A Tree-based Framework for Difference Summarization

Ruoming Jin, Yuri Breitbart, Rong Li
Department of Computer Science
Kent State University
Kent, OH, 44242, USA
Email: {jin,yuri,rli0}@cs.kent.edu

***Abstract*—Understanding the differences between two datasets is a fundamental data mining question and is also ubiquitously important across many real world scientific applications. In this paper, we propose a tree-based framework to provide a parsimonious explanation of the difference between two distributions based on rigorous two-sample statistical test. We develop two efficient approaches. The first one is a dynamic programming approach that finds a minimal number of data subsets that describe the difference between two data sets. The second one is a greedy approach that approximates the dynamic programming approach . We employ the well-known Friedman's MST (minimal spanning tree) statistics for two-sample statistical tests in our summarization tree construction, and develop novel techniques to speedup its computational procedure. We performed a detailed experimental evaluation on both real and synthetic datasets and demonstrated the effectiveness of our tree-summarization approach.**

***Keywords*-difference summarization; minimal spanning tree; two sample test;**

I. Introduction

Understanding the differences between two datasets is a fundamental data mining problem and it is also ubiquitously important across many real world applications and scientific disciplines. In pharmaceutical study, doctors would like to learn patients responses from two competitive or related drugs. Specifically, if the responses are statistically different, doctors need to understand the differences with respect to a list of different factors, such as blood pressure,age, etc.. Frequently, they may find that in most of the parameter ranges or conditions, the responses are quite similar. Thus, the key problem is how can they find a concise description to summarize differences and/or similarities between the responses to these two drugs? Similar problems can also appear frequently in the business domain. For instance, the overall sale of the December of 2008 is quite different from the overall sale of the December of 2007. However, the important problem is to learn where customers actually spend their money, i.e., we need to understand how customers spend their money towards different products at two different time points.

Despite the importance of this problem, little research in data mining has been done to derive a summarization framework to concisely describe the differences between two datasets. Contrast mining is a closely related effort [1], [2], [3], which tries to discover the significant differences between two datasets. It can discover some local differences between two distributions, but it does not try to provide a parsimonious explanation for the difference between two distributions. Another work in a spirit of this problem is to provide a parsimonious explanation of change for OLAP applications [4], [5]. Given a hierarchical framework, they try to summarize the difference of aggregations. However, their work is quite limited since they focus on the aggregation where the hierarchical structure is already given and their approach cannot be applied for the generalized distribution setting.

In this paper, we provide a summarization framework to describe differences between two data sets, each of which consists of points in a multidimensional space. Intuitively, suppose that we were given two sets of points: one set of black and the other set consists of red points, distributed in a multidimensional space. Contrary to being well-separated, these two sets of points seem intertwined with one another. Suppose that despite their similarity, using the well-known *two-sample test* in statistical data analysis [6], we cannot accept the hypothesis that these two datasets are being generated from the same distribution. (For convenience, we say they are statistically different.) Thus, the research problem here is *how can we concisely summarize or explain the differences between these two sets of points?*

We propose here a tree-based framework to provide a parsimonious explanation of the differences between two data sets. At the high level, the tree summarization framework shares certain similarity with the well-known decision-tree scheme. Basically, from the root, each internal node of the tree corresponds to a cut on a specific dimension, which will recursively split the multidimensional space into two parts. However, in contrast to the decision-tree scheme, our goal is not to try to separate the two sets of data points. Instead, the tree illustrates a parsimonious description on where the distributions differ. Specifically,

1. We provide the study on the problem of constructing a concise summarization of the differences between two distributions/datasets and formulate a tree-summarization framework based on rigorous two-sample statistical test.

2. We develop two efficient approaches to construct the summarization tree. The first one is a dynamic programming approach. The other one is a greedy approach.

3. In our summarization tree construction, we employ

the well-known Friedman's MST (minimal spanning tree) statistics for two-sample statistical tests [7], and develop novel techniques to speedup the computation procedure.

4. We perform a detailed experimental evaluation on both real and synthetic datasets and demonstrate the effectiveness and efficiency of our tree-summarization approach.

II. Preliminary: Multivariate Two-Sample Test

In this section, we will first describe the *two-sample test* [6], which forms the basis of our problem formulation and our approach to its solution.

A. Two-Sample Test

Consider two sets of data points D_1=$\{X_1, X_2, \cdots, X_n\}$ and $D_2 = \{Y_1, Y_2, \cdots, Y_m\}$ in d-dimensional space, where m and n are the size of datasets D_1 and D_2, respectively. We interested in determining if they are likely to be generated from the same underlying distribution. This question is addressed in the classical *two-sample* test. Let X_i and Y_j be independent samples from unknown distributions $F(x)$ and $G(x)$, respectively. The two-sample test considers two hypotheses: a null hypothesis H_0: $F(x) = G(x)$ against the alternative hypothesis H_1: $F(x) \neq G(x)$.

A good test statistic is expected to satisfy the *distribution-independent* and *consistent* requirements [8], [9], [10], [11]. An exact distribution-independent test requires that under the null hypothesis, if $min(m, n) \rightarrow \infty$, the test statistic does not depend on the unknown distribution $F(x)$ (asymptotically distribution free), and the limiting distribution is known. The test also needs to be consistent against the alternative H_1, i.e., when $min(m, n) \rightarrow \infty$, the probability of rejecting the null hypothesis converges to one. In other words, when $F(x) \neq G(x)$, we will surely reject the null hypothesis as the number of samples approaches infinity. In addition, we note that when we reject the null hypothesis using any statistical test, we in fact prove that the distributions are different, and thus, we can say these two datasets are *statistically different*. However, in statistics, one can never prove the distributions to be the same even with a large amount of data. Thus, in this paper, when we are not be able to reject the null hypothesis, we just say they are "*statistically consistent*" for simplicity.

There are a lot of two-sample tests being developed over the last several decades [6], [12], [11]. The majority of them are on one-dimension two sample test. To generalize them to multidimensional sets is not trivial. Also, the well-known chi-square test is for *binned distribution* [8]. However, it is not a *consistent* test for continuous space, i.e., the chi-square test will not be able to differ two datasets if they indeed come from different distributions when the sample size approximates infinity.

