For each tuple t in D
3. Let t_1, t_2,...,t_M be multiple copies of t, where M=$\lceil \frac{N}{|D|} \rceil$;
4. Let t_j be the tuple such that $t_j[A_i] = t[A_i] \cdot (1 + q_{ij})$
   for each attribute A_i, where $q_{ij} \in [-P_i, P_i]$ is a random number;
5. Add t_1, t_2, ..., t_M to $D_{\mathbf{NP}}$.

rithm 3.

Figure 6.1 shows how the execution time of SDT-Miner is affected by the number of tuples. Five pairs of synthetic datasets, generated from (LM: PC), are used. All the datasets have 5317 attributes, each dataset pair contains 100, 500, 1000, 1500, 2000 tuples, respectively. Figure 6.1 shows that the execution time increases as the number of tuples increases.

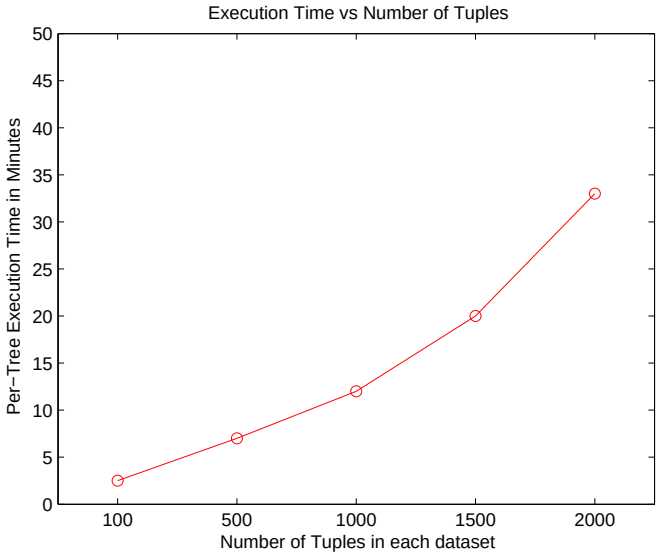

Figure 6.1: Execution time vs number of tuples

6.3 Quality Performance on Real Datasets

We now report experimental results on the quality of the shared tree found by our *SDT-Miner* from the microarray dataset pairs listed in Table 6.2.

6.3.1 Quality of Shared Decision Tree Mined by SDT-Miner

SDT-Miner can be used to mine the shared decision tree. We now show qualities of shared decision trees it mined.

Table 6.3 lists the quality of the shared decision tree mined by *SDT-Miner* for different dataset pairs. The first column is the name of dataset pairs; the second column is the data distribution similarity, (denoted by DS); the third column is the shared decision tree accuracy, (denoted by TA); and the last column is the weighted harmonic mean based quality of the shared decision tree, (denoted by quality of tree). From this table, it is observed that the qualities of decision trees shared by different dataset pairs range from 0.58 to 0.99; the variance of the qualities is 0.03 and the standard deviation is 0.16.

Table 6.3: Quality of tree mined by SDT-Miner

Dataset pairs	DS	TA	Quality of tree
BC vs CN	0.95	0.87	0.91
BC vs DH	0.62	0.69	0.65
BC vs LB	0.97	0.92	0.94
BC vs LM	0.52	0.73	0.61
BC vs PC	0.74	0.73	0.73
CN vs DH	0.88	0.79	0.83
CN vs PC	0.48	0.73	0.58
DH vs LB	0.99	0.95	0.97
LB vs PC	0.99	1	0.99
LM vs PC	0.50	0.90	0.64

6.3.2 Shared Decision Tree Mined from Different Dataset Pairs

Table 6.3 lists the qualities of the shared decision trees mined by *SDT-Miner* for different dataset pairs. After observing the overall qualities of decision trees shared by dataset pairs, we drawn the detailed shared decision trees mined from each specific dataset pair, whose description is also included in this section.

To better understand the shared decision trees, we first give the description of each dataset.

