

Statistical Change Detection for Multi-Dimensional Data

Xiuyao Song, Mingxi Wu, Christopher Jermaine, Sanjay Ranka
 Department of Computer and Information Sciences and Engineering
 University of Florida
 Gainesville, FL, USA, 32611
 xsong,mwu,cjermain,ranka@cise.ufl.edu *

ABSTRACT

This paper deals with detecting change of distribution in multi-dimensional data sets. For a given baseline data set and a set of newly observed data points, we define a statistical test called the *density test* for deciding if the observed data points are sampled from the underlying distribution that produced the baseline data set. We define a test statistic that is strictly distribution-free under the null hypothesis. Our experimental results show that the density test has substantially more power than the two existing methods for multi-dimensional change detection.

Categories and Subject Descriptors

G.3 [Mathematics of Computing]: PROBABILITY AND STATISTICS—*reliability*; H.2.8 [DATABASE MANAGEMENT]: Database Applications—*algorithms*

General Terms

Reliability, Algorithms

Keywords

Change detection, density test, kernel density estimation

1. INTRODUCTION

This paper considers the problem of building a rigorous statistical test for detecting change of distribution in multi-dimensional data. Such a test would have numerous applications in a variety of disciplines. For example, one may monitor the information about the set of sales observed in the most recent day (in a commercial enterprise) or the prescriptions written recently (in a hospital) or the phone calls or emails recently observed (in the security domain) and ask:

*Material in this paper is based upon work supported by the National Science Foundation under grants CCF-0325459, IIS-0347408, and IIS-0612170.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'07, August 12–15, 2007, San Jose, California, USA.
 Copyright 2007 ACM 978-1-59593-609-7/07/0008 ...\$5.00.

is the distribution of recently observed data different from what has been observed before?

In developing a test for distributional change, the primary goal should be to construct a test with the greatest power possible – that is, with the ability to detect the most subtle changes in the data while still maintaining a low false positive rate. The need for power results from the way that such a test will be applied in a typical data mining scenario. The test will usually be run many times during the search process, comparing the data against multiple baselines, or using multiple definitions of “recent” (day, week, month, etc.). In such a situation, a multiple hypothesis testing correction such as Bonferroni’s inequality would be required to deal with the fact that repeating the test increases the chance for error [14]. This lowers the acceptable false positive rate for each test. Unfortunately, lowering the false positive rate always lowers the power. Thus, a test should have naturally high power to be useful in this scenario.

1.1 Problem Definition and Prior Work

The abstract problem we consider is as follows. We assume two sets of i.i.d. samples S and S' , taken from two unknown, multi-dimensional, non-parametric distributions F_S and $F_{S'}$, respectively. In practice, S is a baseline data set, and S' contains the most recently observed data that we wish to test for change. We then define the null hypothesis H_0 , which asserts that F_S and $F_{S'}$ are in fact identical. The goal is to design a proper statistical test that is able to refute H_0 if it is not true. If H_0 is true, then the probability of making an error (where the test says that F_S and $F_{S'}$ are different when in fact they are not) should be at most p , where p is a user-supplied parameter.

For uni-dimensional data, many tests from statistics fit into this framework. However, there has been little attention to the problem of extending such tests to multi-dimensional data. Methods such as outlier detection apply to such data [6, 11], but they do not detect a mismatch in distribution among data sets; they check for single points that are not in-keeping with the baseline distribution. Some work from data mining is relevant to *describing* changes in multi-dimensional data [4, 3] but it does not address the statistical significance of those changes. Statistical tests for significant spatial irregularities such as Kulldorff’s spatial scan statistic [12, 15] and those for detecting disease outbreaks [21] are closely related but are more specific in that they do not check for a generic change of distribution – they look for “hot spots” in the data that may signal some sort of disease outbreak.

In fact, aside from solutions that rely on mapping two or

three-dimensional data to a single dimension and applying one of the uni-dimensional tests [13], one of the only tests proposed by the statistics community for this problem was published recently, called the *cross-match* test [16]. However, as we demonstrate experimentally, the cross-match test may be of questionable power, and it is computationally expensive, requiring a maximal matching computation over a graph having $O((|S| + |S'|)^2)$ edges. To function, all of these edges must be stored in main memory; even then, the running time is cubic in the size of the data set. One additional multidimensional change detection test related to Kulldorff’s test was proposed by Dasu et al.[8], but this is a test that relies on a discretization of the data space. Space partitioning schemes tend to suffer from the curse of dimensionality. As such, one might expect that the power of the test will suffer in more than a few dimensions – an issue that we consider experimentally later in this paper.

1.2 Our Contributions

This paper’s contribution is the definition and experimental evaluation of an alternative test for multidimensional distributional change, which we call the *density test*. The density test itself makes use of several novel statistical techniques to avoid using a space partitioning. For example, in order to obtain a sensitive distance metric even for high-dimensional data, we make use of a unique Expectation Maximization [9] algorithm in conjunction with a kernel density estimator to infer the baseline distribution.

In addition to this key technical contribution, the density test is experimentally compared with two alternative tests via a total of more than 60 different change detection tasks over 13 real data sets, having from 3 to 26 dimensions each. The strengths and weaknesses of the various tests are carefully considered. The density test is found to have substantially more power (especially for detecting changes along multiple dimensions at once) compared to the two existing methods, and has a computational cost that is competitive.

1.3 Paper Organization

The remainder of the paper is organized as follows. In Section 2, we give the high-level overview of the density test. We discuss kernel density estimation in Section 3. In Section 4, we define the test statistic and discuss its null distribution. In Section 5 we explain why we need to run the test on two different directions. Section 6 experimentally tests the power and accuracy of the density test compared with the two existing alternatives. Related work is covered in Section 7, and Section 8 concludes the paper.

2. HIGH-LEVEL VIEW OF DENSITY TEST

A generic hypothesis test framework can be used to test for change from S to S' as follows. First, a test statistic $\delta(S, S')$ is defined. In general, δ may encode any arbitrary computation over S and S' . Let Δ be a random variable whose distribution is exactly the distribution of δ under the null hypothesis. Then in order to refute the null hypothesis at a significance level p , we simply check whether δ is found to have an extreme value (assuming a one-tailed test, we may check whether $Pr[\Delta \leq \delta]$ is less than p). If it is, then we can reject the null hypothesis and declare that there has probably been a distributional change observed in the data.

An excellent example of a test that makes use of the generic hypothesis test framework is Kulldorff’s spatial scan

statistic [12] which identifies abnormally “dense” regions in a spatial data set that may signal distributional change in spatial data. First, the data space is discretized into a set of cells, and then the baseline distribution parameters are learned by counting the number of baseline data points falling into each cell. The test statistic δ used by Kulldorff’s method is the maximum value of a likelihood ratio statistic, tested over all possible spatial regions defined by the discretization. The null distribution of the test statistic is obtained by the Monte-Carlo method based on the assumption that the underlying distribution is either a non-homogeneous Poisson distribution or a Bernoulli distribution.