B. Friedman-Rafsky MultiVariate Two-Sample Test

Several methods [7], [9], [10], [13] have been developed for the two sample test in multidimensional space or simply for the multivariate two sample test. Among them, the Friedman-Rafsky test[7] is one of the most well-known and computationally efficient multivariate two sample test. The basic idea of this approach is to first construct a minimum spanning tree (MST) for all the data points in $D_1 \cup D_2$ in the multi-dimensional Euclidean Space. Then, to remove all the edges in the MST whose two adjacent nodes (data points) come from different datasets, i.e., one from D_1 and another from D_2. Thus, the MST becomes a forest with many disjoint subtrees. In particular, all the subtrees (also referred to as $runs$) in the forest contain the same type of nodes (data points from the same dataset). Finally, the test statistic $R_{m,n}$ is the total number of those disjoint subtrees in the resulting forest.

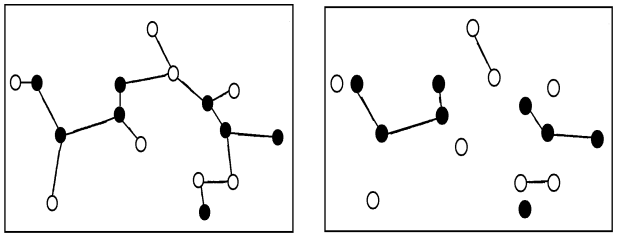


Figure 1. (a) global MST and (b) the runs of MST

These concepts are illustrated in Figures 1. Figure 1(a) shows the global MST of two samples of 8 points (each point in one sample is assigned with same labels), while Figure 1(b) shows the result after removing edges that liking nodes from different samples. Note that a run is a disjoint subtree in the forest whose data points are all from the same data set. Intuitively, a rejection of H_0 corresponds to the case when we have a relatively small number of subtrees (runs). In other words, the two sets of points are relatively well-separated.

Formally, it has been proven that for large sample size ($min(m, n) \rightarrow \infty$ and $m/(m+n) \rightarrow p \neq 0$, under the H_0, the distribution of

$$W(R_{m,n}) = \left(R_{m,n} - (1 + \frac{2mn}{m+n}) \right) / ((m+n)^{1/2} \sigma_d) \quad (1)$$

approaches the standard normal distribution, where $\sigma_d^2 = r(r + \frac{1}{2} Var(D_d)(1 - 2r)$ ($r = 2p(1-p)$ and $Var(D_d)$ is the variance of the degree of any vertex in the MST and can be easily approximated [14]). Given this, for a significance level α, we can reject the null hypothesis if $W < z_\alpha$ where $P(x \leq z_\alpha) = \alpha$, z_α is the critical value for the normal distribution. Using this procedure, we can expect that all the internal nodes are statistically different and most of the leaf nodes are statistically consistent (cannot reject the null hypothesis) (or $min(m, n) < t$, where t is pre-defined threshold).

Friedman-Rafsky's MST statistic test is a clever extension of the univariate Wald-Wolfowitz test[11] (just imagine the tree is projected into one dimension space and becomes a line and a run is simply a set of consecutive points all

from the same datasets). It has been proven that Friedman-Rafsky's MST statistical test is distribution-independent and consistent [14], [7]. It is also a rather computational efficient test procedure since several methods can construct the minimal spanning tree in the Euclidean space in almost $O(N \log N)$ computational complexity, where $N = m + n$ is the total number of data points [15], [16], [17].

We note that the multivariate two sample test is the basis of our summarization framework to describe the statistical difference between two datasets. As we will see, our tree-based summarization approach can in general utilize any available multivariate two-sample tests. But most of them do not have the nice properties of being distribution independent, consistent, and computationally efficient as the Friedman-Rafsky's MST statistic test. Thus, in this work, we focus on the Friedman-Rafsky's MST statistic test method to describe statistical differences between two datasets. Since our summarization-tree will repeatedly invoke the test procedure, we will also develop novel methods to further speedup its computation (Section V).

III. The Summarization Framework: A Tree-based Scheme

In this section, we introduce our tree-based framework to provide a parsimonious description of the differences between two datasets.

Summarization Tree: A summarization tree at the high level is a binary tree which recursively partitions the multidimensional space into smaller regions. Each node of the tree corresponds to a region in the multidimensional space, and all the data points in both datasets D_1 and D_2 that belong to that region. The root has the entire multidimensional space (R^d). The region associated with each node is described recursively through a cut associated with each internal node. Specifically, the cut of each internal node v_i is denoted as $[A_i : x_i]$, where A_i is the cut attribute and x_i is the cut value. This cut will split the region and all the data points in that region into two parts and each part belong to one of the internal node's two children: the left child has all the data points of its parent whose A_i dimension is less than or equal to the x_i ($A_i \leq x_i$) and the right child has those data points whose A_i dimension is bigger than x_i ($A_i > x_i$). Finally, the leaf nodes do not have any cut.

We use this tree structure to describe the differences between any two datasets D_1 and D_2. The tree has the following properties:

1) **Internal Nodes:** Let $D_1[v_i]$ and $D_2[v_i]$ be the sets of data points belonging to the region described by interval node v_i. For any internal node v_i, we can reject the null hypothesis that $D_1[v_i]$ and $D_2[v_i]$ are generated from the same distribution. In other words, the associated two sets of data points of any internal node from D_1 and D_2 are statistically different.
2) **Leaf Nodes:** The sets of data points associated with leaf node v_i, $D_1[v_i]$ and $D_2[v_i]$, either are statistically consistent, or too small to be further partitioned. In the latter case, a lower bound on the minimal number of data points in each set is defined to make the statistical test meaningful.
   At each node v_i, we need to run a statistical test to see if its associated sets of data points, $D_1[v_i]$ and $D_2[v_i]$ statistically come from the same distribution or not. In other words, we have a null hypothesis $F_{D_1[v_i]}(x)$=$G_{D_2[v_i]}(x)$ against the alternative $F_{D_1[v_i]}(x) \neq G_{D_2[v_i]}(x)$. We apply the aforementioned Friedman-Rafsky's MST two sample test of Subsection II-B for this purpose.
3) **Cut:** Each cut is associated with the internal node v_i describes a *summarized view* of the associated datasets, $D_1[v_i]$ and $D_2[v_i]$. Specifically, for each internal node v_i, its cut $[A_i : x_i]$ splits its associated data, $D_1[v_i]$ and $D_2[v_i]$ into two parts: let v_j be its left child and v_k be its right child. Then, we have $D_1[v_j]$ and $D_2[v_j]$ for the left child and $D_1[v_k]$ and $D_2[v_k]$ for the right child. To understand the summarized view of its left child and right child , we assume that the cut creates two bins for the associated data, $D_1[v_i]$ and $D_2[v_i]$. In other words, we have a 2×2 contingency table, where each row corresponds to a dataset, and the columns correspond to the bins.