Dataset *BC* is a "Breast Cancer" dataset first published in [12]; the "relapse" class (denoted by C_1 in trees) is tissues from patients who had developed distance metastases within 5 years, and the "non-relapse" class (C_2) is tissues from patients who remained healthy for at least 5 years from the disease after their initial diagnosis.

Dataset *CN* is a "Central Nervous System Embryonal Tumour" dataset first published in [13]; this dataset is about patient treatment outcome prediction. Survivors (C_1) are patients who are alive after treatment whiles the failures (C_2) are those who succumbed to their disease.

Dataset *DH* is a "Diffuse Large B-Cell Lymphoma-Harvard" dataset first published in [14]; this dataset is to predict the patient outcome of DLBCL. The "cured" class (C_1) is tissues from cured patients, and the "fatal" class (C_2) is tissues from patients with fatal or refractory disease.

Dataset *LB* is a "LungCancer-Brigham And Women Hospital-Harvard Medical School" dataset first published in [15]; this dataset is to classify between malignant pleural mesothelioma (MPM) (C_1) and adenocarcinoma (ADCA) (C_2) of the lung.

Dataset *LM* is a "LungCancer-Michigan" dataset first published in [16]. The "tumor" class (C_1) is tissues form lung adenocarcinomas patients, and the "normal" class (C_2) is tissues from non-neoplastic lung patients.

Dataset *PC* is a "Prostate Cancer" dataset first published in [17]; this dataset is to predict clinical outcome. The "relapse" class (C_1) is tissues from patients having recurrence

following surgery, and the "non-relapse" class (C_2) is tissues from patients having remained relapse free for at least 4 years.

Now we turn to present the detailed shared decision trees.

For each tree, the string inside each internal node is the name of the splitting gene. For each node V, the figure includes the associated class distributions, namely [c_{11}, c_{21}] (shown on the left of the node) and [c_{12}, c_{22}] (shown on the right), where c_{i1} is the number of tuples from the first dataset in class C_i that satisfy all of the conditions on the path from the root to V, and similarly c_{i2} is that number for the second dataset. For each branch, the figure includes the splitting condition, denoted by ($\leq a_V$) or ($> a_V$). The figure also includes the data distribution similarity of V, $DSN(V)$, shown below the node.

Figure 6.2 to Figure 6.11 give the shared decision trees, mined by *SDT-Miner* from the dataset pairs (BC: CN), (BC: DH), (BC: LB), (BC: LM), (BC: PC), (CN: DH), (CN: PC), (DH: LB), (LB: PC), and (LM: PC), respectively. For dataset pairs (BC: CN), (BC: LB), (CN: DH), (DH: LB), and (LB: PC), high quality shared decision trees are mined by *SDT-Miner*. On the other hand, we didn't find high quality shared decision trees from remaining dataset pairs.

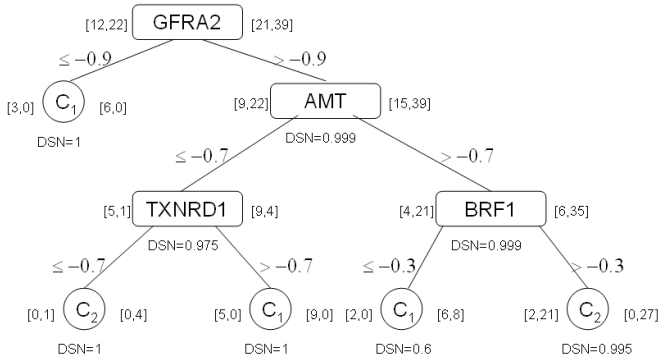

Figure 6.2: Shared decision tree mined from (BC:CN)