This section describes a new test for distributional change that we call the *density test*. At a high level the density test works as follows:

1. First, the baseline S is randomly partitioned into two halves $S = S_1 \cup S_2$. S_1 will be used for modeling and S_2 for testing.¹ By partitioning S into two exclusive halves, we make sure that the model is independent of the data that we use to calculate the test statistic.
2. Next, to compute the statistic δ , we first compute a kernel density estimator using S_1 ; the resulting density function is denoted by \mathcal{K}_{S_1} . δ is defined based on the difference of log probability density of \mathcal{K}_{S_1} at S' and at S_2 (hence the name “density test”). Intuitively, if S' is very different from S , we expect that δ will take on a small negative value since \mathcal{K}_{S_1} has higher log probability density at S_2 than at S' . On the other hand, if S' does not differ much from S , we expect that the value of δ will be reasonably large (if S' is comparable to S_2 or S' is even “closer” to S_1 than S_2).
3. Due to the Central Limit Theorem, we know that Δ has a normal limiting distribution. After deriving the mean and variance of Δ , we perform a standard null hypothesis test. Depending on whether the observed δ is significantly far out in the tail of Δ , we either reject the null hypothesis and declare **change**, or we declare **noChange** since we do not have strong enough evidence to reject the null hypothesis.

While the above describes the basic outline of the density test, a few points regarding some specifics of the test bear further discussion in the remainder of the paper. First, we consider the problem of learning a kernel model for S_1 . Specifically, we discuss why classical fixed-bandwidth methods are a poor choice, and propose a data-dependent learning algorithm for computing a more appropriate kernel model. Second, we consider a few issues associated with computing δ . Third, we show how to obtain the null distribution of δ by applying the Central Limit Theorem. A bootstrap resampling technique is proposed to estimate the variance of the null distribution. Fourth, we discuss why (for maximum power) the density test should actually be run in two different directions: checking for a change from S to S' and also checking for a change from S' to S .

¹There is a tradeoff between $|S_1|$ and $|S_2|$. A larger $|S_1|$ gives more accurate density model, but the variance of the null distribution will increase due to the smaller size of $|S_2|$ (see Section 4). Currently we use two equal-sized halves. Determining an optimal partitioning is an avenue for future research.

3. DENSITY VIA GAUSSIAN KERNELS

In order to compute δ , the density test makes use of a kernel density estimator (KDE), which is one of the most common non-parametric approaches for learning a data distribution. A KDE approximates the data distribution via superposition of many individual kernel functions at points sampled from the distribution that is being modeled. By allowing each sample to spread its density to nearby areas of the data space, kernel estimators smooth out the contribution of each observed data point over a local neighborhood.

Formally, let $x_1, \dots, x_{|S_1|}$ be the data points in data set S_1 . If S_1 is smoothed using a KDE, then the estimated density at any point s is:

$$\mathcal{K}_{S_1}(s) = \sum_{x_i \in S_1} \frac{1}{|S_1|} G(\Sigma_{x_i}, s - x_i) \quad (1)$$

In this equation, G is the kernel function. One kernel is centered at each of the data points in S_1 . Σ_{x_i} is the bandwidth or spread of the kernel function centered at x_i , and in the case of a k -dimensional Gaussian kernel (which is used by the density test), Σ_{x_i} is a $k \times k$ positive-definite co-variance matrix.

It is generally acknowledged that in practice, the quality of a kernel estimate depends less on the shape of kernel defined by G than on the bandwidth of each kernel. Unfortunately, in defining the density test we faced a significant technical hurdle with respect to choosing the kernel bandwidth: virtually all applications of the KDE method (and almost all papers in the literature) make the implicit assumption that all kernels in the model share the same bandwidth; that is, $\Sigma_{x_i} = \Sigma_{x_j}$ for any (x_i, x_j) pair. Since the power of the density test fundamentally relies on the quality of the kernel model and its ability to determine when data are abnormal, such an assumption is very problematic. Different portions of the data space tend to have different characteristics in “real-life” multi-dimensional distributions. Often, there is a clear boundary past which it is clear from the data that no density exists; kernels close to the boundary should have bandwidths that project no density past this boundary. Kernels in an occupied but sparse portion of the data space should have large bandwidths. Kernels in dense regions of the data space should have tighter bandwidths. Visualizations of high dimensional data may show “spokes” radiating out in different directions from a central hub in the data. In such a situation, the bandwidth within each spoke should be elongated along the spoke but not allow the projection of density into the empty areas within the spokes.

Given the dearth of appropriate methods for modeling such real-life distributional characteristics, we choose to use a variation on the standard statistical method of maximum likelihood estimation to choose the kernel bandwidths that were most likely to have produced the data. Rather than attempting to maximize the likelihood (or log-likelihood) directly as is usually done, we instead attempt to maximize the so-called “pseudo log-likelihood” of the model over all possible choices of each Σ_{x_i} , defined as:

$$L = \sum_{x_j \in S_1} \log \left[\sum_{x_i \in S_1 \wedge i \neq j} \frac{1}{|S_1| - 1} G(\Sigma_{x_i}, x_j - x_i) \right] \quad (2)$$

Note that the pseudo log-likelihood is nothing but the log-likelihood of S_1 over the model, with the constraint that it is known that no data point was generated by the kernel

centered at it. Thus, the probability that a data point is generated by itself is zero. We do not maximize the log-likelihood directly because each data point occurs exactly on the center of a kernel. Thus, a model constructed via a direct maximization of the log-likelihood would simply associate a zero-bandwidth kernel with each data point, so that $G(\Sigma_{x_i}, x_i - x_i)$ is always infinity.

To perform this maximization, we derive an Expectation-Maximization (EM) algorithm. Since the kernel model with Gaussian kernels can be viewed as an instance of a Gaussian Mixture Model (GMM) with $|S_1|$ components, we can use a simple variation on classical Gaussian EM in order to learn the “optimal” bandwidths. Actually, applying EM to learn the bandwidths is somewhat simpler than applying EM on a general GMM because the centroids of the Gaussian components are fixed at the data points of S_1 , and all Gaussian components have equal weight $\frac{1}{|S_1|}$. Thus, these parameters do not need to be learned in classical GMM.

The remainder of this section assumes a basic understanding of the EM framework, and how it is used to learn a GMM. Many excellent tutorials cover this subject (Bilmes’ tutorial [5] is one of the best known and most complete).