	Left Child (v_j): $A_i \leq x_i$	Right Child (v_k): $A_i > x_i$
D_1	$\|D_1[v_j]\|$	$\|D_1[v_k]\|$
D_2	$\|D_2[v_j]\|$	$\|D_2[v_k]\|$

Table I
Statistical Test for the Summarized View

   We can use the chi-square test to determine whether the quantities in the first row and the quantities in the second row come from the same distribution [8]. If we can reject that they come from the same distribution, we say the cut is a dependent cut. Otherwise, we say the cut is an independent cut.

The main intuition and/or motivation of this framework is based on the observation that two related datasets/distributions often tend to be the similar in most parts of space. However, they differ either because there is a shift of data distribution from one part of the space to another part of the space, or there is a hot spot or area in the space. Those are likely to be the events resulting into the differences between two datasets. For instance, considering in the business example, the sale of this December is close to the sale of the last December because the customers in total spend less money on the luxury products, such as DVD or games, but their purchase distribution is still similar if we exclude such high level difference. Indeed, the decision tree construction allows us to focus on the local regions and when we do the two-sample tests, the global effect of the difference is isolated and does not affect the two-sample test on the local regions. Specifically, the two sample test on the $D_1[v_i]$ and $D_2[v_i]$ is totally independent from the the rest of data points, i.e., $D_1 \backslash D_1[v_i]$ and $D_2 \backslash D_2[v_i]$.

We also note that in the traditional statistical analysis, the global distribution difference is typically captured through the mean-shift or scaling.

Here, we assume that both distributions are properly normalized and thus we do not need to handle such difference explicitly. Besides, in this framework, we do not consider the multiple comparison/inference problem [6] as we treat each two-sample test independently, and we are not interested in deriving an overall statistical significance for the entire summarization-tree. However, if this is preferred, i.e., we would like to treat all the two-sample tests in a summarization-tree as a whole, then we can utilize the methods, such as Bonferroni correction, to adjust the significance level of each test.

Minimal Summarization Tree Problem: Given this, we introduce the *minimal difference summarization tree (MDST) problem*. Let T be a difference summarization tree. We assign each node in the tree a cost. The main idea is that if we cannot reject two distributions are same (statistically consistent), we do not need to output the description of the two distributions. For other cases, we need to output certain information to describe their differences. Those information will be associated with each node in the tree and the costs of each node reflect the minimal description length for such information.

For each internal node v_i, if the cut is dependent, the cost of the node $cost(v_i) = 6$; if the cut is independent, the cost of the node $cost(v_i) = 2$. This is because if it is independent cut, we only need to record two values: the cut point and the dimension, while if it is dependent cut, we need to record six values: the cut point, the dimension, and the summarized distribution for chi square test. For each leaf node v_i, if we cannot reject that its associated two datasets $D_1[v_i]$ and $D_2[v_i]$ come from the same distribution, its cost $cost(v_i) = 0$, and otherwise, $cost(v_i) = number\ of\ points\ in\ the\ leaf\ node$. However, there are some cases need to be considered carefully. One is that if in the leaf node, the number of points from one dataset is too small, let us say 5, we assign the leaf node $cost(v_i) = number\ of\ points\ in\ the\ leaf\ node$. Another case is that if the number of points from one dataset is just a very small proportion of the number of points from another dataset (e.g. 5%), then applying the statistics test has no meaning, too. Thus we assign the leaf node $cost(v_i) = number\ of\ points\ from\ smaller\ dataset\ in\ the\ leaf\ node$. Given this, the overall description for tree T is simply the sum of all the costs from its nodes, $Cost(T) = \sum_{v_i \in T} cost(v_i)$. (The cost of each node can be modified according to different applications.)

The example of calculating the cost and constructing the tree is given in the experiment results section. See Figure.3 and Figure.4.

Definition 1: ***(Minimal Difference Summarization Tree (MDST) Problem)*** *Given two datasets D_1 and D_2, the most parsimonious description of the differences between two datasets D_1 and D_2 is the difference summarization tree T which has the minimal cost, i.e., the minimal difference summarization tree:*

$$\arg\min_T Cost(T)$$

IV. Algorithms for Tree Construction

In this section, we introduce two approaches to find the minimal difference summarization tree ($MDST$) for two datasets D_1 and D_2. The first approach is a dynamic programming approach, which can guarantee to find MDST but is computationally prohibitive. The second approach is a greedy approach.

In both dynamic programming approach and greedy approach, we will prune the subtree if its total cost is higher than the root node cost (without the subtree), and replace the subtree with a leaf node to indicate the corresponding region is statistically different.

Finally, we note that in both approaches, we need to invoke the Friedman-Rafsky two sample test. Since it needs to repetitively construct minimal spanning tree on each specific region, this becomes very expensive. In the next section, we will introduce two novel techniques which can significantly reduce such a cost.

A. Dynamic Programming for MDST

Different from the classical decision tree construction [18], we will first show MDST problem can be solved by dynamic programming in polynomial time. Then, we introduce heuristics to reduce the computational cost for dynamic programming by discretization.

To facilitate our discussion, we introduce the following notation. Let $[b_1, e_1] : [b_2, e_2] : \cdots : [b_d, e_d]$ be a d-dimensional cube in R^d, and let $D_1[[b_1, e_1] : \cdots : [b_d, e_d]]$ and $D_2[[b_1, e_1] : \cdots : [b_d, e_d]]$ be two sets of points associated with the region from D_1 and D_2, respectively. Clearly, each node in the tree is associated with such a cube and its corresponding datasets. The possible cutting point is simply the median point between any two consecutive values on each dimension. Specifically, for dimension i, we can project all the data points in $D_1[[b_1, e_1] : \cdots : [b_d, e_d]]$ and $D_2[[b_1, e_1] : \cdots : [b_d, e_d]]$ on that, and suppose in the i-dimension, we have the unique points $y_1 < y_2 < \cdots, y_t$. Then, the cut points are $\frac{y_i + y_{i+1}}{2}$, where $1 \leq i \leq t-1$. Further, let $T[b_1, e_1] : \cdots : [b_d, e_d])$ be a difference summarization tree on the cube $[b_1, e_1] : \cdots : [b_d, e_d]$. Let $y_{i,j}$ denote the j-th cut point for i-th dimension. Then, the basic observation for the MDST is the following recursive formulation:

$$\begin{aligned} & \min_T Cost(T([b_1, e_1] : \cdots : [b_d, e_d]) \\ = \ & \min_{i,j} Cut(y_{i,j} | [b_1, e_1] : \cdots : [b_d, e_d]) + \\ & \min_T Cost(T([b_1, e_1] : \cdots : [b_i, y_{i,j}] : \cdots : [b_d, e_d]) + \\ & \min_T Cost(T([b_1, e_1] : \cdots : [y_{i,j}, e_i] : \cdots : [b_d, e_d]) \end{aligned} \quad (2)$$