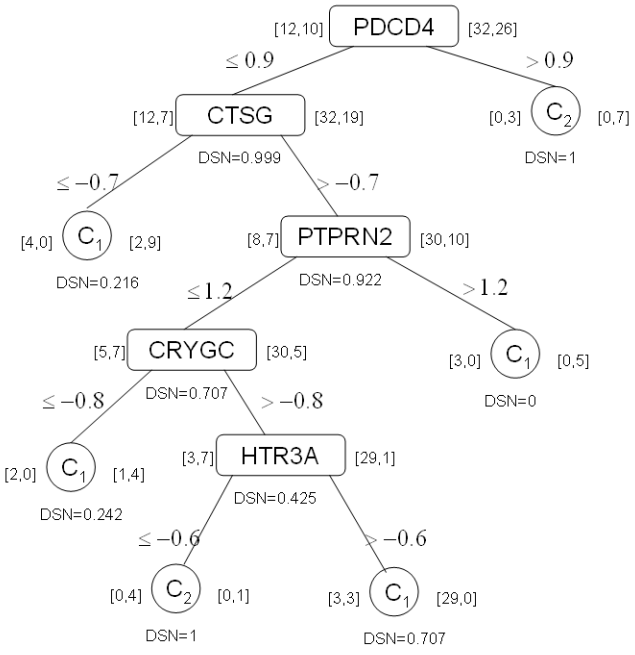

Figure 6.3: Shared decision tree mined from (BC:DH)

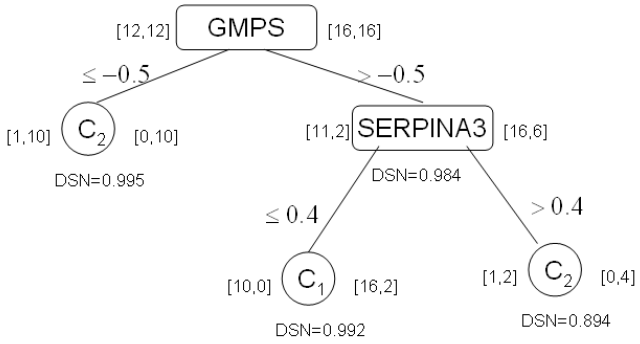

Figure 6.4: Shared decision tree mined from (BC:LB)

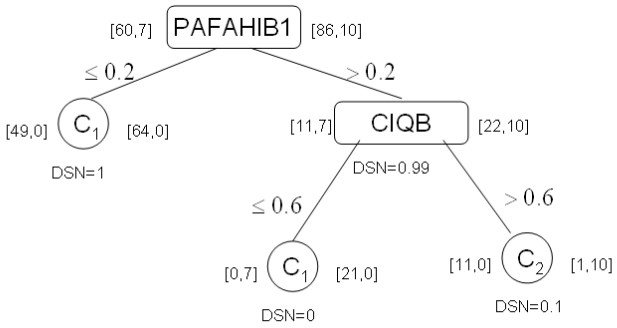

Figure 6.5: Shared decision tree mined from (BC:LM)

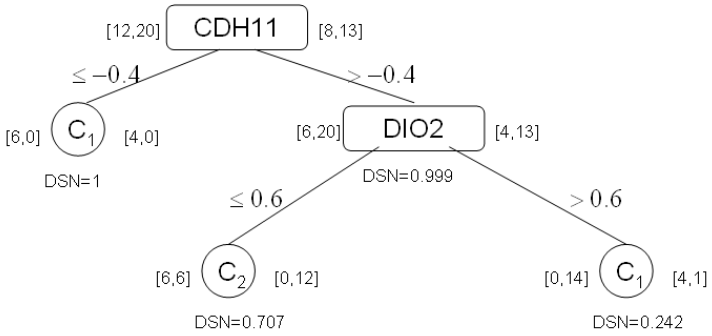

Figure 6.6: Shared decision tree mined from (BC:PC)

