# Mass Estimation and Its Applications

Kai Ming Ting, Guang-Tong Zhou*, Fei Tony Liu, James Tan Swee Chuan†
Gippsland School of Information Technology
Monash University
Australia
{kaiming.ting,tony.liu}@monash.edu, zhouguangtong@gmail.com,
jamestansc@unisim.edu.sg

## ABSTRACT

This paper introduces mass estimation—a base modelling mechanism in data mining. It provides the theoretical basis of mass and an efficient method to estimate mass. We show that it solves problems very effectively in tasks such as information retrieval, regression and anomaly detection. The models, which use mass in these three tasks, perform at least as good as and often better than a total of eight state-of-the-art methods in terms of task-specific performance measures. In addition, mass estimation has constant time and space complexities.

## Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Miscellaneous; I.5 [**Pattern Recognition**]: General

## General Terms

Algorithms, Theory

## 1. INTRODUCTION

> 'Estimation of densities is a universal problem of statistics (knowing the densities one can solve various problems.)' — V.N. Vapnik [16].

Density estimation has been the base modelling mechanism used in many techniques designed for tasks such as classification, clustering, anomaly detection and information retrieval. For example in classification, density estimation is employed to estimate class-conditional density function (or likelihood function) $p(x|j)$ or posterior probability $p(j|x)$—the principal function underlying many classification methods e.g., mixture models, Bayesian networks, and Naive Bayes. Examples of density estimation include kernel density estimation, $k$-nearest neighbours density estimation, maximum likelihood procedures or Bayesian methods.

We shows in this paper that a new base modelling mechanism called **mass estimation** possesses different properties from those offered by density estimation:

- A mass distribution stipulates an ordering from core points to fringe points in a data cloud. In addition, this ordering accentuates the fringe points with a concave function—fringe points have markedly smaller mass than points close to the core points. These are the fundamental properties required for many tasks, including anomaly detection and information retrieval. In contrast, density estimation is not designed to provide an ordering.
- Mass estimation is more efficient than density estimation because mass is computed by simple counting and it requires only a small sample through an ensemble approach. Density estimation (often used to estimate $p(x|j)$ and $p(j|x)$) require a large sample size in order to have a good estimation and is computationally expensive in terms of time and space complexities [7].
- Mass can be interpreted as a measure of relevance with respect to the concept underlying the data, i.e., core points indicate that they are highly relevant and fringe points indicates that they are less relevance. We demonstrate in this paper that a relevance feature space consists of a vector of masses estimated from data is very effective for three data mining tasks: information retrieval, regression and anomaly detection.

Mass estimation has two advantages in relation to efficacy and efficiency. First, the concavity property mentioned above ensures that fringe points are 'stretched' to be farther from the core points in a mass space—making it easier to separate fringe points from those points close to core points. This property, otherwise hidden, can then be exploited by a data mining algorithm to achieve a better result for the intended task than the one without it. We show the efficacy of mass in improving the task-specific performance of four existing state-of-the-art algorithms in information retrieval and regression tasks in this paper. The significant improvements are achieved through a simple mapping from the original space to a mass space using the mass estimation mechanism introduced here.

*Guang-Tong is a student at School of Computer Science and Technology, Shandong University, China. He was visiting Monash University, supported by a scholarship from Shandong University, when this research was conducted.

†James is now at SIM University, Singapore.


*KDD'10,* July 25–28, 2010, Washington, DC, USA.

**Table 1: Symbols and notations.**

| | |
|---|---|
| $\mathcal{R}^u$ | A real domain of $u$ dimensions |
| $x$ | An one-dimensional instance in $\mathcal{R}$ |
| $\mathbf{x}$ | An instance in $\mathcal{R}^u$ |
| $D$ | A data set of $\mathbf{x}$, where $\|D\| = n$ |
| $\mathcal{D}$ | A subset of $D$, where $\|\mathcal{D}\| = \psi$ |
| $\mathbf{z}$ | An instance in $\mathcal{R}^t$ |
| $D'$ | A data set of $\mathbf{z}$ |
| $h$ | Level of mass distribution |
| $t$ | Number of mass distributions in $\widetilde{\mathbf{mass}}(.)$ |
| $m_i(.)$ | Mass base function defined using binary split $s_i$ |
| $mass(.)$ | Mass function which returns a real value |
| $\widetilde{\mathbf{mass}}(.)$ | Mass function which returns a vector of $t$ values |

Second, mass estimation offers to solve a problem more efficiently using the ordering derived from data directly—without distance or related expensive calculation—when the problem demands ranking. An example of inefficient application is in anomaly detection tasks where many methods have employed distance or density—a computationally expensive process—to provide the required ranking. An existing state-of-the-art density-based anomaly detector `LOF` [4] (which has quadratic time complexity) cannot complete a job involving half a million data points in less than two weeks; yet the mass-based anomaly detector we have introduced here completes it in less than 40 seconds! Section 4.3 provides the detail of this example.

Section 2 introduces mass and mass estimation, together with their theoretical properties. We also describe an efficient method to estimate mass in practice. Section 3 describes a mass-based formalism which serves as a basis of applying mass to different data mining tasks. We present a realisation of the formalism in three different tasks: information retrieval, regression and anomaly detection, and report the empirical evaluation results in Section 4. The relation to kernel density estimation is given in Section 5. We provide related work, the conclusions and future work in the last two sections.

# 2. MASS AND MASS ESTIMATION

**Data mass** or **mass** is defined as the number of points in a region; and two groups of data can have the same mass regardless of the characteristics of the regions (e.g., density, shape or volume). Mass in a given region is defined by a rectangular function which has the same value for the entire region in which the mass is measured.

Identifying a region occupied by a group of data in itself is a clustering problem, but mass can nonetheless be estimated without clustering. We show in this section that mass can be estimated in a way similar to kernel density estimation without involving clustering at all by using a function similar to a kernel function.

Note that **mass is not a probability mass function, and it does not provide probability**, as probability density function does through integration.

The detail of mass estimation is provided in the following two subsections. In Section 2.1, we show how to estimate a mass distribution for a given data set, and the theoretical properties of mass estimation. Section 2.2 describes an approximation to the theoretical mass estimation which works more efficiently in practice. This paper focuses on one-dimensional mass distribution only. The symbols and notations used are provided in Table 1.

## 2.1 Mass distribution estimation

We first show level-1 mass distribution estimation in Section 2.1.1. We then generalise the treatment for high level mass estimation in Section 2.1.2.

### 2.1.1 Level-1 mass distribution estimation

Here, we employ a binary split to divide the data set into two separate regions and compute the mass in each region. The mass distribution at point $x$ is estimated to be the sum of all 'weighted' masses from regions occupied by $x$, as a result of $n-1$ binary splits for a set of data of size $n$.

Let $x_1 < x_2 < \cdots < x_{n-1} < x_n$ on the real line[1], $x_i \in \mathcal{R}$ and $n > 1$. Let $s_i$ be the binary split between $x_i$ and $x_{i+1}$, yielding two non-empty regions having two masses $m_i^L$ and $m_i^R$.