Here, the $Cut(y_{i,j}|[b_1, e_1] : \cdots : [b_d, e_d])$ is the cost of the summarization difference on the two datasets (Section III). Using the chi-square test, if the cut is dependent, then, the cut cost is 6, and otherwise, the cost is 2. Finally we note that for the root node, b_i and e_i, is simply the smallest and largest value in the i-th dimension.

Algorithm 1 Dynamic Programming for MDST

Procedure MinTreeCost($[b_1, e_1], \cdots, [b_k, e_k]$)
Require: Datasets D_1 and D_2
1: **if** Find_Cube($[b_1, e_1] : \cdots : [b_k, e_k]$){computed before} **then**
2: **return** Cost($[b_1, e_1] : \cdots : [b_k, e_k]$)
3: **end if**
4: **if** ($|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]|/|D_2[[b_1, e_1] : \cdots : [b_k, e_k]]| < b$) || ($|D_2[[b_1, e_1] : \cdots : [b_k, e_k]]|/|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]| < b$) {b is the lower bound for the statistical tests} **then**
5: Cost($[b_1, e_1] : \cdots : [b_k, e_k]$) $\leftarrow \min(|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]|, |D_2[[b_1, e_1] : \cdots : [b_k, e_k]]|)$ {treat as dependent cut}
6: **return** Cost($[b_1, e_1] : \cdots : [b_k, e_k]$)
7: **end if**
8: **if** $\min(|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]|, |D_2[[b_1, e_1] : \cdots : [b_k, e_k]]|) < t$ {t is the lower bound for the statistical tests} **then**
9: Cost($[b_1, e_1] : \cdots : [b_k, e_k]$) $\leftarrow (|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]| + |D_2[[b_1, e_1] : \cdots : [b_k, e_k]]|)$ {treat as dependent cut}
10: **return** Cost($[b_1, e_1] : \cdots : [b_k, e_k]$)
11: **end if**
12: **if** NOT Test-Diff($D_1[[b_1, e_1] : \cdots : [b_k, e_k]], D_2[[b_1, e_1] : \cdots : [b_k, e_k]]$) {statistically consistent using two sample test} **then**
13: Cost($[b_1, e_1] : \cdots : [b_k, e_k]$) $\leftarrow 0$
14: **return** 0
15: **end if**
16: $MinCost \leftarrow (|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]| + |D_2[[b_1, e_1] : \cdots : [b_k, e_k]]|)$
17: **for all** dimension i, $1 \leq i \leq d$ **do**
18: **for all** cut point $y_{i,j}$ in i-dimension **do**
19: $L \leftarrow [b_1, e_1] : \cdots : [b_i, y_{i,j}] : \cdots : [b_d, e_d]$
20: $R \leftarrow [b_1, e_1] : \cdots : [y_{i,j}, e_i] : \cdots : [b_d, e_d]$
21: $Cut \leftarrow \chi^2 \begin{pmatrix} |D_1[L]| & |D_1[R]| \\ |D_2[L]| & |D_2[R]| \end{pmatrix}$
22: $MinCost \leftarrow \min(MinCost, Cut + MinTreeCost(L) + MinTreeCost(R))$
23: **end for**
24: **end for**
25: Cost($[b_1, e_1] : \cdots : [b_k, e_k]$) $\leftarrow MinCost$
26: **return** $MinCost$

The algorithm description of the dynamic programming is shown in Algorithm 1. For each given cube, this algorithm will first check if the difference summarization tree has been computed before (Line 1) or if it does not have enough data for the two sample statistical tests (Line 4 and Line 8), in the three cases, we directly return its cost. Then, we will perform a two-sample test on the two datasets in the cube (Line 12). If we cannot reject they come from the same distribution, we will assign this cube cost 0 (a possible leaf node), and return the cost immediately. Otherwise, we find the two datasets are statistically different, and then, first we assume that the min cost of this cube is the total number of points in this cube (Line 16), this is used for pruning the subtree. Later we need to try all possible cuts on each dimension (Line 17 and 18), and apply the aforementioned recursive formula to find the minimal cost for the difference summarization tree on this cube (Line 21 and 22). It is easy to see that this approach guarantees to find the minimal difference summarization tree on datasets D_1 and D_2. Its computational cost in the worst case is $O((N^{3d} + N^{2d} \times N^2)$, where N^{3d} is the cost of searching on all possible subproblems and $N^{2d} \times N^2$ is the total cost of invoking the two-sample test, which in the worst case is $O(N^2)$ considering we use the Friedman-Rafsky's MST test ($N = |D_1| + |D_2| = m + n$). It also needs $O(N^{2d})$ to memorize the intermediate computation (all cubes). When N and d are large this procedure is apparently computationally prohibitive.

To make it more computationally feasible, we can try to avoid looking at all possible cut points. Instead, we would like to look at candidates that are more promising to produce a small different summarization tree. In this work, we employ a discretization preprocessing step to procedure a fixed number (K) of intervals at each dimension on the entire datasets, and then we only look at the discretization points as the possible cutting points. This can reduce the computational cost to $O((K^{3d} + K^{2d} \times N^2)$. In addition, we note that many discretization methods have been developed for both supervised and unsupervised learning. Here, we apply the Chi-merge [19] procedure which can merge two statistically independent intervals together and each dataset is used as a class label in the discretization step.