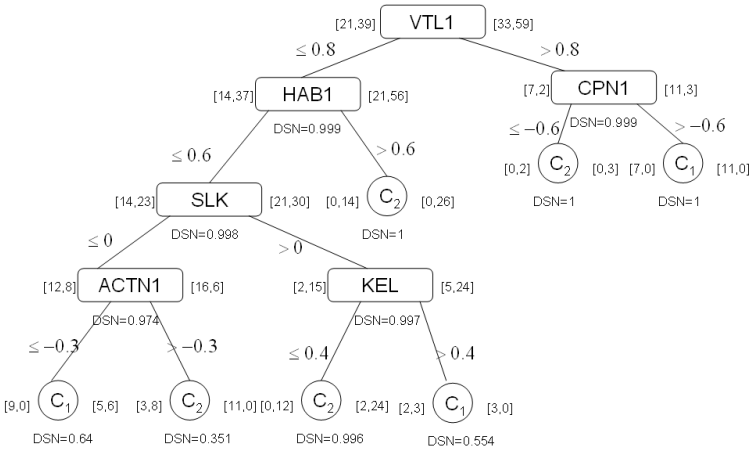

Figure 6.7: Shared decision tree mined from (CN:DH)

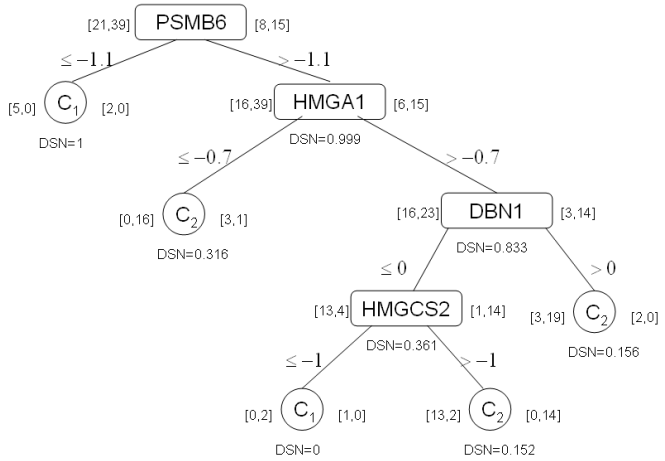

Figure 6.8: Shared decision tree mined from (CN:PC)

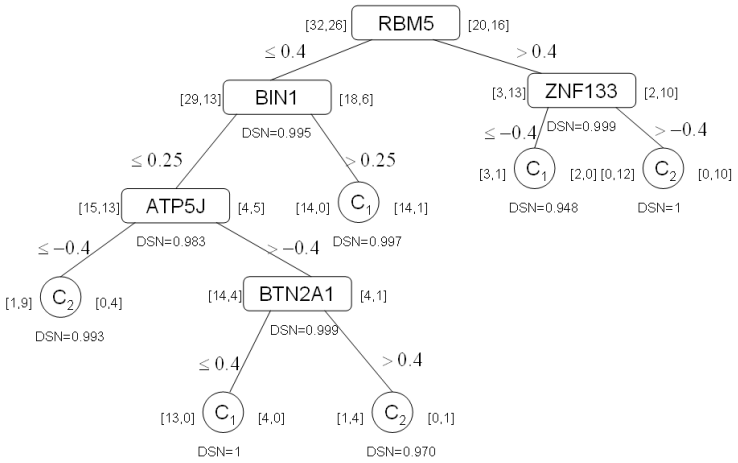

Figure 6.9: Shared decision tree mined from (DH:LB)

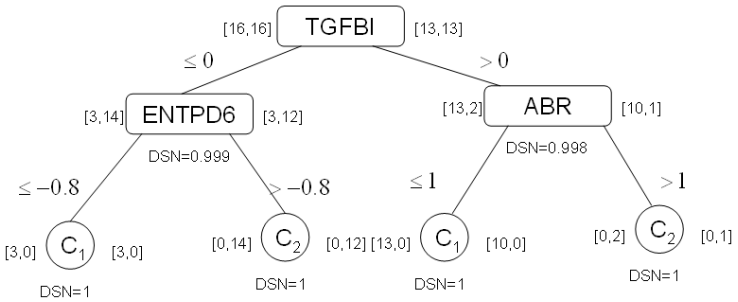

Figure 6.10: Shared decision tree mined from (LB:PC)