**Definition 1.** *Mass base function: $m_i(x)$ as a result of $s_i$, is defined as*

$$m_i(x) = \begin{cases} m_i^L & \text{if } x \text{ is on the left of } s_i \\ m_i^R & \text{if } x \text{ is on the right of } s_i \end{cases}$$

*Note that $m_i^L = n - m_i^R = i$.*

**Definition 2.** *Mass distribution: $mass(x_a)$ for a point $x_a \in \{x_1, x_2, \cdots, x_{n-1}, x_n\}$ is defined as a summation of a series of mass base function $m_i(x)$ weighted by $p(s_i)$ over $n-1$ splits as follows.*

$$\begin{aligned} mass(x_a) &= \sum_{i=1}^{n-1} m_i(x_a) p(s_i) \\ &= \sum_{i=a}^{n-1} m_i^L p(s_i) + \sum_{j=1}^{a-1} m_j^R p(s_j) \\ &= \sum_{i=a}^{n-1} i p(s_i) + \sum_{j=1}^{a-1} (n-j) p(s_j) \end{aligned} \quad (1)$$

$p(s_i)$ is the probability of selecting $s_i$. Note that we have defined $\sum_{i=q}^{r} f(i) = 0$, when $r < q$ for any function $f$.

**Example.** For an example of five points $x_1 < x_2 < x_3 < x_4 < x_5$, Figure 1 shows the resultant $m_i(x)$ due to each of the four binary splits $s_1, s_2, s_3, s_4$; and their associated masses over four splits are given below:

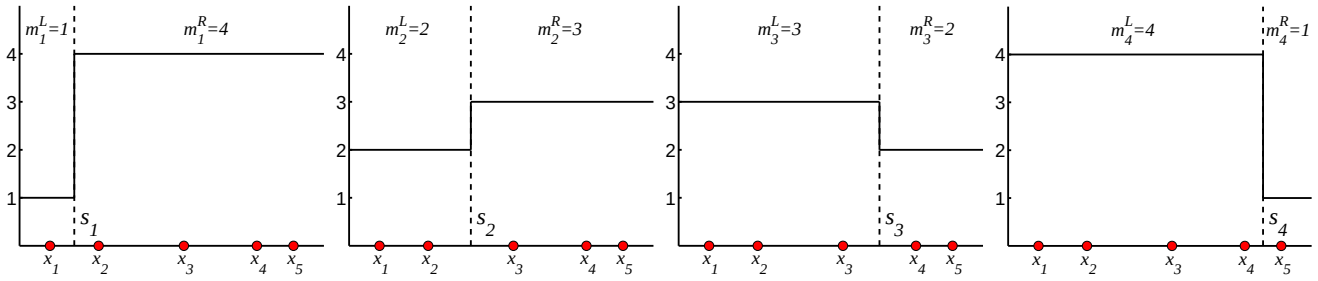
$$\begin{aligned} mass(x_1) &= 1p(s_1) + 2p(s_2) + 3p(s_3) + 4p(s_4) \\ mass(x_2) &= 4p(s_1) + 2p(s_2) + 3p(s_3) + 4p(s_4) \\ mass(x_3) &= 4p(s_1) + 3p(s_2) + 3p(s_3) + 4p(s_4) \\ mass(x_4) &= 4p(s_1) + 3p(s_2) + 2p(s_3) + 4p(s_4) \\ mass(x_5) &= 4p(s_1) + 3p(s_2) + 2p(s_3) + 1p(s_4) \end{aligned}$$

For a given data set, $p(s_i)$ can be estimated on the real line as $p(s_i) = (x_{i+1} - x_i)/(x_n - x_1) > 0$, as a result of random selection of splits based on a uniform distribution[2].

For a point $x \notin \{x_1, x_2, \cdots, x_{n-1}, x_n\}$, $mass(x)$ is defined as an interpolation between two masses of adjacent points $x_i$ and $x_{i+1}$, where $x_i < x < x_{i+1}$.

[1] In data having a pocket of points of the same value, an arbitrary order can be 'forced' by adding multiples of an insignificant small value $\epsilon$ to each point of the pocket, without changing the general distribution.

[2] The estimated $mass(x)$ values can be calibrated to a finite data range $\Delta$ by multiplying a factor $(x_n - x_1)/\Delta$.




(a) $m_i(x)$ due to $s_1$ (b) $m_i(x)$ due to $s_2$ (c) $m_i(x)$ due to $s_3$ (d) $m_i(x)$ due to $s_4$

**Figure 1: Examples of mass base function $m_i(x)$ due to each of the four binary splits: $s_1, s_2, s_3, s_4$.**

**Theorem 1.** *$mass(x_a)$ is the maximum at $a = n/2$ for any density distribution of $\{x_1, \cdots, x_n\}$; and the points $x_a$, where $x_1 < x_2 < \cdots < x_{n-1} < x_n$ on the real line, can be ordered based on mass as follows.*

$$mass(x_a) < mass(x_{a+1}),\ a < n/2$$
$$mass(x_a) > mass(x_{a+1}),\ a > n/2$$

PROOF. The difference in mass between two subsequent points $x_a$ and $x_{a+1}$ differs in only one term, i.e., the mass for $p(s_a)$ only; and $\forall i \neq a$, the terms for $p(s_i)$ have the same mass.

$$\begin{aligned} & mass(x_a) - mass(x_{a+1}) \\ = & \sum_{i=a}^{n-1} ip(s_i) + \sum_{j=1}^{a-1}(n-j)p(s_j) \\ & - \sum_{i=(a+1)}^{n-1} ip(s_i) - \sum_{j=1}^{a}(n-j)p(s_j) \\ = & ap(s_a) - (n-a)p(s_a) \\ = & (2a-n)p(s_a) \end{aligned} \quad (2)$$

Thus,

$$sign(mass(x_a) - mass(x_{a+1})) = \begin{cases} negative & \text{if } a < n/2 \\ 0 & \text{if } a = n/2 \\ positive & \text{if } a > n/2 \end{cases}$$

□

The point $x_{n/2}$ can be regarded as the median. Note that the number of points with the maximum mass depends on whether $n$ is odd or even: When $n$ is an odd integer, only one point has the maximum mass at $x_{median}$, where $median = \lceil n/2 \rceil$; when $n$ is an even integer, two points have the maximum mass at $a = n/2$ and $a = 1 + n/2$.

**Theorem 2.** *$mass(x_a)$ is a concave function defined w.r.t. $\{x_1, x_2, \ldots, x_n\}$, when $p(s_i) = (x_{i+1} - x_i)/(x_n - x_1)$.*

PROOF. We only need to show that the gradient of $mass(x_a)$ is non-increasing, i.e., $g(x_a) > g(x_{a+1})$ for each $a$.

Let $g(x_a)$ the gradient between $x_a$ and $x_{a+1}$, and from (2):

$$g(x_a) = \frac{mass(x_{a+1}) - mass(x_a)}{x_{a+1} - x_a} = \frac{n-2a}{x_n - x_1}$$

The result follows: $g(x_a) > g(x_{a+1})$ for $a \in \{1, 2, \ldots, n-1\}$.

□

**Corollary 1.** *A mass distribution estimated using binary splits stipulates an ordering, based on mass, of the points in a data cloud from $x_{n/2}$ (with the maximum mass) to the fringe points (with the minimum mass at either side of $x_{n/2}$), irrespective of the density distribution including uniform density distribution.*

**Corollary 2.** *The concavity of mass distribution stipulates that fringe points have markedly smaller mass than points close to $x_{n/2}$.*

The implication from Corollary 2 is that fringe points are 'stretched' to be farther away from the median in a mass space than in the original space—making it easier to separate fringe points from those points close to the median. (The mass space is mapped from the original space through $mass(x)$.) This property, otherwise hidden, can then be exploited by a data mining algorithm to achieve a better result for the intended task than the one without it. We will show that this simple mapping significantly improves the performance of four existing algorithms in information retrieval and regression tasks in Sections 4.1 and 4.2.

Equation (1) is sufficient to provide a mass distribution corresponds to a unimodal density function or a uniform density function. To better estimate multi-modal distributions, a high level mass distribution is required. This is provided in the following.