B. Greedy Construction Algorithm

Clearly, even with the discretization, the dynamic programming can be prohibitive at high dimension datasets (d is large). Thus, our second algorithm employs a greedy construction procedure similar to the classical decision tree classifier construction. At each step from the root node, it greedily finds an optimal cut according to certain criteria. Then it splits both datasets into two parts, respectively, and constructs two children based on the cut. Then, it will recursively split each child until we cannot reject that the associated datasets are generated from the same distribution, or not enough sample data points for the two-sample test.

However, the key problem is how to find the optimal cut for a given node and its associated datasets. This problem is clearly different from the classical decision tree construction where they try to split the data into well-separated classes. Typically, either the information entropy or gini index is used to determine the best cut. Here, however, the two datasets are most likely inseparable from one another, and our goal is to provide the concise summarization of their differences. Thus, a key insight is that at each node, we should try to split its corresponding datasets into two parts, where both datasets at each part are more likely to be statistically consistent. In other words, if the two datasets are statistically different globally on the cube, we can find where they differ most

and then select that point as the splitting point.

Our intuition can be formalized by the two-sample Kolmogorov-Smirnov (K-S) test statistic, which is one of the most widely used test for one-dimension two sample test [6]. The K-S test statistic is defined on the top of the empirical distribution functions from datasets D_1 and D_2. For any real value t,

$$F_n(t) = \frac{\text{number of } X_i, X_i \leq t}{n}$$

and

$$G_m(t) = \frac{\text{number of } Y_i, Y_i \leq t}{m}$$

The functions $F_n(t)$ and $G_m(t)$ are the empirical marginal distributions on any dimension of the two sets D_1 and D_2, respectively. Given this, the K-S statistic is defined as the maximum difference between the empirical distributions:

$$D = \max_{-\infty<t<\infty} |F_n(t) - G_m(t)|$$

K-S statistic test is distribution-free, consistent, and more importantly, it is invariant under any local slide or stretching as long as the rank of these data points are unchanged along the specific dimension.

Given this, our greedy algorithm finds $K-S$ statistic at each dimension and chooses the cut which results in the maximal $K-S$ statistic. Thus, we find the largest difference for each marginal distribution and our cut will help to reduce such a difference. Our overall greedy algorithm is in Algorithm IV-B.

Algorithm 2 Greedy Programming for MDST

Procedure GreedyConstruction($[b_1, e_1], \cdots, [b_k, e_k]$)
Require: Datasets D_1 and D_2
if ($|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]|/|D_2[[b_1, e_1] : \cdots : [b_k, e_k]]| < b$) || ($|D_2[[b_1, e_1] : \cdots : [b_k, e_k]]|/|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]| < b$) {b is the lower bound for the statistical tests} **then**
  construct leaf node;
  return ;
end if
if $\min(|D_1[[b_1, e_1] : \cdots : [b_k, e_k]]|, |D_2[[b_1, e_1] : \cdots : [b_k, e_k]]|) < t$ {t is the lower bound for the statistical tests} **then**
  construct leaf node;
  return ;
end if
if NOT Test-Diff($D_1[[b_1, e_1] : \cdots : [b_k, e_k]], D_2[[b_1, e_1] : \cdots : [b_k, e_k]]$) {statistically consistent using two sample test} **then**
  construct leaf node;
  return ;
end if
$MaxD \leftarrow 0$
for all dimension i, $1 \leq i \leq d$ **do**
  $MaxD \leftarrow \max(MaxD, K - S(D_1, D_2, [b_i, e_i]))$;
end for{recursively construct the difference summarization tree}
$y_{i,j} \leftarrow$ the cut with $MaxD$;
GreedyConstruction($[b_1, e_1], \cdots, [b_i, y_{i,j}], \cdots, [b_k, e_k]$);
GreedyConstruction($[b_1, e_1], \cdots, [y_{i,j}, e_i], \cdots, [b_k, e_k]$)

The worst case time complexity of this algorithm is $O(|T| \times N^2)$ where $|T|$ is the number of nodes in the summarization tree which is typically small and N^2 is the worst case time-complexity for the two-sample Friedman-Rafsky's MST test. Clearly, the two-sample test, just like in the dynamic programming, is the dominating factor for the summarization-tree construction as both algorithms need to invoke this procedure repetitively. How to speedup the two sample test is the topic of the next section.

V. Techniques for Efficient Two-Sample Tests

As discussed in the Section IV, the major computational step of the Friedman-Rafsky test is the MST construction of two datasets (Figure 1). In this section, we introduce a novel techniques in alleviating the cost of the MST construction. The basic idea of our approach follows from the observation that the goal of MST is to extract the number of runs (Figure 1) or homogeneous subtrees from MST for the test statistics. Thus, if we can maintain very tight bound on the number of runs, we may avoid the repeatedly reconstruction of the MST. In order to achieve this, we establish several interesting lemmas and theorems in this section.

Why bounds on the runs can help? We start with the simple question on why bounds on the runs for MST can help with the two sample statistical tests. Given two datasets D_1, D_2, and a subregion (a cube) r, let $D_1[r]$ and $D_2[r]$ be the datasets in the region. Let $m = |D_1[r]|$ and $n = |D_2[r]|$. Suppose the exact number of runs (homogeneous subtrees) in the MST on $D_1[r] \cup D_2[r]$ is R and suppose we can derive its lower bound and upper bound, denoted as R^L and R^U. Further, recall that we use the normal distribution of $W(R)$ to determine the significant level (Formula 1). Then we have the following lemma.

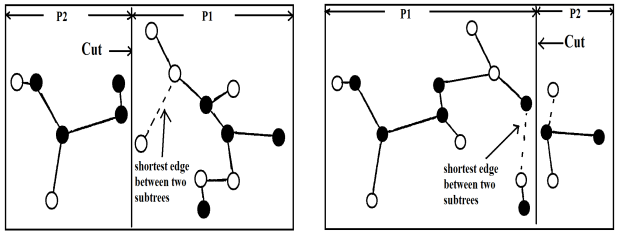
Lemma 1: Let z_α be the critical value for the hypothesis test where the significant level is α. Then, if $W(R^L) \geq z_\alpha$, we cannot reject the null hypothesis (H_0) that the two datasets $D_1[r]$ and $D_2[r]$ are generated from the same distribution; and if $W(R^u) < z_\alpha$, we can reject the null hypothesis.