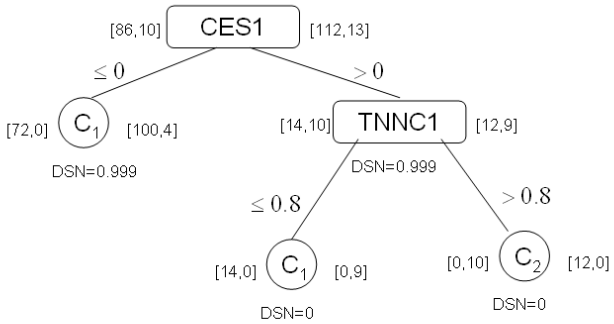

Figure 6.11: Shared decision tree mined from (LM:PC)

Discussion

7.1 Existence of High Quality Shared Decision Tree

The table 6.3 illustrates the qualities of shared decision trees mined from different dataset pairs with the range from 0.58 to 0.99. From our experiments, it is evident that for the dataset pair (CN:PC), the SDT-Miner mined a shared decision tree with only 0.58 tree quality, which means there is few similarity catched between these two datasets. However, when building the decision tree shared by dataset pair (LB:PC), SDT-Miner mined much higher quality tree with the tree quality 0.99, indicating high similarity between datasets LB and PC. This specific high quality shared tree is shown in Figure 6.10.

Therefore, it is obvious that the proposed miner can distinguish different degree of similarities according to tree qualities. High quality shared tree doesn't exist between dataset pair without much shared behavior.

7.2 Class Pairing

For each dataset pair D_1 and D_2, we have two ways to match two classes C_{11}, C_{21} in D_1 and two classes C_{12}, C_{22} in D_2. The first class pairing method is that C_{11} matches C_{12}, C_{21} matches C_{22}, the second class pairing method is that C_{11} matches C_{22}, C_{21} matches C_{12}.

In experimental evaluation section, the experimental results on the quality of the

shared tree found by our *SDT-Miner* from the microarray dataset pairs are based on the first class pairing method.

In this section we select the second class pairing method to match the classes between two datasets. Then we mined shared decision trees using our *SDT-Miner*. Table 7.1 lists the quality of the shared decision tree mined by *SDT-Miner* for the second class pairing method. The first column is the name of dataset pairs; the second column is the data distribution similarity, (denoted by DS); the third column is the shared decision tree accuracy, (denoted by TA); and the last column is the weighted harmonic mean based quality of the shared decision tree, (denoted by quality of tree). From this table, it is observed that the qualities of decision trees shared by different dataset pairs range from 0.50 to 0.98; the variance of the qualities is 0.03 and the standard deviation is 0.18.

Table 7.1: Quality of tree mined by SDT-Miner

Dataset pairs	DS	TA	Quality of tree
BC vs CN	0.77	0.55	0.64
BC vs DH	0.97	0.90	0.93
BC vs LB	0.99	0.92	0.96
BC vs LM	0.51	0.75	0.61
BC vs PC	0.42	0.62	0.50
CN vs DH	0.82	0.69	0.75
CN vs PC	0.96	0.90	0.93
DH vs LB	0.55	0.67	0.60
LB vs PC	0.99	0.96	0.98
LM vs PC	0.48	0.82	0.61

7.3 Looking into Attributes Used by Trees

We have mined shared decision trees for both class pairings. For each dataset pair and for each class pairing, we find the set of attributes used in the shared decision tree.

Table 7.2 to Table 7.11 list the sets of attributes used in shared decision trees, mined

by *SDT-Miner* from the dataset pairs (BC: CN), (BC: DH), (BC: LB), (BC: LM), (BC: PC), (CN: DH), (CN: PC), (DH: LB), (LB: PC), and (LM: PC), respectively.