### 2.1.2 Level-$h$ mass distribution estimation

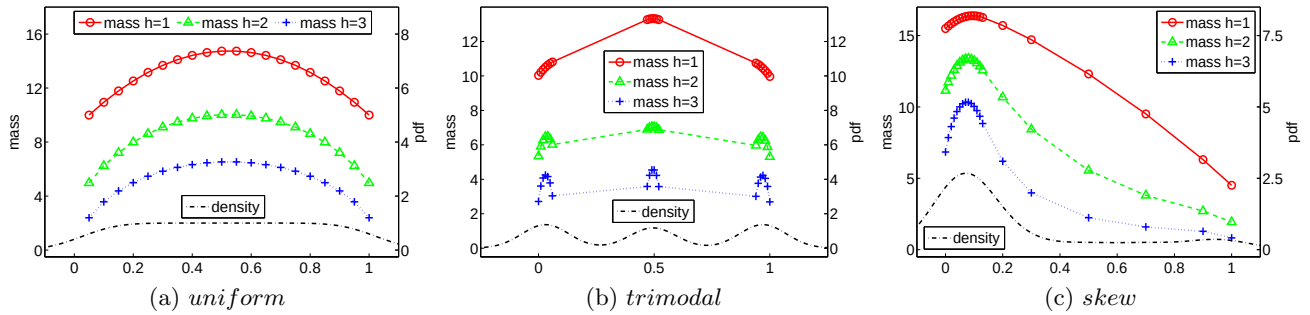
**Definition 3.** *Level-$h$ mass distribution for a point $x_a \in \{x_1, \ldots, x_n\}$, where $h < n$, is expressed as*

$$\begin{aligned} mass(x_a, h) &= \sum_{i=1}^{n-1} mass_i(x_a, h\text{-}1)p(s_i) \\ &= \sum_{i=a}^{n-1} mass_i^L(x_a, h\text{-}1)p(s_i) + \\ &\quad \sum_{j=1}^{a-1} mass_j^R(x_a, h\text{-}1)p(s_j) \end{aligned} \quad (3)$$

Here a high level mass distribution is computed recursively by using the mass distributions obtained at lower levels. A binary split $s_i$ in a level-$h$($>$1) mass distribution produces two level-($h$-1) mass distributions: (a) $mass_i^L(x, h\text{-}1)$—the mass distribution on the left of split $s_i$ which is defined using $\{x_1, \ldots, x_i\}$; and (b) $mass_i^R(x, h\text{-}1)$—the mass distribution on the right which is defined using $\{x_{i+1}, \ldots, x_n\}$. Equation (1) is the mass distribution at level-1.

Figure 2 shows part of the intermediate process in calculating $mass_i^L(x, h = 1)$ and $mass_i^R(x, h = 1)$ for two example splits $s_{i=7}$ and $s_{i=11}$ in order to obtain $mass(x, h = 2)$.

Using the same analysis in the proof for Theorem 1, the above equation can be re-expressed as:




(a) *uniform* (b) *trimodal* (c) *skew*

**Figure 3: Examples of level-$h$ mass distribution for $h = 1, 2, 3$ and density distribution from kernel density estimation: Gaussian kernel with bandwidth= 0.1. All three data sets have 20 points each.**

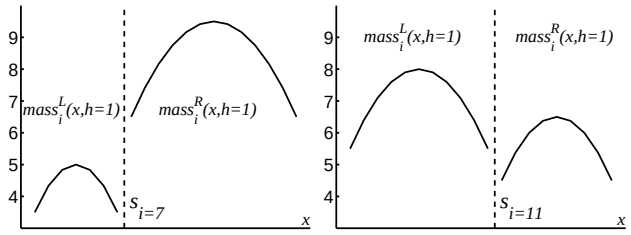



**Figure 2: Two examples of $mass_i^L(x, h = 1)$ and $mass_i^R(x, h = 1)$ due to $s_{i=7}$ and $s_{i=11}$ in the process to get $mass(x, h = 2)$ from a data set of 20 points with uniform density distribution. The resultant $mass(x, h = 2)$ is shown in Figure 3(a).**

$$mass(x_{a+1}, h) = mass(x_a, h) + \begin{cases} [mass_a^R(x_a, h\text{-}1) - mass_a^L(x_a, h\text{-}1)]p(s_a), & h > 1 \\ (n - 2a)p(s_a), & h = 1 \end{cases} \quad (4)$$

As a result, only the mass for the first point $x_1$ needs to be computed using Equation (3). Note that it is more efficient to compute the mass distribution from the above equation which has time complexity $O(n^{h+1})$; the computation using Equation (3) has $O(n^{h+2})$.

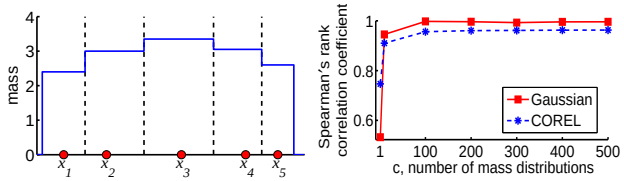
**Definition 4.** *A level-$h$ mass distribution stipulates an ordering of the points in a data cloud from $\alpha$-core points to the fringe points. The $\alpha$-core point(s) of a data cloud have the highest mass value within $\alpha$ distance from the core point(s). A small $\alpha$ defines local core point(s); and a large $\alpha$, which covers the entire value range for $x$, defines global core point(s).*

Examples of level-$h$ mass estimation in comparison with kernel density estimation are provided in Figure 3. Note that $h = 1$ mass estimation treats the entire data as a group, and it produces a concave function. As a result, an $h = 1$ mass estimation always has its global core point(s) at the median, regardless of the underlying density distribution—see three examples of $h = 1$ mass estimation in Figure 3.

For $h > 1$ mass distribution, though there is no guarantee for a single concave function for the entire data set, each cluster within the data cloud still exhibits a concave function and it becomes more distinct (as a concave function) as $h$ increases. This is shown in Figure 3(b) which has a trimodal density distribution. Notice that the $h > 1$ mass distributions have three $\alpha$-core points for some $\alpha$, e.g., 0.2.

Traditionally, one can determine the core-ness or the fringe-ness of a non-uniformly distributed data to some degree by using density or distance (but not in uniform density distribution.) Mass allows one to do that in any distributions without density or distance calculation—the key computational expense in all methods that employ them. For example in Figure 3(c) which has a skew density distribution, the distinction between near fringe points and far fringe points are less obvious using density, unless distances are computed to reveal the difference. In contrast, mass distribution depicts the relative distance from $x_{median}$ using the fringe points' mass values, without further calculation.

This section has described properties of mass distribution from a theoretical perspective. Though it is possible to estimate mass distribution using Equations (1) and (3), they are limited by its high computational cost. We suggest a practical mass estimation method in the next section. We use the term 'mass estimation' and 'mass distribution estimation' interchangeably hereafter.




(a) (b)

**Figure 4: (a) An example of practical mass distribution $mass(x, h|\mathcal{D})$ for 5 points, assuming a rectangular function for each point. (b) Correlation between the orderings provided by $mass(x, 1)$ and $\overline{mass}(x, 1)$ for two data sets: one-dimensional Gaussian density distribution and the COREL data set used in Section 4.1 (whose result is averaged over 67 dimensions).**

## 2.2 Practical mass estimation

Here we devise an approximation to Equation (3) using random subsamples from the given data set.