Proof Sketch: This can be derived by the observation: $W(R^L) \leq W(R) \leq W(R^U)$. □

Based on this lemma, we can see that if we can derive a tight bound for the number of runs R, then, in the most of the cases, $W(R^L) \geq z_\alpha$ or $W(R^u) < z_\alpha$, we can directly reach a statistical conclusion for the hypothesis testing. In other words, we only need to compute the exact $R$when z_α falls between $W(R^L)$ and $W(R^U)$, i.e., $W(R^L) \leq z_\alpha \leq W(R^u)$.

The relationship between global MST and local MST: In order to derive the lower bound and upper bound without repeatedly reconstructing MST, we utilize an interesting relationship between global and local MSTs. Given two datasets D_1, D_2, and a region (a cube) r, let $D_1[r]$ and $D_2[r]$ be the datasets in the region. Let T be the global MST on $D_1[r] \cup D_2[r]$. Let r_1 and r_2 be the two subregions

produced by a cut on r. Let T_1 and T_2 be the two local MSTs on the two subregions r_1 and r_2 ($D_1[r_1] \cup D_2[r_1]$, and $D_1[r_2] \cup D_2[r_2]$), respectively. Further, let us remove all the edges in the global spanning tree which intersect the cut, i.e., those edges whose two adjacent vertices in different subregions. After the removal, the entire MST T forms two forests, one in each subregion. We denote them as F_1 and F_2 for regions r_1 and r_2, respectively. See Figure 2 for illustration.



(a) (b)

Figure 2. (a) the shortest edge with two nodes from same sample and (b) the shortest edge with two nodes from different samples

We introduce the following procedure to construct local MSTs based on the forests. *Let the number of disjoint subtrees in the forest F_1 is n_1. For any two subtrees i and j in F_1, denoted as ST_i and ST_j, we find an edge e_{ij}, which has the shortest distance to link these two subtrees. Specifically, we have*

$$e_{ij} = \arg \min_{(p_1,p_2)} dist(p_1, p_2)$$

where, p_1, p_2 in subtree i and j, respectively. Given this, we construct a concise graph $G = (V, E)$, where V corresponds to each disjoint subtrees in the forest, i.e., we contract each subtree into a vertex, and $E = \{e_{ij} | 1 \leq i < j \leq n_1\}$, i.e., those edges with the shortest distance connecting the corresponding subtrees. Note that G is a complete graph. Then, we find a minimal spanning tree T_G from G. Finally, we can recover all the edges in the T_G to the original edges in $D_1[r_1] \cup D_2[r_1]$, and we denote them as $E(T_G)$. The union between $E(T_G) \cup F_1$ is the minimal spanning tree of $D_1[r_1] \cup D_2[r_1]$.

The following theorem reveals the interesting relationship between the forest F_1 and MST T_1 and the correctness of procedure in constructing the local MST.

Theorem 1: The minimal spanning tree (MST) T_1 on the local region r_1 can be constructed to include all the edges in the global MST T on region r whose both adjacent data points are in subregion r_1. Further, $E(T_G) \cup F_1$ is a minimal spanning tree on local region r_1.

Due to page limitation, the proof will be containd in the full version of the paper.

Bound Maintenance: The above theorem shows that we can have a simple way to derive the MST for each subregion from the global MST. However, it can still be costly to find the edge with the shortest distance between any two subtrees. Here, we will show that we may avoid explicitly reconstruct MST on each part, by maintaining the tight bound of the number of runs for each region utilizing the forest directly.

Theorem 2: Let R be exact number of runs in the MST for region r_1. Let R_i be the number of runs for the i-th subtree, ST_i, of the forest F_1 and let n_1 be the total number of disjoint trees in F_1 Then our bound for the number of runs is:

$$\sum_{i=1}^{n_1} R_i - (n_1 - 1) \leq R \leq \sum_{i=1}^{n_1} R_i$$

The detailed proof will be contained in the full version of the paper.

VI. Experimental Results

In this section, we are particularly interested in the following questions: 1) How concise are the summarization-trees generated from the greedy algorithm compared with the ones generated from the dynamic programming method? 2) How fast does the greedy algorithm perform compared with the dynamic programming method? 3) How much we can gain from the techniques which use the bound estimation and compute the MST using Theorem 1?

To answer these questions, we first collected the cardiology data from Cleveland Clicnic, and then collected a number of real datasets from UCI machine learning archive and CMU Statlib datasets archive. We first run our algorithm on cardiology data with 2 dimensions, and then on synthetic data with both low dimension and high dimension.

We have implemented our methods using C++. Our experiments were performed on a computational cluster featuring 3.20 GHz dual-core Intel Xeon processors, and 4GB main memory.

A. Cardiology Data Set

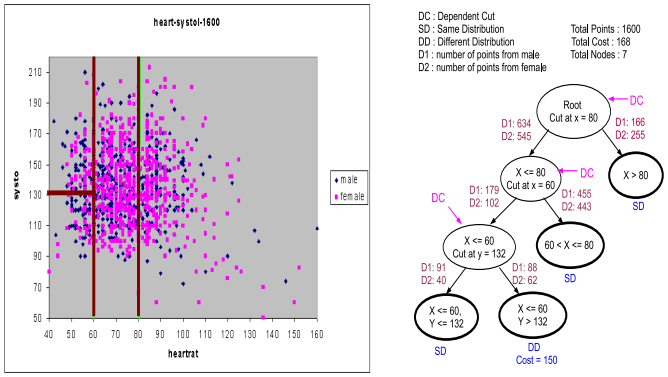
We consider two data sets of cardiological data. The first set includes only male patients whereas the second set includes only female patients. Both sets have 4 attributes: heart rate (heartrat), systolic blood pressure (systol), blood vessel diameter (refdm), and platelet count (pltbsln). Our first expirement determines differences between distributions of heart rate and systolic blood pressure for males and females. This test included 800 data points from each data set.

We found that the distribution for the heart rate and the systolic blood pressure, is different between males and females. Applying our greedy algorithm, we discovered that when the heart rate is bigger than 80, or between 60 to 80, or the heart rate is less than or equal to 60 but the systolic blood pressure is no more than 132, the distribution of heart rate and systolic blood pressure is the same for males and females. Only when the heart rate is less than 60 and the systolic blood pressure is more than 132, males and females have different distribution of heart rate and systolic blood pressure. The total cost of our summarization tree is 168, the number of nodes is 7, running time is 405 ms. On the other hand, applying dynamic programming algorithm the

cost of summariztion tree is 146, the number of nodes is 5, and the running time is 14397 ms.