Assume S_1 has dimensionality k . Let $\omega_1, \dots, \omega_{|S_1|}$ be the Gaussian kernels centered at $x_1, \dots, x_{|S_1|}$ respectively. The update rule for the bandwidth of the kernel associated with data point i is:

$$\Sigma_{x_i} = \frac{\sum_{x_j \in S_1} P(\omega_i | x_j) (x_j - x_i)(x_j - x_i)^T}{\sum_{x_j \in S_1} P(\omega_i | x_j)} \quad (3)$$

In Equation (3), $P(\omega_i | x_j)$ is the probability that x_j is generated by kernel associated with x_i . This probability is often referred to as the “soft membership” of point x_j in Gaussian component ω_i . The soft membership is computed with the formula given in Equation (4). The second line of the equation is customized to our particular situation, where we take into account the constraint that the j^{th} data point cannot be generated by the kernel centered at that data point. Note that $\sum_{i=1}^{|S_1|} P(\omega_i | x_j) = 1$ still holds for all j ’s, $j = 1, 2, \dots, |S_1|$:

$$\begin{cases} P(\omega_i | x_j) = \frac{p(x_j | \omega_i)}{\sum_{t=1}^{|S_1|} p(x_j | \omega_t)} & i, j = 1, 2, \dots, |S_1|, i \neq j \\ P(\omega_i | x_i) = 0 & i = 1, 2, \dots, |S_1| \end{cases} \quad (4)$$

In this equation, $p(x_j | \omega_i)$ is the density of the i^{th} Gaussian kernel at the point x_j and it is calculated by Equation (5). Again, the second line in the equation is given because of the special pseudo log-likelihood objective function:

$$\begin{cases} p(x_j | \omega_i) = \frac{1}{(2\pi)^{k/2} |\Sigma_{x_i}|^{1/2}} \exp\left(-\frac{1}{2}(x_j - x_i)^T \Sigma_{x_i}^{-1} (x_j - x_i)\right) & i, j = 1, 2, \dots, |S_1|, i \neq j \\ p(x_i | \omega_i) = 0, & i = 1, 2, \dots, |S_1| \end{cases} \quad (5)$$

Now we can summarize how we optimize the bandwidth of the Gaussian kernels in Algorithm 1. According to this algorithm, we stop the computation whenever the iteration number exceeds the maximum iteration number allowed, or when the objective function L converges toward its optimal value. We decide that the objective function has converged if the fractional difference of two consecutive L values is less than ϕ . In our implementation, $\phi = 0.01$ and $MaxIteration = 100$.

Algorithm 1 LearnBandwidth(S_1 , $MaxIteration$, ϕ)

```

1:  $t = 0$ 
2: while  $t < MaxIteration$  do
3:   Compute density  $p(x_j|\omega_i)$  for all  $i, j$  by Equation (5)
4:   Compute soft membership  $P(\omega_i|x_j)$  for all  $i, j$  by Equation (4)
5:   Compute bandwidth  $\Sigma_{x_i}$  for all  $i$  by Equation (3)
6:   Compute the objective function  $L_t$  by Equation (2)
7:   if  $\frac{L_t - L_{t-1}}{L_{t-1}} < \phi$  then
8:     break
9:   end if
10:   $t++$ 
11: end while

```

Is This Method Effective?

Ultimately, the answer to this question is determined by the accuracy of the resulting change detection test, but it is easy to give some anecdotal evidence of how effective the KDE resulting from the EM method described above is.

For example, consider the following simple experiment over the 24-dimensional *Streamflow* data set (see Section 6 for a description of this data set). We first sample 1000 points from the distribution associated with the data set, and then learn two different kernel models from those 1000 points. The first is learned using Scott’s rule [17], which is a standard, uniform bandwidth selection method. The second is learned using the EM algorithm.

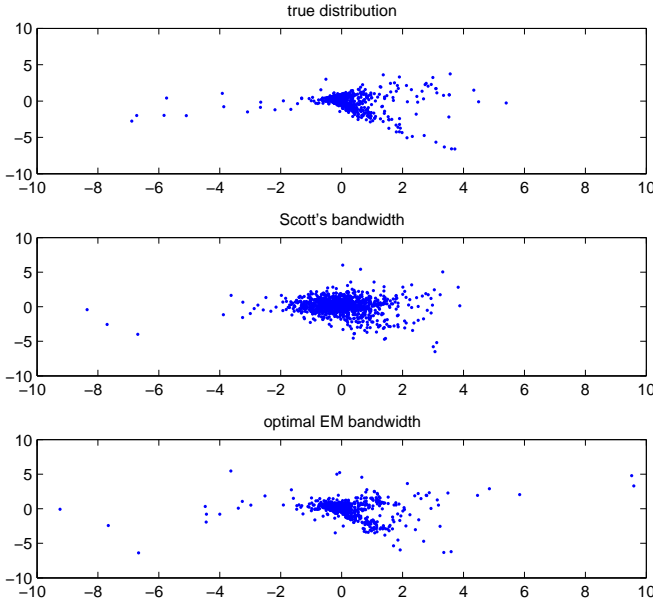


Figure 1: Samples from the original distribution (top), a KDE using Scott’s bandwidth (center), and a KDE learned using our EM method (bottom).

We then sample three additional 1000-point samples. One from the original distribution, one from a standard bandwidth estimator, and one from our EM estimator. Projections of these three samples onto two dimensions (the 23rd and 24th dimensions of the *Streamflow* data set) are shown in Figure 1. It is very clear from the three plots that a

great deal of information regarding the original distribution has been lost through the application of Scott’s estimator. Sampling from the resulting KDE seems to have created a large and shapeless “blob” of points. On the other hand, the EM estimator seems to do an excellent job of preserving the characteristics of the original distribution. The experiments in Section 6 will show how sensitive the resulting KDE can be when it is used to detect changes in high-dimensional data.

4. THE TEST STATISTIC

In the density test, δ is defined using the log probability density of \mathcal{K}_{S_1} at S' and at S_2 . Specifically, δ is simply the difference between the log-likelihood (LLH) of S' and the scaled log-likelihood of S_2 under the inferred density function \mathcal{K}_{S_1} :

$$\delta = LLH(\mathcal{K}_{S_1}, S') - \frac{|S'|}{|S_2|} \times LLH(\mathcal{K}_{S_1}, S_2) \quad (6)$$

Since both S_2 and S_1 are from F_S , if S' is quite different from S_2 , \mathcal{K}_{S_1} will be more likely to generate S_2 than to generate S' , which will result in an extreme small negative value for δ . Thus, a hypothesis test using the above test statistic is a lower one-sided test.