**Definition 5.** *$mass(x, h|\mathcal{D})$ is the approximate mass distribution for a point $x \in \mathcal{R}$, defined w.r.t. $\mathcal{D} = \{x_1, \ldots, x_\psi\}$, where $\mathcal{D}$ is a random subset of the given data set $D$, and $\psi \ll |D|$, $h < \psi$.*

Assume a rectangular function for each point $x \in \mathcal{D}$ (as shown in Figure 4(a)), $mass(x, h|\mathcal{D})$ is implemented using a lookup table with each rectangle function covers a range $(x_{i-1} + x_i)/2 \leq x < (x_{i+1} + x_i)/2$ for each point $x_i \in \mathcal{D}$ having the same $mass(x_i, h|\mathcal{D})$ value. The range for each of the two end-points is set to have equal length on either side of the point. In addition, a number of mass distributions

needs to be constructed from different samples in order to have a good approximation, that is,

$$\overline{mass}(x,h) = \frac{1}{c}\sum_{k=1}^{c} mass(x,h|\mathcal{D}_k) \quad (5)$$

The computation of $mass(x,h)$ using the given data set $D$ costs $O(|D|^{h+1})$; whereas $\overline{mass}(x,h)$ costs $O(c\psi^{h+1})$.

Only relative, not absolute, mass is required to provide an ordering between instances. Because the relative mass is w.r.t. median and median is a robust estimator [1]—that is why small subsamples produce a good order estimator.

Figure 4(b) shows the correlation (in terms of Spearman's rank correlation coefficient) between the orderings provided by $mass(x,1)$ using the entire data set and $\overline{mass}(x,1)$ using $\psi = 8$ in two data sets, each having 10000 data points. They achieve very high correlations when $c \geq 100$.

The ability to use a small sample, rather than a large sample, is a key characteristic of mass estimation. We show in this paper that $mass(x,h|\mathcal{D})$ can be employed very effectively for three different tasks: information retrieval, regression and anomaly detection, through a mass-based formalism to be described in the next section. Although the mass estimation is designed for one dimension only, we show that it can be employed to solve multi-dimensional problems.

## 3. MASS-BASED FORMALISM

Let $\mathbf{x}_i = [x_i^1, \ldots, x_i^u]$; $\mathbf{x}_i \in D$; and $\mathbf{z}_i = [z_i^1, \ldots, z_i^t]$; $\mathbf{z}_i \in D'$. The proposed formalism consists of three components:

**C1** The first component constructs a number of mass distributions. A mass distribution $mass(x^d,h|\mathcal{D})$ for dimension $d$ is obtained using our proposed mass estimation, as given in Definition 5. A total number of $t$ mass distributions is generated which forms $\widetilde{\mathbf{mass}}(\mathbf{x}) \rightarrow \mathcal{R}^t$, where $t \gg u$. This procedure is given in Algorithm 1.

**C2** The second component maps the data set $D$ in the original space of $u$ dimensions into a new data set $D'$ of $t$ dimensions using $\widetilde{\mathbf{mass}}(\mathbf{x}) = \mathbf{z}$. This procedure is described in Algorithm 2.

**C3** The third component employs a decision rule to determine the final outcome for the task at hand. It is a task-specific decision function applied to $\mathbf{z}$ in the new feature space.

**Algorithm 1** : Mass_Estimation$(D,\psi,h,t)$

**Inputs**: $D$ - input data; $\psi$ - data size for $\mathcal{D}_k$; $h$ - level of mass distribution; $t$ - number of mass distributions.
**Output**: $\widetilde{\mathbf{mass}}(\mathbf{x}) \rightarrow \mathcal{R}^t$ - a function consists of $t$ mass distributions, $mass(x^d,h|\mathcal{D}_k)$.

1: **for** $k = 1$ to $t$ **do**
2: $\quad\mathcal{D}_k \leftarrow$ a random subset of size $\psi$ from $D$;
3: $\quad d \leftarrow$ a randomly selected dimension from { 1,. . . ,u };
4: $\quad$Build $mass(x^d,h|\mathcal{D}_k)$;
5: **end for**

The formalism becomes a blueprint for different tasks. Components **C1** and **C3** are mandatory in the formalism, but component **C2** is optional, depending on the task.

For information retrieval and regression, the task-specific **C3** procedure is simply using an existing algorithm for the task except that the process is carried out in the new mapped mass space, instead of the original space. This procedure is given in Algorithm 3. The task-specific **C3** procedure for anomaly detection is shown in steps 2-5 in Algorithm 4. Note that anomaly detection requires **C1** and **C3** only; whereas the other two tasks require all three components.

**Algorithm 2** : Mass_Mapping$(D,\widetilde{\mathbf{mass}})$

**Inputs**: $D$ - input data; $\widetilde{\mathbf{mass}}$ - a function consists of $t$ mass distributions, $mass(x^d,h|\mathcal{D})$.
**Output**: $D'$ - a set of mapped instances $\mathbf{z}_i$ in $t$ dimensions.

1: **for** $i = 1$ to $|D|$ **do**
2: $\quad\mathbf{z}_i \leftarrow \widetilde{\mathbf{mass}}(\mathbf{x}_i)$;
3: **end for**

**Algorithm 3** : Perform task in MassSpace$(D,\psi,h,t)$

**Inputs**: $D$ - input data; $\psi$ - data size for $\mathcal{D}$; $h$ - level of mass distribution; $t$ - number of mass distributions.
**Output**: Task-specific model.

1: $\widetilde{\mathbf{mass}}(.) \leftarrow$ Mass_Estimation$(D,\psi,h,t)$;
2: $D' \leftarrow$ Mass_Mapping$(D,\widetilde{\mathbf{mass}})$;
3: Perform task (information retrieval or regression) in the mapped mass space using $D'$;

**Algorithm 4** for Anomaly Detection : MassAD$(D,\psi,h,t)$

**Inputs**: $D$ - input data; $\psi$ - data size for $\mathcal{D}$; $h$ - level of mass distribution; $t$ - number of mass distributions.
**Output**: Ranked instances in $D$.

1: $\widetilde{\mathbf{mass}}(.) \leftarrow$ Mass_Estimation$(D,\psi,h,t)$;
2: **for** $i = 1$ to $|D|$ **do**
3: $\quad\mathsf{m}_i \leftarrow$ Average of $t$ masses from $\widetilde{\mathbf{mass}}(\mathbf{x}_i)$;
4: **end for**
5: Rank instances in $D$ based on $\mathsf{m}_i$ with low mass denotes anomalies and high mass denotes normal points;

## 4. EXPERIMENTS

We evaluate the performance of MassSpace and MassAD for three tasks in the following three subsections. In information retrieval and regression tasks, the mass estimation uses $\psi = 8$ and $t = 1000$. These settings are obtained by examining the rank correlation as shown in Figure 4(b)—having a high rank correlation between $mass(x,1)$ and $\overline{mass}(x,1)$. Note that this is done before any method is applied and no further fine-tuning. In anomaly detection tasks, $\psi = 256$ and $t = 100$ are used so that they are comparable to those used in a benchmark method for a fair comparison. $h = 1$ is used in all tasks, unless stated otherwise. All the experiments are run in Matlab and conducted on a Pentium 4 machine with an AMD Opteron machine with a 1.8 GHz processor and 4 GB memory. The performance of each method is measured in terms of task-specific performance measure and runtime. Paired $t$-tests at 5% significance level are conducted to examine whether the difference in performance is significant between two algorithms under comparison.

Note that we treat information retrieval and anomaly detection as unsupervised learning tasks. Classes/labels in the original data are used as ground truth for evaluation of performance only; they are not used in building mass distributions. In regression, only the training set is used to build mass distributions in step 1 of Algorithm 3; the mapping in step 2 is conducted for both the training and testing sets.