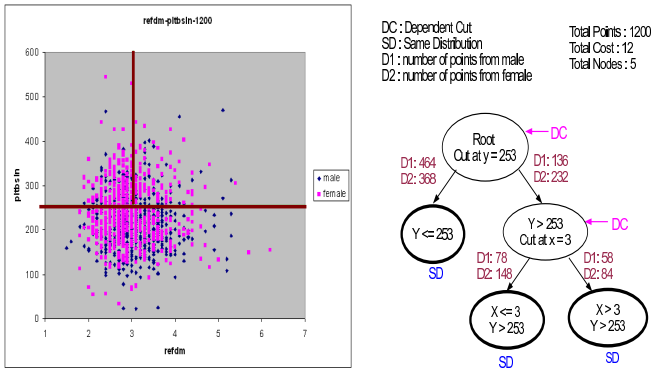
Our second experiment determines differences between distributions of blood vessel diameter and platelet count for males and females. This test included 600 data points from each data set. We first observe that data distributions for both sets are very different. However, applying our greedy algorithm, we found that when the platelet count is no more than 253, or the platelet count is more than 253 and the vessel diameter is no more than 3, or the platelet count is more than 253 and the vessel diameter is more than 3, male and female hava the same distribution of the vessel diameter and platelet count. The summarization tree for greedy algorithm has cost 12, consists of 5 nodes, and running time of 488 ms. For dynamic programming, the tree also has cost 12, consists of 5 nodes but with running time of 14427 ms.

The result of greedy algorithm is shown in Figure.3 and Figure.4. Clearly, our approach shows an efficient and effective way to summarize the difference between two closely related datasets.



(a) (b)

Figure 3. (a) the cut for heart rate and systolic blood pressure with greedy algorithm and (b) the corresponding summarization tree



(a) (b)

Figure 4. (a) the cut for vessal diameter and platelet count with greedy algorithm and (b) the corresponding summarization tree

B. Synthetic Data

Here we list the synthetic datasets in Table II. For experimental purpose, we have filtered out the categorical attributes from the data as we currently focus on the continuous attributes. Table II records the number of the dimension of each dataset after such preprocessing. Further, we generate two slightly different datasets from the original datasets through sampling. The first dataset D_1 is generated by the sampling the original dataset without replacement and by adding a small random noise to all dimensions of each sampled data point. The second dataset D_2 is generated by first performing the same sampling and randomization procedure as the first dataset. Then, we "bump up" certain regions of the sampled data (filling in those regions with oversampling and randomly shifted data points from other regions). For our experiments, we set up $|D_1| = |D_2| = 1000$.

We report our results on two groups, the low-dimension group with only 2 or 3 dimensions and the high-dimension group with more than 3 dimensions.

Table II
DATASETS DESCRIPTION

Data Source	Dimension	# Records	Data Archive
Dorothea	2-D	800	UCI
Madelon	2-D	1800	UCI
Concrete	3-D	1030	UCI
Haberman	3-D	306	UCI
Blood	4-D	374	UCI
Irish	4-D	500	CMU Statlib
Kidney	5-D	76	CMU Statlib
Riverflow	6-D	308	CMU Statlib
Fraser-river	6-D	157	CMU Statlib
Forest Fires	7-D	517	UCI
Cloud	9-D	1024	UCI
Places	10-D	329	CMU Statlib
Bodyfat	14-D	252	CMU Statlib

Results on Low Dimension Datasets: In this group of experiments, we run both dynamic programming and greedy algorithm on the four datasets which have either 2 or 3 dimensions. Table III shows the cost and the size (total number of nodes) of the summarization-tree constructed by both dynamic programming and greedy algorithm. We can see that the greedy algorithm actually produce the summarization tree with similar size to the dynamic programming.

Table IV shows the running time comparison between dynamic programming and greedy algorithms. We also consider the three different versions for the MST reconstruction on dataset on each local region. *Scratch* is to reconstruct the MST from the scratch, and *Forest* is to reconstruct the MST using Theorem 1, and *Bound* is to apply the bound estimation (Theorem 2) to avoid the reconstruction and only reconstruct when the bound fails (Lemma 1) and the construction method is using the *Forest* in the latter case. We apply these three methods for the greedy algorithm and the

dynamic programming. We can see that the *Bound* is much faster than *Forest* in dynamic programming while slightly faster in greedy algorithm. And both of them are much faster than the *Scratch* in dynamic programming and greedy algorithm. Finally, the greedy algorithm is much faster than the dynamic programming approach.

Table III
RESULT ON LOW DIMENSION

	Dynamic Algo.		Greedy Algo.	
	Cost	# Nodes	Cost	# Nodes
Dorothea	392	19	429	19
Madelon	6	3	6	3
Concrete	243	19	314	23
Haberman	6	3	6	3

Table IV
RUN TIME ON LOW DIMENSION (IN MILLISECONDS)
B: Bound F: Forest S: Scratch

	Dynamic Algo.			Greedy Algo.		
	B	F	S	B	F	S
Dorothea	24669	36494	57241	323	338	956
Madelon	10022	17110	43187	252	253	581
Concrete	56232	67444	107491	457	459	1294
Haberman	118639	178055	452179	456	461	1197

Results on High Dimension Datasets: Since the dynamic programming cannot be applied on the high dimensional datasets (it runs out of memory), we focus on reporting the greedy method here. Table V shows the cost and size of the constructed summarization tree in the second and third column, respectively. The fourth, fifth and sixth columns shows the running time of the *Bound*, *Forest* and *Scratch* technique for reconstructing the MST. We can see here the *Bound* and *Forest* techniques gain a 250% to 300% speedup from the *Scratch*. This is higher than the gain we get from the low dimension dataset. We believe one of the reasons for the gain from the high dimension datasets is that the distance computational cost becomes a more significant factor as the dimension increases.

To sum, we can see that the *Bound* and *Forest* approach work very well for speeding up the summarization tree construction.

VII. RELATED WORK

In database research community, change detection is a very important topic in understanding data streams. Ganti et al.[20] developed a framework to measure the deviation between two data sets. They focus on deriving the deviation measure through data mining models, including frequent itemsets, decision trees, and clustering. Kifer et al.[21] proposed a two-sample test which can allow user specify the interested regions and domains for the difference detection between two datasets. However, their method is built upon the order statistics and Kolmogorov-Smirnoff tests which cannot be easily expanded to multidimensional case. Aggarwal [22] proposes the velocity density estimation method based on the kernel methods to diagnose the change in evolving data streams. Zhu et al.[23] proposed a data structure called *Shifted Wavelet Tree* to monitor elastic bursts. Dasu et al. [24] provides a change detection methods by generalizing Kulldorff's spatial scan statistics. Different from these works, we try to derive a parsimonious explanation of the difference between datasets, and our work is built upon the two sample statistic tests.