This particular test statistic is chosen for three reasons:

1. δ calculated under the null hypothesis is a sample from Δ , which has a normal distribution (Section 4.1.1).
2. The expected value of mean of Δ is always zero, which means that Δ is always centered at the origin (Section 4.1.2).
3. The variance of Δ can be estimated in a robust fashion (Section 4.1.3).

For these reasons, the density test is easily applicable since the null distribution has a known parametric distribution and only one parameter must be estimated.

4.1 Normality of Δ

In this subsection, we consider the distribution function of Δ in detail.

4.1.1 Applying Central Limit Theorem

We can apply the Central Limit Theorem to argue for the normality of Δ by expanding the formula for the test statistic δ :

$$\begin{aligned} \delta &= LLH(\mathcal{K}_{S_1}, S') - \frac{|S'|}{|S_2|} \times LLH(\mathcal{K}_{S_1}, S_2) \\ &= \log \left\{ \prod_{y \in S'} \mathcal{K}_{S_1}(y) \right\} - \frac{|S'|}{|S_2|} \times \log \left\{ \prod_{y \in S_2} \mathcal{K}_{S_1}(y) \right\} \\ &= \sum_{y \in S'} \log \sum_{x \in S_1} \frac{1}{|S_1|} G(\Sigma_x, y - x) \\ &\quad - \sum_{y \in S_2} \frac{|S'|}{|S_2|} \times \log \left\{ \sum_{x \in S_1} \frac{1}{|S_1|} G(\Sigma_x, y - x) \right\} \quad (7) \end{aligned}$$

Under the null hypothesis, S' and S_2 are two sets of i.i.d. samples from F_S . That means S' and S_2 are sample points from a set of independent random variables whose probability distribution is F_S . Assume S' is generated by the set of

random variables $\{T_1, T_2, \dots, T_{|S'|}\}$, and S_2 is generated by the set of random variables $\{T_{|S'|+1}, T_{|S'|+2}, \dots, T_{|S'|+|S_2|}\}$.

Beginning with Equation (7), the random variable used to produce δ can then be denoted as:

$$\begin{aligned} \Delta &= \sum_{i=1}^{|S'|} \log \sum_{x \in S_1} \frac{1}{|S_1|} G(\Sigma_x, T_i - x) \\ &\quad - \sum_{i=|S'|+1}^{|S'|+|S_2|} \frac{|S'|}{|S_2|} \times \log \left\{ \sum_{x \in S_1} \frac{1}{|S_1|} G(\Sigma_x, T_i - x) \right\} \\ &= \sum_{i=1}^{|S'|} f(T_i) - \sum_{i=|S'|+1}^{|S'|+|S_2|} g(T_i) \end{aligned} \quad (8)$$

where,

$$f(T_i) = \log \sum_{x \in S_1} \frac{1}{|S_1|} G(\Sigma_x, T_i - x)$$

$$g(T_i) = \frac{|S'|}{|S_2|} \times f(T_i)$$

It is well known that if we apply a measurable function to a random variable, the result will still be a random variable. Since both f and g are measurable functions, both $f(T_i)$ and $g(T_i)$ are random variables. Let:

$$\Delta = \Delta_1 - \Delta_2 \quad (9)$$

$$\Delta_1 = \sum_{i=1}^{|S'|} f(T_i) \quad (10)$$

$$\Delta_2 = \sum_{i=|S'|+1}^{|S'|+|S_2|} g(T_i) \quad (11)$$

Given this, we can make the following three observations with respect to Δ_1 :

1. Since all T'_i 's, $i = 1, 2, \dots$ follow the same distribution F_S , it follows that all $f(T_i)$'s have an identical distribution $f(F_S)$.
2. Since all T'_i 's, $i = 1, 2, \dots$ are independent random variables, it follows that all $f(T_i)$'s are independent.
3. Δ_1 is the sum of $|S'|$ independent and identically-distributed random variables. Formally, $\Delta_1 = f(T_1) + f(T_2) + \dots + f(T_{|S'|})$.

The Central Limit Theorem can then be applied, and as a result, we know that Δ_1 approaches a normal distribution – in practice, this happens quite quickly, with only a few dozen samples. A similar argument holds for Δ_2 , which also has a normal limiting distribution. In Equation (10) and Equation (11), all T'_i 's, $i = 1, 2, \dots, |S'| + |S_2|$ are independent, so Δ_1 and Δ_2 are independent and Δ follows a normal limiting distribution as well.

4.1.2 Mean of Δ

The normal distribution has two parameters: its mean and variance. To calculate the mean, assume $\mathbf{E}[f(T_i)] = \mu$.

The expected value of Δ is given by:

$$\begin{aligned} \mathbf{E}[\Delta] &= \mathbf{E}[\Delta_1 - \Delta_2] \\ &= \mathbf{E}[\Delta_1] - \mathbf{E}[\Delta_2] \\ &= \mathbf{E} \left[\sum_{i=1}^{|S'|} f(T_i) \right] - \mathbf{E} \left[\sum_{i=|S'|+1}^{|S'|+|S_2|} g(T_i) \right] \\ &= \sum_{i=1}^{|S'|} \mathbf{E}[f(T_i)] - \sum_{i=|S'|+1}^{|S'|+|S_2|} \mathbf{E}[g(T_i)] \\ &= |S'| \times \mu - |S_2| \times \mathbf{E} \left[\frac{|S'|}{|S_2|} \times f(T_i) \right] \\ &= |S'| \times \mu - |S_2| \times \frac{|S'|}{|S_2|} \times \mu \\ &= 0 \end{aligned}$$

4.1.3 Variance of Δ

Similarly we can derive the variance. Assuming $\mathbf{Var}[f(T_i)] = \sigma^2$, the variance of Δ is given by:

$$\begin{aligned} \mathbf{Var}[\Delta] &= \mathbf{Var}[\Delta_1 - \Delta_2] \\ &= \mathbf{Var}[\Delta_1] + \mathbf{Var}[\Delta_2] \\ &= \mathbf{Var} \left[\sum_{i=1}^{|S'|} f(T_i) \right] + \mathbf{Var} \left[\sum_{i=|S'|+1}^{|S'|+|S_2|} g(T_i) \right] \\ &= \sum_{i=1}^{|S'|} \mathbf{Var}[f(T_i)] + \sum_{i=|S'|+1}^{|S'|+|S_2|} \mathbf{Var}[g(T_i)] \\ &= |S'| \times \sigma^2 + |S_2| \times \mathbf{Var} \left[\frac{|S'|}{|S_2|} \times f(T_i) \right] \\ &= |S'| \times \sigma^2 + |S_2| \times \left(\frac{|S'|}{|S_2|} \right)^2 \times \sigma^2 \\ &= (|S'| + \frac{|S'|^2}{|S_2|}) \sigma^2 \end{aligned}$$

4.2 Estimating σ^2

As long as we know the value of σ^2 , we have the exact null distribution. Unfortunately, this value is unknown and must be estimated.