**Table 2: CBIR results (the higher the better for BEP.)**

| | BEP ($\times 10^{-2}$) | | | | | | Processing time (second) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MRBIR′ | MRBIR | Qsim′ | Qsim | InstR′ | InstR | MRBIR′ | MRBIR | Qsim′ | Qsim | InstR′ | InstR |
| One query | **11.52** | 9.69 | **10.31** | 7.78 | **10.31** | 7.78 | 1.980 | **1.111** | 0.410 | **0.034** | 0.410 | **0.034** |
| Round 1 | **15.14** | 12.72 | **15.39** | 10.59 | **13.45** | 9.40 | 2.499 | **2.155** | 0.588 | **0.078** | 0.558 | **0.046** |
| Round 2 | **16.81** | 13.90 | **17.46** | 11.81 | **15.07** | 9.99 | 2.501 | **2.155** | 0.646 | **0.139** | 0.559 | **0.047** |
| Round 3 | **17.94** | 14.75 | **18.46** | 12.59 | **16.15** | 10.36 | 2.499 | **2.155** | 0.737 | **0.227** | 0.560 | **0.048** |
| Round 4 | **18.74** | 15.33 | **19.18** | 13.16 | **16.96** | 10.78 | 2.501 | **2.155** | 0.862 | **0.355** | 0.561 | **0.049** |
| Round 5 | **19.39** | 15.71 | **19.62** | 13.55 | **17.62** | 11.05 | 2.499 | **2.155** | 1.016 | **0.516** | 0.562 | **0.050** |

**Table 3: Regression results (the smaller the better for MSE; the larger the better for SCC.)**

| | data | | MSE ($\times 10^{-2}$) | | | SCC ($\times 10^{-2}$) | | | Processing time | | Factor increase | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | size | $u$ | SVR′ | SVR | W/D/L | SVR′ | SVR | W/D/L | SVR′ | SVR | time | dimension |
| tic | 9822 | 85 | **5.58** | 5.62 | 17/0/3 | **2.12** | 1.07 | 18/0/2 | 63.61 | **29.85** | 2.1 | 12 |
| wine_white | 4898 | 11 | **1.21** | 1.36 | 20/0/0 | **45.18** | 38.60 | 20/0/0 | 17.30 | **7.24** | 2.4 | 91 |
| quake | 2178 | 3 | **2.86** | 2.92 | 18/0/2 | **0.84** | 0.31 | 14/0/6 | 3.18 | **1.09** | 2.9 | 333 |
| wine_red | 1599 | 11 | 1.62 | 1.62 | 11/0/9 | **38.20** | 37.76 | 13/0/7 | 2.00 | **0.76** | 2.6 | 91 |
| concrete | 1030 | 8 | **0.33** | 0.57 | 20/0/0 | **92.62** | 87.17 | 20/0/0 | 1.08 | **0.44** | 2.5 | 125 |

## 4.1 Content-Based Image Retrieval

We use a Content-Based Image Retrieval (CBIR) task as an example of information retrieval. The MassSpace approach is compared with three state-of-the-art CBIR methods that deal with relevance feedbacks: a manifold based method MRBIR [9], and two recent techniques for improving similarity calculation, i.e., Qsim [19] and InstRank [8]; and we employ the Euclidean distance to measure the similarity between instances in these two methods. The default parameter settings are used for all these methods. Because the same CBIR method is employed in the mapped space in the MassSpace approach, we denote them as MRBIR′, Qsim′ and InstRank′ for those employ MRBIR, Qsim and InstRank, respectively.

Our experiments are conducted using the COREL image database [18] of 10000 images, which contains 100 categories and each category has 100 images. Each image is represented by a 67-dimensional feature vector, which consists of 11 shape, 24 texture and 32 color features. To test the performance, we randomly select 5 images from each category to serve as the initial queries. For a query, the images within the same category are regarded as relevant and the rest are irrelevant. For each query, we continue to perform 5 rounds of relevance feedback. In each round, 2 positive and 2 negative feedbacks are provided. This relevance feedback process is also repeated 5 times, each up to 5 feedback rounds. Finally, the average results with one query and in different feedback rounds are recorded. The retrieval performance is measured in terms of Break-Even-Point (BEP) [19, 18] of the precision-recall curve. The online processing time reported is the time required in each method for a query plus the stated feedback rounds. The reported result is an average over $5 \times 100$ runs for query only; and an average over $5 \times 100 \times 5$ runs for query plus feedbacks. The offline costs of constructing the mass distributions and the mapping of 10000 images are 2.87 and 1.25 seconds, respectively.

The results are presented in Table 2 where the best performance at each round has been boldfaced. The results are grouped in pairs for ease of comparison.

The BEP results clearly show that the MassSpace approach achieves a better retrieval performance than that using the original space in all three methods MRBIR, Qsim and InstR, regardless it is with one query only or in relevance feedbacks. Paired $t$-tests at 5% significance level also indicate that the MassSpace approach significantly outperforms each of the three methods in all experiments, without exception. These results show that the mass space provides useful additional information that is hidden in the original space.

The processing time for each of the three methods in the mass space is expected to be longer than that in the original space because the number of dimensions in the mass space is significantly higher than those in the original space, where $t = 1000$ and $u = 67$.

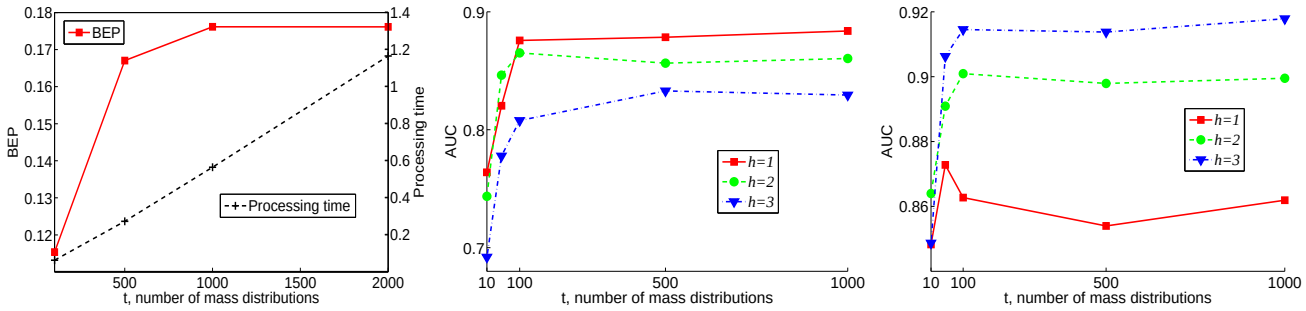
Figure 5(a) shows an example of performance for InstR′—BEP increases as $t$ increases until it reaches a plateau at some $t$ value; and the processing time for InstR′ is linear w.r.t. the number of dimensions of the mass space, $t$.

## 4.2 Regression

In this experiment, we compare SVR′ with SVR—support vector regression [16] that employs the mapped mass space versus that employs the original space. SVR is the $\epsilon$-SVR algorithm with RBF kernel, implemented by LIBSVM [6]. SVR is chosen here because it is one of the top performing regression models.

We utilize five benchmark data sets including four selected from UCI repository [2] and one earthquake data [14] from www.cs.waikato.ac.nz/ml/weka/ distribution. The data characteristics are summarized in the first three columns of Table 3. We select only those data sets which are more than 1000 data points with all real-valued attributes and without missing values—in order to get a result with a higher confidence than those obtained from small data sets.