Table 7.2: Attributes used by trees from (BC:CN)

First Class Pairing	Second Class Pairing
GFRA2	NAGA
AMT	KEL
BRF1	CFTR
TXNRD1	

Table 7.3: Attributes used by trees from (BC:DH)

First Class Pairing	Second Class Pairing
PDCD4	SLC25A13
CTSG	CTSG
PTPRN2	GATA3
CRYGC	CPT1A
HTR3A	

Table 7.4: Attributes used by trees from (BC:LB)

First Class Pairing	Second Class Pairing
GMPS	LSR
SERPINA3	IQGAP1

Then we introduce a concept, namely the fraction of attributes (F_A). It is used to determine the fraction of attributes shared by the trees mined from two class pairings over attributes used by the tree mined from either class pairing. In formula, we have:

$$F_A = \frac{N_{common}}{N_{either}}, \tag{7.1}$$

where N_{common} is the number of attributes shared by the trees mined from the two class pairings, and N_{either} is the set of attributes used by the tree mined from either class pairing.

Table 7.5: Attributes used by trees from (BC:LM)

First Class Pairing	Second Class Pairing
PAFAH1B1	CHN1
C1QB	FOXF1

Table 7.6: Attributes used by trees from (BC:PC)

First Class Pairing	Second Class Pairing
CDH11	IFI6
DIO2	PRY
	DFFA

Now we discuss the F_A of all dataset pairs. The following table 7.12 lists the N_{common}, N_{either} and F_A for each dataset pair and each class pairing. The N_{common} and N_{either} are obtained from the shared decision trees mined by *SDT-Miner*. And we also list the set of attributes used in the shared decision trees for both class pairings, mined by *SDT-Miner*.

Table 7.7: Attributes used by trees from (CN: DH)

First Class Pairing	Second Class Pairing
VIL1	IGF2
CPN1	RFC4
HAB1	PMPCA
SLK	SLK
KEL	DNAH3
ACTN1	ABCC8

Table 7.8: Attributes used by trees from (CN: PC)

First Class Pairing	Second Class Pairing
PSMB6	FBN1
HMGA1	CFD
DBN1	PTPRO
HMGCS2	SHC1

Table 7.9: Attributes used by trees from (DH: LB)

First Class Pairing	Second Class Pairing
RBM5	DBP
ZNF133	ATP5J
BIN1	CAPN2
ATP5J	
BTN2A1	

Table 7.10: Attributes used by trees from (LB: PC)

First Class Pairing	Second Class Pairing
TGFBI	SMARCC2
ABR	SLC35E2
ENTPD6	ATP5B

Table 7.11: Attributes used by trees from (LM: PC)

First Class Pairing	Second Class Pairing
CES1	TRAF4
TNNC1	CAV1

Table 7.12: F_A of dataset pairs

Dataset pairs	N_{common}	N_{either}	F_A
BC vs CN	0	7	11.1%
BC vs DH	1	9	0%
BC vs LB	0	4	0%
BC vs LM	0	4	0%
BC vs PC	0	5	0%
CN vs DH	1	12	8.3%
CN vs PC	0	8	0%
DH vs LB	0	8	0%
LB vs PC	0	6	0%
LM vs PC	0	4	0%

Conclusion and Future Work

This thesis addressed the shared models, patterns and structures mining problem and proposed an algorithm, namely SDT-Miner, for mining a high quality shared decision tree. The effectiveness of the algorithm is verified by experiments on synthetic and real world datasets.

In the future, there are several areas in which we can improve our work.

Firstly, both the shared decision tree mining problem and *SDT-Miner* can be generalized to three or more datasets. The experiment results demonstrate that our *SDT-Miner* could mine a high quality decision tree shared by two datasets. In practice, our proposed method could also be used to mine a high quality decision tree shared by three or more datasets. To apply our method for multiple datasets, we need to find equivalent attributes among multiple datasets. Furthermore, we need to change the definition of distribution similarity (DSN) at a node V and the tree accuracy (TA) for multiple datasets. Then we could employ same *SDT-Miner* algorithm to mine a high quality decision tree shared by multiple datasets.