The first idea that comes to mind is to estimate σ^2 with the unbiased estimate $\frac{|S_2|}{|S_2|-1} \mathbf{Var}[f(S_2)]$. Unfortunately, this is a bad idea. If this estimator is less than the true σ^2 , the estimated distribution of Δ will improperly shrink towards the mean. This will introduce an additional type I (false positive) error, that must be quantified.

In order to define an estimator V such that $\mathbf{Pr}[\sigma^2 > V] = \beta$, we use the bootstrap percentile method [10]. Rather than constructing a two-tailed confidence interval, we need only compute the one-sided $(1 - \beta)$ upper confidence limit on σ^2 and then use this upper limit as the estimator of σ^2 . Algorithm 2 gives the process, which introduces an additional type I error of β .

There are now two sources of type I error. First, δ could be far out in the tail of null distribution, even if it is a sample from Δ . Assume this error is α . Second, we could underestimate the variance of Δ by using any particular estimate. This error is denoted by β in Algorithm 2. A bound on the type I error is given by $p = \alpha + \beta$.

For a user-supplied p , there is a tradeoff between α and β .

Algorithm 2 EstVar($V, S_2, estSize, \beta$)

- 1: Initialize the array Est
 - 2: **for** $t = 1$ to $estSize$ **do**
 - 3: Bootstrap resample R from S_2 , where $|R| = |S_2|$
 - 4: $Est[t] \leftarrow \frac{|S_2|}{|S_2|-1} \mathbf{Var}[f(R)]$
 - 5: **end for**
 - 6: $V \leftarrow [estSize \times (1 - \beta)]^{th}$ percentile of Est
 - 7: $\hat{\mathbf{Var}}[\Delta] \leftarrow (|S'| + \frac{|S'|^2}{|S_2|})V$
-

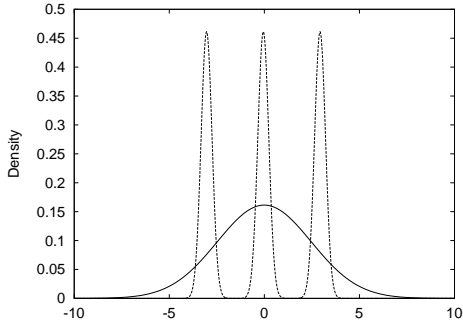


Figure 2: Two distributions that must be checked for deviation in both directions, because one is covered by the other.

Fortunately, it is easily possible to choose α and β such that $\alpha + \beta = p$ and the power of the resulting test is maximized. To do this, we consider each possible β value (each of which corresponds to a different bootstrapped variance), and compute the resulting cutoff value for declaring a **change** result. Over all possible β values, we choose the resulting cutoff that is most aggressive, and use that during our actual test. Algorithm 3 gives the process to find such a critical value.

Algorithm 3 CriticalValue($p, stepSize$)

- 1: $M = \frac{p}{stepSize} - 1$, M is the number of (α, β) pairs
 - 2: $\alpha_i = i \times stepSize, i = 1, 2, \dots, M$
 - 3: $\beta_i = p - \alpha_i, i = 1, 2, \dots, M$
 - 4: Let C be an array of critical values
 - 5: **for** $i = 1$ to M **do**
 - 6: Run EstVar($V, S_2, estSize, \beta_i$) to obtain $\hat{\mathbf{Var}}[\Delta]$.
 - 7: Find c such that $P(\Delta \leq c) = \alpha_i, C[i] = c$.
 - 8: **end for**
 - 9: $C_{max} \leftarrow$ largest value in array C .
-

4.3 Applying The Full Test Procedure

In this subsection, we briefly review the full test procedure. First, S is partitioned into S_1 and S_2 . Then EM is used to learn the kernel model over S_1 . Next we calculate the value of δ for a given pair of S_2 and S' through Equation (7). Second, we obtain the critical value C_{max} through Algorithm 3. Third, we compare δ with this critical value. If $\delta < C_{max}$, we will declare a **change**. Otherwise, it means δ is not significantly small and we will declare **noChange**.

5. RUNNING IN TWO DIRECTIONS

The density test of Section 4.3 is complete, and can be used exactly as described to check for a change of distribu-

tion. However, the power of the test may be substantially increased if it is run twice, once in each direction. This is because the test is not symmetric, and change from S' to S may be more obvious than change from S to S' . For example, imagine that F_S is a uni-dimensional Gaussian, while $F_{S'}$ is a uni-dimensional mixture of low-variance Gaussians, where $\mathbf{Var}(F_S) = \mathbf{Var}(F_{S'})$ and $\mathbf{E}(F_S) = \mathbf{E}(F_{S'})$. This is illustrated in Figure 2, where $F_{S'}$ is a mixture of three Gaussians. A typical sample from $F_{S'}$ will have data points located only in relatively high-density regions of F_S , which means that the δ statistic may be useless for comparing F_S and $F_{S'}$. However, a large enough sample from F_S will likely have data points located in the low-density troughs between the peaks in $F_{S'}$. This renders a test for distributional change from S' to S far more sensitive than a test of change from S to S' .

In order to remove the directionality, we run the test twice. First, the test is run exactly as described in Section 4.3, except that the false positive rate is held at $p/2$. If no change is observed, the test is run a second time in exactly the same way, except that the roles of S and S' are reversed. Running the test twice with a false positive rate of $p/2$ for each run ensures that the overall false positive rate is less than p .

6. EXPERIMENTS

This section details a set of experiments designed to test the utility of the density test, as well as two other available tests for multidimensional data. We describe three sets of experiments, each of which is designed to address one of the following questions:

1. Is the false positive rate of our test correctly governed by the input parameter p ?
2. How does the power (false negative rate) of our approach compare with that of the other two alternative methods, the cross-match test and kdq-tree test?
3. How scalable is our approach compared with the cross-match test and kdq-tree test?

Datasets. Our tests make use of 13 experimental data sets, which are generated from the 13 real data sources described in Table 1. The dimensionality is indicated after each data source.

We require that the size of each data source be reasonably large so that we can sample with replacement in order to obtain a true multivariate probability distribution without too many duplicate values. Among the 13 data sources, five have relatively small size: *CAPeak*, *Bodyfat*, *Boston*, *FCA-Tread* and *FCATmath*. For these small-sized data sources, we “bump up” the data set size to 20,000 points while maintaining the distributional characteristics of the data as follows. We first randomly draw a sample point A from the small-sized data source. We then sample five points, A_1 to A_5 (with replacement) from A ’s five nearest neighbors. A and A_1 to A_5 are averaged to produce a new data point. We repeat the process 20,000 times.