On each data set, we randomly sample two-thirds of the instances for training and the remaining one-third for testing. This is repeated 20 times and we report the average result of these 20 runs. The data set, whether in the original space or the mass space, is min-max normalized before an $\epsilon$-SVR model is trained. To select optimal parameters for the $\epsilon$-SVR algorithm, we conduct a 5-fold cross validation based on mean squared error using the training set only. The kernel parameter $\gamma$ is searched in the range $\{2^{-15}, 2^{-13}, 2^{-11}, \cdots, 2^3, 2^5\}$; the regularization parameter




(a) An example of CBIR-round 5 result. (b) Effect of $h$ in the Forest data set. (c) Effect of $h$ in the Smtp data set.

**Figure 5: (a) The retrieval performance and the processing time as $t$ increases for InstR′. (b) High $h$ produces a poorer detection performance in this case. (c) High $h$ produces a better detection performance in this case.**

**Table 4: Data characteristics of the data sets in anomaly detection tasks. The percentage in brackets indicates the percentage of anomalies.**

| | data size | $u$ | anomaly class |
|---|---|---|---|
| Http | 567497 | 3 | attack (0.4%) |
| Forest | 286048 | 10 | class 4 (0.9%) vs class 2 |
| Mulcross | 262144 | 4 | 2 clusters (10%) |
| Smtp | 95156 | 3 | attack (0.03%) |
| Shuttle | 49097 | 9 | classes 2,3,5,6,7 (7%) vs class 1 |

$C$ in the range $\{0.1, 1, 10\}$, and $\epsilon$ in the range $\{0.01, 0.05, 0.1\}$. We measure regression performance in terms of mean squared error (MSE) and squared correlation coefficient (SCC), and runtime in seconds. The runtime reported is the runtime for SVR only. The total cost of mass estimation (from the training set) and mapping (of training and testing sets) is 3.95 seconds in the largest data set, tic. The cost of normalisation and the parameter search using 5-fold cross-validation is not included in the reported result for both SVR′ and SVR.

The result is presented in Table 3. SVR′ performs significantly better than SVR in all data sets in both MSE and SCC measures; the only exception is in the wine_red data set. Although SVR′ takes more time to run as it runs on the data with a significantly higher dimension, yet the factor of increase in time ranges from 2 to 3 only when the factor of increase in the number of dimensions ranges from 12 to over 300 (shown in the last two columns of Table 3). This is because the time complexity in the key optimisation process in SVR is not dependent on the number of dimensions.

## 4.3 Anomaly Detection

This experiment compares MassAD with four state-of-the-art anomaly detectors: isolation forest (or iForest) [10], a distance-based method ORCA [3], a density-based method LOF [4], and one-class support vector machine (or 1-SVM) [13]. MassAD is built with $t = 100$ and $\psi = 256$, the same default settings as used in iForest [10], which also employs a multi-model approach. The parameter settings employed for ORCA and LOF are as stated in [10]. 1-SVM uses Radial Basis Function kernel and an inverse width parameter estimated by the method suggested in [5].

All the methods are tested on the five largest data sets used in [10]. The data characteristics are summarized in Table 4, which include one anomaly data generator Mulcross [12] and the other four are from UCI repository [2]. The performance is evaluated in terms of averaged AUC (area under ROC curve) and processing time (a total of training time and testing time) over ten runs (following [10]). MassAD and iForest are implemented in Matlab and tested on an AMD Opteron machine with a 1.8 GHz processor and 4 GB memory. The results for ORCA, LOF and 1-SVM are conducted using the same experimental setting but on a faster 2.3 GHz machine, the same machine used in [10].

**Table 5: AUC values for anomaly detection.**

| | MassAD | | iForest | ORCA | LOF | 1-SVM |
|---|---|---|---|---|---|---|
| | $\psi$=8 | $\psi$=256 | | | | |
| Http | 0.98 | **1.00** | **1.00** | 0.36 | N/A | 0.90 |
| Forest | 0.88 | **0.91** | 0.87 | 0.83 | 0.57 | 0.90 |
| Mulcross | 0.97 | **0.99** | 0.96 | 0.33 | 0.59 | 0.59 |
| Smtp | 0.86 | 0.86 | **0.88** | 0.87 | 0.32 | 0.78 |
| Shuttle | 0.99 | 0.99 | **1.00** | 0.60 | 0.55 | 0.79 |

**Table 6: Runtime (second) for anomaly detection.**

| | MassAD | | iForest | ORCA | LOF | 1-SVM |
|---|---|---|---|---|---|---|
| | $\psi$=8 | $\psi$=256 | | | | |
| Http | **27** | 34 | 147 | 9487 | > 2weeks | 35872 |
| Forest | **14** | 18 | 79 | 6995 | 224380 | 9738 |
| Mulcross | **13** | 17 | 75 | 2512 | 156044 | 7343 |
| Smtp | **4** | 7 | 26 | 267 | 24281 | 987 |
| Shuttle | **2** | 4 | 15 | 157 | 7490 | 333 |

The AUC values of all methods are presented in Table 5 where the figures boldfaced are the best performance for each data set. The results show that MassAD with $\psi = 256$ achieves the best performance on the three largest data sets; and even on the other two data sets, MassAD is also competitive since the AUC gap is small between MassAD and the best method, i.e., iForest. It is noteworthy that MassAD significantly outperforms the traditional density-based, distance-based and SVM anomaly detectors in all data sets, except two: one in Smtp when compared with ORCA and another in Forest when compared with 1-SVM. The above observations validate the effectiveness of our proposed mass estimation on anomaly detection tasks.

Table 6 shows the runtime result. Although MassAD is run on a slower machine, it still has a significant advantage in term of processing time over ORCA, LOF and 1-SVM. The comparison with iForest is presented in Table 7 with a breakdown of training time and testing time. Note that MassAD takes the same time as iForest in training, but it only takes about one-tenth of the time required by iForest in testing. These results show that MassAD is an efficient anomaly detector.

Figures 5(b) and 5(c) show the effect of $h$ on the detection

**Table 7: Training time and testing time (second) for MassAD and iForest, using $t = 100$ and $\psi = 256$.**

| | Training time | | Testing time | |
|---|---|---|---|---|
| | MassAD | iForest | MassAD | iForest |
| Http | 20.96 | 19.72 | 12.93 | 127.47 |
| Forest | 11.26 | 11.47 | 6.97 | 67.45 |
| Mulcross | 10.54 | 10.69 | 6.82 | 64.34 |
| Smtp | 4.97 | 4.1 | 2.22 | 22.39 |
| Shuttle | 3.43 | 3.23 | 1.01 | 11.79 |

performance of MassAD with $\psi = 8$—higher $h$ degrades the detection performance in Forest; but it improves in Smtp. This shows that for best performance in individual data set, some parameter tuning is required, like most other algorithms. Note that there is no attempt to tune this parameter (or any other parameters) in the result reported in Tables 5, 6 and 7 where $h = 1$ is used throughout.

The time and space complexities for four methods are given in Table 8. MassAD and iForest have the best time and space complexities due to their ability to use small $\psi \ll n$ and $h = 1$. Note that MassAD ($h = 1$) is faster by a factor of $log(\psi = 256) = 8$ which shows up in the testing time—ten times faster than iForest given in Table 7. The training time disadvantage, compared to iForest, did not show up because of small $\psi$. MassAD also has an advantage over iForest in space complexity by a factor of $log(\psi)$.