Secondly, It is not enough to mine only one shared decision tree. Indeed, one shared decision tree may give only a limited view on the pattern similarities shared by the given datasets. In order to give users multiple diversified view on the pattern similarities shared by the datasets, we will extend our method to mine a small set of multiple diversified high quality decision trees. The tree set satisfies these properties: (a) each tree in the set (1) has highly similar data distributions and (2) has high classification accuracy in the datasets, and

(b) different trees in the set are highly different from each other.

In the end, we will try to develop new methods to mine other forms of shared models, including correlation/association patterns, graph-like interaction patterns, hidden Markov models, clusterings, and so on.

Bibliography

[1] M. Klenk and K. Forbus, "Domain transfer via cross-domain analogy," *Cognitive Systems Research*, pp. 240–250, 2009.

[2] D. N. Perkins, G. Salomon, and P. Press, "Transfer of learning," *International Encyclopedia of Education*, 1992.

[3] U. Zander and B. Kogut, "Knowledge and the speed of the transfer and limitation of organizational capabilities: An empirical test," *Organization Science*, vol. 6, pp. 76–92, 1995.

[4] R. J. Evaristo, "Knowledge transfer across borders: a process model," *Knowledge and Process Management*, vol. 14, pp. 203–210, 2007.

[5] Z. Su, P. Dam, X. Chen, V. Olman, T. Jiang, B. Palenik, and Y. Xu, "Computational inference of regulatory pathways in microbes," *Workshop on Genome Informatics*, vol. 14, pp. 631–633, 2003.

[6] V. Olman, H. Peng, Z. Su, and Y. Xu, "Mapping of microbial pathways through constrained mapping of orthologous genes," *Systems Bioinformatics Conference*, pp. 363–370, 2004.

[7] L. McCue and W. Thompson, "Factors influencing the identification of transcription factor binding sites by cross-species comparison," *Genome research*, 2002.

[8] W. Dai, G. Xue, Q. Yang, and Y. Yu, "Transferring naive bayes classifiers for text classification," *The 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pp. 540–545, 2007.

[9] J. Gao, W. Fan, J. Jiang, and J. Han, "Knowledge transfer via multiple model local structure mapping," *Proc. 2008 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'08)*, Aug. 2008.

[10] S. Mao, C. Wang, and G. Dong, "Evaluation of inter-laboratory and cross-platform concordance of dna microarrays through discriminating genes and classifier transferability," *Bioinformatics and Computational Biology*, vol. 7, pp. 157–173, 2009.

[11] W. Tong, X. Cao, S. Harris, H. Sun, H. Fang, J. Fuscoe, A. Harris, H. Hong, Q. Xie, R. Perkins, L. Shi, and D. Casciano, "Arraytrack - supporting toxicogenomic research at the fda's national center for toxicological research (nctr)," *EHP Toxicogenomics*, vol. 111, pp. 1819–1826, 2003.

[12] L. J. van't Veer, "Gene expression profiling predicts clinical outcome of breast cancer," *Nature*, vol. 415, pp. 530–536, 2002.

[13] S. L. Pomeroy, "Prediction of central nervous system embryonal tumour outcome based on gene expression," *Nature*, vol. 415, pp. 436–442, Jan. 2002.

[14] M. A. Shipp, "Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning," *Nature Medicine*, vol. 8, pp. 68–74, Jan. 2002.

[15] G. J. Gordon, "Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma," *Cancer Research*, vol. 62, pp. 4963–4967, 2002.

[16] D. G. Beer, S. L. Kardia, C. ching Huang, T. J. Giordano, and A. M. Levin, "Gene-expression profiles predict survival of patients with lung adenocarcinoma," *Nature Medicine*, vol. 8, pp. 816–823, Aug. 2002.

[17] D. Singh, "Gene expression correlates of clinical prostate cancer behavior," *Cancer Cell*, vol. 1, pp. 203–209, Mar. 2002.