For our experiments, we divide the data sources into 2 groups, the *low-D group* and the *high-D group*. The low-D group consists of data sources from 3 to 10 dimensions. The high-D group consists of data sources from 15 dimensions to 26 dimensions. All three change detection methods are tested using $|S| = |S'| = 850$ for the low-D group. 850 is

Table 2: False positive % on low-D group data sets.

<i>test</i>	<i>average</i>	D_1	D_2	D_3	D_4	D_5	D_6
<i>density</i>	5	4	3	8	3	8	6
<i>cross-match</i>	8	7	11	7	7	8	10
<i>kdq-tree</i>	5	5	6	3	7	1	8

Table 3: False positive % on high-D group data sets.

<i>test</i>	<i>average</i>	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}
<i>density</i>	6	4	7	3	8	10	2	8
<i>kdq-tree</i>	1	1	0	0	0	0	4	0

chosen because this is the largest size that is computationally feasible using the cross-match test. However, for the high-D group, more samples are required to accurately detect distributional change – a consequence of the well-known “curse of dimensionality”. For this group, $|S| = |S'| = 3500$ is used, and the cross-match test is not evaluated.

6.1 Experiment One: False Positive Rate

To test whether the various methods can correctly control the false positive rate, we sample $N(1700 \text{ or } 7000)$ data points (with replacement) from a data space source and call these N data points an *instance*. We repeat this sampling 100 times to obtain 100 instances. Because an instance is sampled with replacement from a finite population, each of the data points in an instance is identically distributed and any split of the instance in two (without looking at the data) will result in two subsets with the same generative distribution.

To estimate the false positive rate of each of the change detection methods over a data set, we split each instance evenly and run the various tests. The change detection test either says there is a distributional change in this instance, or says there is not any distributional change in this instance. The false positive of a test can be estimated by computing the fraction of the time that the test refutes the null hypothesis over the 100 instances of a data set.

The p value is 0.08 for all the three methods. In the density test, $estSize = 4000$ in Algorithm 2. In Algorithm 3, $p = 0.04$ (due to the two-directional test) and $stepSize = 0.002$. In the kdq-tree test, we use the same parameter settings as in the original paper [8]. All experiments were run on an Intel Xeon machine with dual CPU of 2.8GHz, and 5GB of RAM. We keep this setup for all the experiments throughout this section.

The false positive results are reported in Table 2 for low-D group and in Table 3 for high-D group. In those tables, D_i represents the experiment data set created out of the i^{th} data source given in Table 1. The average false positive rate over 13 experiment data sets is given in the “average” column of the tables.

Discussion. These results show that all the three change detection methods have an average false positive rate at or less than the 8% allowed by the supplied p value, and hence a positive result from any of the tests seems to be a safe indicator of distributional change. Both the density test and the cross-match test have a false positive rate very close to the desired p value, and the kdq-tree test tends to be rather conservative.

We have also run similar tests at other p values, and find

that the density test also correctly controls the false positive rate in this case. For $p = 0.04$ and $p = 0.06$, the average false positive rate over all 13 data sets was 3.5% and 4.7% respectively.

6.2 Experiment Two: Power

Experiment Setup. In order to test the false negative rate of the various tests (a.k.a the “power”), it is necessary to create an instance $\langle S, S' \rangle$ such that S and S' are samples from F_S and $F_{S'}$ respectively, where $F_{S'}$ is not equivalent to F_S . Because any distributional change in the real world takes place gradually, if we take a snapshot of the distributional change at some moment, $F_{S'}$ will be a mixture of distribution F_S and a different distribution F_C . Thus, we choose to model distributional change by creating $F_{S'}$ by sampling from some F_C with a probability of L and sampling from F_S with a probability of $1 - L$ for some parameter L . Unless otherwise specified, F_S is the data source described in the beginning of Section 6.

We consider five different types of distributional change, created by five different methods for defining F_C . Three of the changes take place in the full-dimensional space, and two are single-dimensional changes. F_C is defined as follows.

Full-Dimensional Changes

gauss: In this case, F_C is obtained by sampling from the original data set and adding Gaussian noise to all the dimensions of the data source.

gmm: In this case, F_C is a GMM which has three equal-weight components. The centroids of the Gaussian mixture model are three outliers chosen from the original data source. The covariance matrix uses the variance of the subset on the diagonal.

cluster: We divide the data source into two clusters by a K-means clustering algorithm. F_S is the cluster of larger size, and F_C is the cluster of smaller size.

Single-Dimensional Changes

add1D: In this case, F_C is created by adding a standard 1-D Gaussian variable to a single randomly-chosen dimension of the original data set. In each instance of the changed data set a random dimension is selected.

scale1D: Here F_C is created just as in the previous case, except that the value of the randomly-selected dimension is multiplied by two.

For each data source, we create five experimental data sets, with each one corresponding to one type of distributional change. Each experimental data set contains 100 individual instances. The parameter L is chosen based upon the characteristics of the data and the type of change. Although the value of L will not affect the fairness of experiment because all the three detection methods are tested on the same sets of data, we still need to carefully choose L in order to make the experiment informative: it should not be too easy to detect the change, nor should it be too difficult. We calibrate the value of L so that the *cross-match* test (for low-D group) and the *kdq-tree* test (for high-D group) can detect around 50% of the changes.

The false negative rates over the low-D group are reported in Table 4. The false negatives of high-D group are reported in Table 5. In order to make our experiments reproducible, we give the values of parameter L for each type of change over all the 13 data sources in Table 6.

Table 4: The false negative (%) of low-D group.

changes	test	avg	D_1	D_2	D_3	D_4	D_5	D_6
gauss	density	1	0	0	3	0	0	0
	cross-match	46	47	47	45	66	29	44
	kdq-tree	55	0	23	64	87	77	79
gmm	density	0	0	0	0	0	1	1
	cross-match	43	24	46	34	56	53	44
	kdq-tree	23	0	2	0	62	51	21
cluster	density	0	1	0	0	0	0	1
	cross-match	47	49	36	67	40	51	41
	kdq-tree	13	0	0	78	0	0	0
add1D	density	30	26	33	36	27	27	29
	cross-match	60	77	56	62	60	52	51
	kdq-tree	74	76	20	82	85	96	87
scale1D	density	14	2	2	17	23	20	20
	cross-match	52	49	44	49	68	55	48
	kdq-tree	65	7	82	76	68	66	90

Table 5: The false negative (%) of high-D group.

changes	test	avg	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}
gauss	density	6	0	0	0	1	39	0	0
	kdq-tree	55	44	32	36	61	60	79	70
gmm	density	0	0	0	1	0	0	0	0
	kdq-tree	63	69	52	69	67	70	56	55
cluster	density	7	14	1	1	3	18	6	7
	kdq-tree	69	86	57	72	79	60	75	51
add1D	density	28	0	0	70	82	42	1	0
	kdq-tree	75	87	79	75	80	60	71	70
scale1D	density	22	0	2	38	58	52	5	1
	kdq-tree	70	62	72	68	54	96	74	65

Discussion. Table 4 and Table 5 show that the density test is the most powerful out of the three methods compared. Though it is true that there was some variation from data set to data set, it seems clear that the density test is the most obvious, general-purpose choice. For each of the five types of changes, the density test had the lowest average false negative rate. Depending upon the type of change, the kdq-tree test and the cross-match test were generally comparable on the low-D group, with the former doing better on full-dimensional changes, and the latter doing better on single-dimensional changes.