**Table 8: A comparison of time and space complexities. The time complexity includes both training and testing. $n$ is the given data set size and $u$ is the number of dimensions. For MassAD and iForest, the first part of the summation is the training time and the second the testing time.**

| | Time complexity | Space complexity |
|---|---|---|
| MassAD | $O(t(\psi^{h+1} + n))$ | $O(t\psi)$ |
| iForest | $O(t(\psi + n) \cdot log(\psi))$ | $O(t\psi \cdot log(\psi))$ |
| ORCA | $O(un \cdot log(n))$ | $O(un)$ |
| LOF | $O(un^2)$ | $O(un)$ |

## 4.4 Constant time and space complexities

In this section, we show that $mass(x, h|\mathcal{D})$ (in step 4 of Algorithm 1) takes only constant time, regardless of the given data size $n$, when the algorithmic parameters are fixed. Table 9 reports the runtime time for sampling (to get a random sample of size $\psi$ from the given data set—steps 2 and 3 of Algorithm 1) and the runtime for mass estimation—to construct $mass(x, h|\mathcal{D})$ $t$ times, for five data sets which include the largest and smallest data sets in regression and anomaly detection tasks.

The results show that the sampling time increases linearly with the size of the given data set, and it takes a significantly longer (in the largest data set) than the time to construct the mass distribution—which is constant, regardless of the given data size. Note that the training time provided in Table 7 includes both the sampling time and mass estimation time, and it is dominated by the sampling time.

The memory required for each construction of $mass(x, h|\mathcal{D})$ is to store one lookup table of size $\psi$ which is constant, again independent of the given data size.

**Table 9: Runtime (second) for sampling, $mass(x, 1|\mathcal{D})$ and $mass(x, 3|\mathcal{D})$, where $t = 1000$ and $\psi = 8$.**

| | data size | sampling | $mass(x, 1\|\mathcal{D})$ | $mass(x, 3\|\mathcal{D})$ |
|---|---|---|---|---|
| Http | 567497 | 185.21 | 0.57 | 17.15 |
| Shuttle | 49097 | 12.47 | 0.59 | 17.37 |
| COREL | 10000 | 2.34 | 0.53 | 17.28 |
| tic | 9822 | 2.28 | 0.56 | 17.23 |
| concrete | 1030 | 0.36 | 0.48 | 17.28 |

## Summary

The above results in all three tasks show that the orderings provided by mass distributions deliver additional information about the data that would otherwise hidden in the original features. The additional information improves the task-specific performance significantly, especially in the information retrieval and regression tasks.

Using Algorithm 3, the runtime is expected to be higher because the new space has much higher dimensions than the original space ($t \gg u$). It shall be noted that the runtime increase (linearly or worse) is solely a characteristic of the existing algorithms used, not due to the mass space mapping which has constant time and space complexities.

We believe that a more tailored approach that better integrates the information provided by mass (into the **C3** component in the formalism) for the specific task can potentially further improve the current level of performance in terms of either task-specific performance measure or runtime. We have demonstrated this 'direct' application using Algorithm 4 for the anomaly detection task, in which MassAD performs equally well or significantly better than four state-of-the-art methods in terms of task-specific performance measure, and it executes faster than all other methods in terms of runtime.

Why does one-dimensional mapping work when tackling multi-dimensional problems? The mapping transforms each original feature to approximately $\frac{t}{u}$ features in the mass space—unearth hidden information for each original feature. It is more of a question whether an algorithm can make full use of this information in the new space; as both the original and new spaces are multi-dimensional. A multi-dimensional mapping may better enhance information in some domains. It is thus worthwhile to explore this extension.

# 5. RELATION TO KERNEL DENSITY ESTIMATION

A comparison of mass estimation and kernel density estimation is provided in Table 10.

**Table 10: A comparison of kernel density estimation and mass estimation. Kernel density estimation requires two parameter settings: kernel function $K(.)$ and bandwidth $h_w$; mass estimation has one: $h$.**

| |
|---|
| Kernel density$(x) = \frac{1}{nh_w} \sum_{i=1}^{n} K(\frac{x-x_i}{h_w})$ |
| $mass(x, h) = \begin{cases} \sum_{i=1}^{n-1} mass_i(x, h\text{-}1)p(s_i), & h > 1 \\ \sum_{i=1}^{n-1} m_i(x)p(s_i), & h = 1 \end{cases}$ |

Like kernel estimation, mass estimation at each point is computed through a summation of a series of values from a mass base function $m_i(.)$, equivalent to a kernel function $K(.)$. The two methods differ in the following ways:

Table 11: CBIR results: Compare with Qsim″ and InstR″ which use Gaussian kernel density estimation.

| | BEP ($\times 10^{-2}$) | | | | | | Processing time (second) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Qsim′ | Qsim″ | Qsim | InstR′ | InstR″ | InstR | Qsim′ | Qsim″ | Qsim | InstR′ | InstR″ | InstR |
| One Query | **10.31** | 2.51 | 7.78 | **10.31** | 2.51 | 7.78 | 0.410 | 0.409 | **0.034** | 0.410 | 0.409 | **0.034** |
| Round 1 | **15.39** | 2.72 | 10.59 | **13.45** | 2.66 | 9.40 | 0.588 | 0.633 | **0.078** | 0.558 | 0.571 | **0.046** |
| Round 2 | **17.46** | 2.67 | 11.81 | **15.07** | 2.51 | 9.99 | 0.646 | 0.780 | **0.139** | 0.559 | 0.574 | **0.047** |
| Round 3 | **18.46** | 2.56 | 12.59 | **16.15** | 2.31 | 10.36 | 0.737 | 0.989 | **0.227** | 0.560 | 0.577 | **0.048** |
| Round 4 | **19.18** | 2.53 | 13.16 | **16.96** | 2.20 | 10.78 | 0.862 | 1.275 | **0.355** | 0.561 | 0.580 | **0.049** |
| Round 5 | **19.62** | 2.46 | 13.55 | **17.62** | 2.07 | 11.05 | 1.016 | 1.629 | **0.516** | 0.562 | 0.582 | **0.050** |

Table 12: Anomaly detection: MassAD vs DensityAD.

| | AUC | | Time (second) | |
|---|---|---|---|---|
| | MassAD | DensityAD | MassAD | DensityAD |
| Http | 1.00 | 0.99 | 34 | 33 |
| Forest | **0.91** | 0.69 | 18 | 18 |
| Mulcross | 0.99 | 1.00 | 17 | 17 |
| Smtp | **0.86** | 0.60 | 7 | 7 |
| Shuttle | **0.99** | 0.92 | 4 | 4 |

- *Aim*: Kernel estimation is aimed to do probability density estimation; whereas mass estimation is to estimate an order from the core points to the fringe points.

- *Kernel function*: While kernel estimation can use different kernel functions for probability density estimation; we doubt that mass estimation requires a different base function for two reasons. First, a more sophisticated function is unlikely to provide a better ordering than a simple rectangular function. Second, the rectangular function keeps the computation simple and fast. In addition, a kernel function must be fixed (i.e., having user-defined values for its parameters); e.g., the rectangular kernel function has fixed width or fixed per unit size. But the rectangular function used in mass has no parameter and no fixed width.

- *Sample size*: Kernel estimation or other density estimation methods require a large sample size in order to estimate the probability accurately [7]. Mass estimation using $mass(x, h|\mathcal{D})$ needs only a small sample size in an ensemble to accurately estimate the ordering.

- *Definition*: Probability density can be defined independent of data, whereas mass (in its current form) must be defined w.r.t. a set of data.