Table V
RESULT ON HIGH DIMENSION (IN MILLISECONDS)

	Greedy Algo. (B: Bound F: Forest S: Scratch)				
	Cost	# Nodes	Time (B)	Time (F)	Time (S)
Blood	326	16	384	403	1305
Irish	83	14	700	726	1878
Kidney	102	22	666	691	2118
Riverflow	72	11	728	746	2169
Fraserriver	74	13	670	672	2032
ForestFires	39	9	836	855	2182
Cloud	7	3	1061	1062	2624
Places	14	3	1476	1479	3743
Bodyfat	65	13	1644	1683	4748

Statistical research community has developed methods for multivariate two sample tests. Besides the aforementioned Friedman-Rafsky test[7], several other methods have been developed for the same purpose [9], [25], [10]. These methods either have a higher computational complexity or are not as efficient in describing differences between two sets.

Finally, we note that recent work [13] in data mining community is developed for statistical test if there is a change between two datasets. Their method is based on the likelihood ratio and kernel distribution construction. Even though they showed empirically this method can detect the change quite effectively, it is still not clear if it is a consistent test.

VIII. CONCLUSIONS

In this paper, we have introduced a new research problem on how to derive a parsimonious explanation of the difference between two datasets. We have proposed a tree based approach for this purpose. It is built upon rigorous two sample statistical tests. We have described two basic approaches, a dynamic programming approach and a greedy method, to construct the summarization-tree. Besides, we also studied on how to apply Friedman-Rafsky's MST statistic test for tree construction. We discover a couple of interesting results on the global minimal spanning tree (MST) and its subgraph (which is forest). These results allow us to significantly reduce the complete reconstruction of the minimal spanning tree on a subsets of the data when a global MST is given. Our experimental results show

that the dynamic programming approach can work on the low dimension data but is very computationally demanding. The greedy algorithm is very fast and can construct the summarization-tree with conciseness close to the ones being constructed by dynamic programming.

We believe that this work provides a first attempt in constructing a parsimonious explanation of the difference between two datasets. However, this work also gives rise to a number of interesting research questions. First, how can we extend the work to the datasets which have both categorical and numerical attributes? Second, if a local region move from one place in one dataset to a different place in another dataset? Can we automatically detect such difference and add the "moving" as a primitive for difference explanation? In our future work, we plan to investigate these problems.

IX. Acknowledgment

Authors are extremely grateful to Dr. S. Ellis from Cleveland Clinic for a permission to use his cardiological data file.

References

[1] G. I. Webb, S. Butler, and D. Newlands, "On detecting differences between groups," in *KDD '03*.

[2] S. D. Bay and M. J. Pazzani, "Detecting group differences: Mining contrast sets." *Data Min. Knowl. Discov.*, vol. 5, no. 3, pp. 213–246, 2001.

[3] G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences." in *KDD '99*.

[4] S. Sarawagi, "Explaining differences in multidimensional aggregates," in *VLDB '99*.

[5] D. Agarwal, D. Barman, D. Gunopulos, N. E. Young, F. Korn, and D. Srivastava, "Efficient and effective explanation of change in hierarchical summaries," in *KDD '07*.

[6] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods, 2nd Edition*. Wiley Series in Probability and statistics, 1999.

[7] J. H. Friedman and L. C. Rafsky, "Multivariate generalizations of the wald-wolfowitz and smirnov two-sample tests," *The Annals of Statistics*, vol. 7, no. 4, pp. 697–717, 1979.

[8] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes. The Art of Scientific Computing, 3rd Edition*. Cambridge University Press, 2007.

[9] N. Henze, "A multivariate two-sample test based on the number of nearest neighbor type coincidences," *The Annals of Statistics*, vol. 16, no. 2, pp. 772–783, 1988.

[10] P. R. Rosenbaum, "An exact distribution-free test comparing two multivariate distributions based on adjacency," *Journal Of The Royal Statistical Society Series B*, vol. 67, no. 4, pp. 515–530, 2005.

[11] A. Wald and J. Wolfowitz, "On a test whether two samples are from the same population," *The Annals of Mathematical Statistics*, vol. 11, no. 2, pp. 147–162, 1940.

[12] D. A. Darling, "The kolmogorov-smirnov, cramer-von mises tests," *Ann. Math. Statist*, vol. 28, no. 4, pp. 823–838, 1957.

[13] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in *KDD '07*.

[14] N. Henze and M. D. Penrose, "On the multivariate runs test," *Ann. Statist.*, vol. 27, no. 1, pp. 290–298, 1999.

[15] J. L. Bentley and J. H. Friedman, "Fast algorithms for constructing minimal spanning trees in coordinate spaces," *IEEE Trans. Comput.*, vol. 27, no. 2, pp. 97–105, 1978.

[16] O. Nevalainen, J. Ernvall, and J. Katajainen, "Finding minimal spanning trees in a euclidean coordinate space," *BIT*, vol. 21, no. 1, pp. 46–54, 1981.

[17] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl, "Euclidean minimum spanning trees and bichromatic closest pairs," *Discrete Comput. Geom.*, vol. 6, no. 5, pp. 407–422, 1991.

[18] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[19] R. Kerber, "Chi-merge: Discretization of numeric attributes," in *the 9th National Conference on Artificial Intelligence*, 1992.

[20] V. Ganti, J. Gehrke, and R. Ramakrishnan, "A framework for measuring changes in data characteristics," in *PODS*, 1999, pp. 126–137.

[21] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *VLDB '04*.

[22] C. Aggarwal, "A framework for diagnosing changes in evolving data streams," in *ACM SIGMOD*, 2003.

[23] Y. Zhu and D. Shasha, "Efficient elastic burst detection in data streams." in *KDD*. ACM, 2003, pp. 336–345.

[24] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams," in *Interface*, 2006.

[25] M. Schilling, "Multivariate two-sample tests based on nearest neighbors," *J. Amer. Math. Soc.*, vol. 81, pp. 799–806, 1986.