6.3 Experiment Three: Scalability Analysis

Our final set of experiments test ability of the three change detection methods to scale to larger data sets.

Experiment Setup. We test the scalability of the methods in terms of their running time on data sets of different size. The experiment is run on the 24-dimensional data source *Streamflow*. We sample repeatedly (with replacement) from

the *Streamflow* source to create a series of data sets of size 100, 200, 300, . . . 1000, 2000, . . . 7000.

We record the length of time required to run the change detection test to completion. The density test was implemented in Matlab. The cross-match test was implemented in Matlab as well, with the matching algorithm that is required by the test as an external subroutine being implemented in C (the C implementation was the one recommended by the designer of the cross-match test). The kdq-tree test is implemented in C. The results are reported in Table 7.

Discussion. The kdq-tree has significant performance benefits over both the density test and the cross-match test. Given that the kdq-tree relies mostly on a computationally efficient recursive partitioning of the data space, this is not too surprising. However, the density test still scales to reasonable data set sizes (see the discussion in the Conclusion Section of the paper). The cross-match test is generally unusable past a few thousand data points.

We also point out that the density test has a very high, one-time, fixed cost for the construction of the KDE. After this cost has been paid, the model can be saved and the test can be run repeatedly in a very small fraction of the time reported in Table 7. To illustrate the fraction of the total time associated with the density test that is a one-time cost, consider Table 8. This Table shows the fraction of the running time for the density test that is devoted to one-time computation for each of the data set sizes. The average fraction is around 84%. These results show that if the same algorithm has to be repeated multiple times, the computational cost of the density test is only about 4 times more than the tree-based method. For a dataset with 7000 points, the overall requirements are on the order of a minute. This makes it practical for a number of applications.

7. RELATED WORK

In this section, we briefly review the three tests from the literature that are closest to our own. We also briefly discuss alternative methods for kernel bandwidth selection.

The first test, based upon Kulldorff’s spatial scan statistic [12], can be used to find significant spatial clusters in data; the particular application that it was developed for is detecting disease outbreaks. This statistic is obtained by first partitioning the geographic space into cells which may be of irregular shapes. Then a zone (denoted by z) is defined as the combined region of any number of contiguous cells which can be enclosed by a circle. Denote the probability that an individual is a case within a zone by p , and use q to denote the probability associated with the region outside the zone. Kulldorff’s test statistic is defined as the likelihood ratio of $\frac{\sup_{z \in Z, p > q} L(z, p, q)}{\sup_{p=q} L(z, p, q)}$, where $L(z, p, q)$ denotes the likelihood of observing the cases inside and outside zone z simultaneously. The null hypothesis for the resulting test is $H_0 : p = q$, and the alternative hypothesis is $H_1 : p > q$. The null distribution of the likelihood ratio test statistic is obtained by the Monte-Carlo method based on the assumption that the points are generated by either a non-homogeneous Poisson process or a Bernoulli process. Several research groups have looked at speeding this test [1, 2, 15].

Though Kulldorff’s test does check for a change of distribution, it looks for a very specific distributional change. As such, Kulldorff’s test is not really suitable for general-purpose change detection. Dasu et al. [8] proposed an

information-theoretic approach to detecting changes in multi-dimensional data streams that generalizes Kulldorff's test. Their approach depends on a spatial partitioning scheme (called a *kdq-tree*) to partition the data space into small cells. Based on the data count in each cell, two empirical distributions are built, representing the reference window and the testing window respectively. The Kullback-Leibler (KL) distance is used to measure the distance between the two empirical distributions. In order to measure the significance of the obtained KL-distance, they use a non-parametric bootstrap method.

A third test for multivariate distributional change from the statistics literature is the cross-match test [16]. In the cross-match test, every observation in $S \cup S'$ is ranked along each dimension. The rank of an observation x_i , denoted by r_i , is a vector with k elements. Then the inter-point distance δ_{ij} between two observations x_i and x_j is defined to be the Mahalanobis distance between their ranks. Formally, the distance is given by $\delta_{ij} = (r_i - r_j)^T M^{-1} (r_i - r_j)$, where M is the sample variance-covariance matrix of the ranks r_i . After that, all the $\binom{|S| + |S'|}{2}$ pairwise distances are used to construct an optimal non-bipartite matching, i.e. a matching of the observations into disjoint pairs to minimize the total distance within pairs. The cross-match statistic, A_1 , is defined to be the number of pairs containing one observation from S and one from S' . A_1 is shown to have a restricted occupancy distribution. Furthermore, the conditional distribution of A_1 given $|S'|$ converges to the normal distribution.

Several data-driven methods have been proposed for kernel bandwidth selection. The plug-in rule [18] seems to be the most widely used. This method assumes fixed bandwidth for all the kernels and is not suitable when data exhibits local scale variations. The optimal bandwidth selection methods proposed by Silverman [19] and Wand and Jones [20] are of limited practical use for multidimensional data as the derivation of asymptotics involves multivariate derivatives and higher order Taylor expansions. We did not find any method in the literature that associates different bandwidth for each kernel for more than three dimensional datasets. The method proposed by Comaniciu [7] allows for variable bandwidth but assumes that the range of data scales is known and chooses the bandwidth that is the most stable across scales for each data point.

8. CONCLUSION

We have described a new test for distributional change called the *density test*, and evaluated the test via an extensive set of experiments. Across all of our experiments, the density test uniformly shows the most power compared to the available alternatives. In terms of running time, the density test is not the most efficient of the three methods tested, but it does easily scale to data sets that are thousands of points in size and is competitive. Furthermore, for an application where new data are repeatedly tested against the same baseline, the high one-time cost associated with constructing the kernel model can be amortized across all runs of the test, lowering the computational cost by nearly 90%.