Because of a lack of concavity, density will not perform as successfully as mass. Here we present the results using a Gaussian kernel density estimation, replacing $mass(x, h|\mathcal{D}_k)$, using the same subsample size in an ensemble approach. The bandwidth parameter is set to be the standard deviation of the subsample; and all the other parameters are the same.

The results for information retrieval and anomaly detection are provided in Tables 11 and 12. Compare to mass, density performs significantly worse in information retrieval task in all experiments using Qsim and InstR, denoted as Qsim″ and InstR″, respectively. They are even worse than those run in the original space. In anomaly detection, DensityAD, which uses a Gaussian kernel density estimation, performs significantly worse than MassAD in three out of five data sets in the anomaly detection tasks, and equally well in the other two data sets.

# 6. RELATED WORK

There is a close relationship between the proposed mass and data depth [11]: they both delineate the centrality of a data cloud (as opposed to compactness in the case of density.) The properties common to both measures are: (a) the centre of a data cloud has the maximum value of the measure; (b) an ordering from the centre (having the maximum value) to the fringe points (having the minimum values).

However, there are three fundamental differences. First, data depth can deal with unimodal data only; whereas mass can deal with both unimodal and multi-modal data by setting $h = 1$ or $h > 1$.

Second, mass is a simple and straightforward measure, and has an efficient estimation method; whereas data depth has many different definitions, depending on the construct used to define depth. The constructs could be Mahalanobis, Convex Hull, simplicial and so on [11], all of which are expensive to compute [1]—this has been the main obstacle in applying data depth for real applications in multi-dimensional problems. In addition, the centre of a data cloud varies depending on the construct used to define data depth; whereas mass ($h = 1$) always has the centre located at the mid-point in the series of data points.

Third, the $h = 1$ mass estimation guarantees concavity—the reason why a simple mass space mapping improves the task-specific performance of four existing algorithms in information retrieval and regression tasks. In contrast, there is no such guarantee in data depth. Because of a lack of concavity, like density, data depth is unlikely to be as successful as mass in the three tasks we have reported here, even if we ignore the runtime issue.

Mass estimation can be implemented in different ways. For example, we have reported an implementation using a tree structure (instead of a lookup table) in [15] using Half-Space Trees. It reduces the time complexity to $O(th(\psi+n))$ from $O(t(\psi^{h+1}+n))$, making it feasible for very high level-$h$ mass estimation. We have repeated the experiments reported in this paper using Half-Space Trees, and it produces almost identical results.

Half-Space Trees extends naturally from one-dimensional mass estimation to multi-dimensional mass estimation. This has been tested in anomaly detection task [15].

iForest [10] and MassAD shares some common features: Both are ensemble methods which build $t$ models, each from a random sample of size $\psi$, and they both combine the outputs of the models through averaging during testing. Although iForest [10] is designed specifically for anomaly detection which employs path length—an instance traverses from the root of a tree to its leaf—as the anomaly score, we have shown in [15] that the path length used in iForest is in fact a proxy to mass. In other words, iForest is a kind of

mass-based method—that is why `MassAD` and `iForest` have similar detection accuracy.

We have already established a direct application of mass in content-based image retrieval [17]. In addition to the mass-space mapping we have shown here, [17] presents a framework that assigns a weight (based on `iForest`, thus, mass) to each feature w.r.t. a query image; and then it ranks images in the database according to their weighted average feature values. The framework also incorporates relevance feedback which modifies the ranking based on the feedbacks through reweighted features. This framework makes use of all three components of the formalism stated in Section 3. This direct application of mass performs significantly better than the indirect approach we have shown in Section 4.1, in terms of both retrieval performance and processing time. Like `MassAD`, no distance calculations are used at all—the key reason for its superior time complexity.

# 7. CONCLUSIONS AND FUTURE WORK

This paper makes two key contributions. First, we introduce a base measure, mass, and delineate its three properties: (i) a mass distribution stipulates an ordering from core points to fringe points in a data cloud; (ii) this ordering accentuates the fringe points with a concave function—the essential property that is easily exploited by existing algorithms to improve their task-specific performance; and (iii) it is a constant-time-and-space-complexities estimation method. Density estimation has been the base modelling mechanism employed in many techniques thus far. Mass estimation introduced here provides an alternative choice, and it is better suited for many tasks which require an ordering rather than probability density estimation.

Second, we present a mass-based formalism which forms a basis to apply mass for different tasks. The three tasks (i.e., information retrieval, regression and anomaly detection) in which we have successfully applied are just examples of its application. Mass estimation has potentials in applications as diverse as density estimation has applied now.

There are potential extensions to the current work. First, one shall consider a new way to best utilise mass when solving a problem. In other words, we advocate a direct application of mass, rather than an indirect application. Second, the algorithms provided here for the three tasks are by no means definitive, and even the formalism can be improved or extended to include more tasks. Third, because the purposes and their properties differ, mass estimation is not intended to replace density estimation—it is thus important to identify areas in which each is best suited for. This will ascertain areas in which density has been a mismatch, unbeknown thus far.

## Acknowledgements

This work is partially supported by the Air Force Research Laboratory, under agreement# FA2386-10-1-4052. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Phil Rayment, David Albrecht, Zhouyu Fu and Geoff Webb have provided many helpful comments in the early draft. Suggestions from the anonymous reviewers have helped to improve the clarity of this paper.

The Matlab source code of mass estimation is available at http://sourceforge.net/projects/mass-estimation/.

# 8. REFERENCES

[1] G. Aloupis. Geometric measures of data depth. *DIMACS Series in Discrete Math and Theoretical Computer Science*, 72:147–158, 2006.

[2] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[3] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of SIGKDD*, pages 29–38, 2003.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of SIGKDD*, pages 93–104, 2000.

[5] B. Caputo, K. Sim, F. Furesjo, and A. Smola. Appearance-based object recognition using svms: which kernel should i use? In *NIPS workshop on Statitsical methods for computational experiments in visual processing and computer vision*, 2002.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001.

[7] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Second Edition. John Wiley, 2001.

[8] G. Giacinto and F. Roli. Instance-based relevance feedback for image retrieval. In *Advances in NIPS*, pages 489–496, 2005.

[9] J. He, M. Li, H. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *Proceedings of ACM Multimedia*, pages 9–16, 2004.

[10] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Proceedings of ICDM*, pages 413–422, 2008.

[11] R. Liu, J. M. Parelius, and K. Singh. Multivariate analysis by data depth. *The Annals of Statistics*, 27(3):783–840, 1999.

[12] D. M. Rocke and D. L. Woodruff. Identification of outliers in multivariate data. *Journal of the American Statistical Association*, 91(435):1047–1061, 1996.

[13] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Advances in NIPS*, pages 582–588, 2000.

[14] J. S. Simonoff. *Smoothing Methods in Statistics*. Springer-Verlag, 1996.

[15] K. M. Ting, S. C. Tan, and F. T. Liu. Mass: A new ranking measure for anomaly detection. *Gippsland School of Information Technology, Monash University*, Technical Report TR2009/1, 2009.

[16] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Second Edition. Springer, 2000.

[17] G.-T. Zhou, K. M. Ting, F. T. Liu, and Y. Yin. Relevance feature mapping for content-based image retrieval. In *Proceedings of Multimedia Data Mining Workshop at KDD*, 2010.

[18] Z.-H. Zhou, K.-J. Chen, and H.-B. Dai. Enhancing relevance feedback in image retrieval using unlabeled data. *ACM Transactions on Information Systems*, 24(2):219–244, 2006.

[19] Z.-H. Zhou and H.-B. Dai. Query-sensitive similarity measure for content-based image retrieval. In *Proceedings of ICDM*, pages 1211–1215, 2006.