9. REFERENCES

- [1] D. Agarwal, A. McGregor, J. Phillips, S. Venkatasubramanian, and Z. Zhu. Spatial scan statistics: Approximations and performance study. In *SIGKDD*, 2006.
- [2] D. Agarwal, J. M. Phillips, and S. Venkatasubramanian. The hunting of the bump: On maximizing statistical discrepancy. In *SODA*, 2006.
- [3] C. Aggarwal. A framework for change diagnosis of data streams. In *SIGMOD*, pages 575–586, 2003.
- [4] S. Bay and M. Pazzani. Detecting group differences: Mining contrast sets. *Data Min. Knowl. Discov.*, 5(3):213–246, 2001.
- [5] J. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models, 1997.
- [6] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *SIGMOD*, pages 93–104, 2000.
- [7] D. Comaniciu. An algorithm for data-driven bandwidth selection. *IEEE Transactions on PAMI*, 25(2):281–288, 2003.
- [8] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Interface*, 2006.
- [9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JRSS Series B*, 39(1):1–38, 1977.
- [10] B. Efron and R. J. Tibshirani. *An introduction to the Bootstrap*, volume 57 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, 1993.
- [11] E. Knorr, R. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB J.*, 8(3-4), 2000.
- [12] M. Kulldorff. A spatial scan statistic. *Comm. in Statistics: Theory and Methods*, 26(6):1481–1496, 1997.
- [13] J.-F. Maa, D. Pearl, and R. Bartoszynski. Reducing multidimensional two-sample data to one-dimensional interpoint comparisons. *The Annals of Statistics*, 24(3):1069–1074, 1996.
- [14] R. Miller. *Simultaneous Statistical Inference*. McGraw-Hill, New York, 1966.
- [15] D. Neill and A. Moore. Rapid detection of significant spatial clusters. In *SIGKDD*, 2004.
- [16] P. R. Rosenbaum. An exact distribution-free test comparing two multivariate distributions based on adjacency. *JRSS Series B*, 67(4):515–530, 2005.
- [17] D. Scott. *Multivariate Density Estimation: Theory, Practice and Visualization*. Wiley-Interscience, New York, 1992.
- [18] S. Sheather and M. Jones. A reliable databased bandwidth selection method for kernel density estimation. *JRSS Series B*, (53):683–690, 1991.
- [19] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.
- [20] M. Wand and M. Jones. *Kernel Smoothing*. Chapman and Hall, 1995.
- [21] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner. Bayesian network anomaly pattern detection for disease outbreaks. In *ICML*, pages 808–815, 2003.

Table 1: The 13 experiment data sources.

Data source	Description
<i>CAPeak</i> (3-D)	Contains 1,817 records of mountain peaks in California, and includes each peak’s longitude, latitude and height.
<i>El Nino</i> (5-D)	The data is from UCI KDD archive. It contains oceanographic and surface meteorological readings taken from a series of buoys positioned throughout the equatorial Pacific. we remove those data records with missing values and end up to have 93,935 records. We also remove the Spatio-Temporal attributes from the dataset since the changes on these attributes are not meaningful.
<i>Houses</i> (9-D)	The data set is from CMU Statlib datasets Archive. It contains 20,640 observations on housing prices with 9 economic covariates.
<i>Pine</i> (10-D)	This data set and following two data sets, <i>spruce</i> and <i>krummholz</i> , are from UCI KDD Archive. These data sets describe the forest cover type for 30 x 30 meter cells. We extract the quantitative attributes of the datasets. The number of records are 35,754, 211,840 and 20,510 for Ponderosa Pine, Spruce-Fir and Krummholz respectively.
<i>Spruce</i> (10-D)	Refer to the <i>pine</i> data set.
<i>Krummholz</i> (10-D)	Refer to the <i>pine</i> data set.
<i>Bodyfat</i> (15-D)	The data set is from CMU Statlib datasets archive. It gives the estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men.
<i>Boston</i> (16-D)	The data set is from CMU Statlib datasets Archive. It consists of 506 records of Boston house price data, with corrections and augmentation of the data with the latitude and longitude of each observation. We remove 5 attributes that are irrelevant to finding interesting changes.
<i>Batting</i> (17-D)	This data set and the following <i>pitching</i> data set are from The Lahman Baseball Database. They contain pitching and batting statistics for Major League Baseball players from 1871 through 2005. The number of records is 85,979 and 36,245 for <i>batting</i> and <i>pitching</i> respectively.
<i>Pitching</i> (22-D)	Refer to <i>batting</i> data set.
<i>Streamflow</i> (24-D)	We create this data set by the source data from U.S. Geological Survey (USGS) and National Climate Data Center (NCDC). It contains 7305 records of the weather and stream flow status on some observation stations in CA state. We applied PCA reduction on the data set to reduce the dimensionality down to 24.
<i>FCATread</i> (25-D)	This data set and the following <i>FCATmath</i> dataset is from National Center for Education Statistics (NCES) and Florida Information Resource Network. we create the data sets by matching the regular FL schools’ information and the grade 3’s FCAT reading and FCAT math scores of year 2002. The number of records is 1404 for <i>FCATread</i> and 1405 for <i>FCATmath</i> .
<i>FCATmath</i> (26-D)	refer to <i>FCATread</i> data set.

Table 6: The parameter L of every change on all the 13 data sources.

parameter L	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}
<i>gauss</i>	.17	.2	.15	.1	.2	.15	.1	.11	.08	.068	.19	.05	.05
<i>gmm</i>	.2	.15	.15	.1	.11	.12	.04	.047	.067	.0435	.065	.058	.035
<i>cluster</i>	.12	.14	.12	.12	.1	.12	.019	.035	.035	.03	.11	.025	.03
<i>add1D</i>	.12	.35	.25	.3	.4	.3	.3	.25	.08	.1	.8	.25	.25
<i>scale1D</i>	.18	.4	.3	.25	.35	.3	.2	.2	.2	.2	.8	.2	.3

Table 7: The running time on *Streamflow* data set. Time unit is seconds by default. Otherwise, m stands for minutes, and h stands for hours.

size	100	200	300	400	500	600	700	800	900	1k	2k	3k	4k	5k	6k	7k
<i>density</i>	0.2	0.4	0.9	1	2	2	5	5	7	6	22	52	91	2m	3m	4m
<i>cross-match</i>	1.2	4.6	10	19	31	46	65	88	114	146	15m	51m	2.4h	4.6h	8h	26h
<i>kdq-tree</i>	0.01	0.02	0.03	0.06	0.08	0.1	0.2	0.2	0.3	0.3	0.8	2	3	7	10	13

Table 8: Fraction of the time (in %) reported in table 7 that is devoted to one-time computation that could be re-used for multiple tests.

average	100	200	300	400	500	600	700	800	900	1k	2k	3k	4k	5k	6k	7k
84	73	71	78	78	83	81	90	89	91	87	89	87	88	88	87	88