XStream

1 Histogram Forests (HF)

Fitting. For each histogram $i = 1, \ldots, T$:

1. **Initialize the random projection matrix A_i.** This may be (i) sparse with 2/3 zeros, or (ii) Gaussian random projections (which demonstrate better performance, but are computationally expensive).
2. **Project the data.** $Y = XA_i^T$.
3. **Sample a shift.** $b_i \sim \text{Uniform}(0, w)$.
4. **Bin the data.** For each projected data point $Y_j = (y_{j1}, \ldots, y_{jK})$, its bin vector is given by $\bar{Y}_j = (\lfloor \frac{y_{j1}+b_i}{w} \rfloor, \ldots, \lfloor \frac{y_{jK}+b_i}{w} \rfloor)$.
5. **Store bin counts.** In an exact hashtable or count-min sketch H_i, increment the count $H_i(\bar{Y}_j)$ for $j = 1, \ldots, N$.

Scoring. The anomaly score of a point X_j is the negative of the average of $H_i(\bar{Y}_j)$ across all histograms i.

	Histogram Forest Parameters
K	Dimension of projected data
T	Number of histograms
w	Bin width
	Other variables
X	$N \times D$ data matrix
Y	$N \times K$ projected data matrix
A_i	$K \times D$ projection matrix
b_i	Shift $\sim$ Uniform$(0, w)$
H_i	Hashtable/count-min sketch

Table 1: Notation.

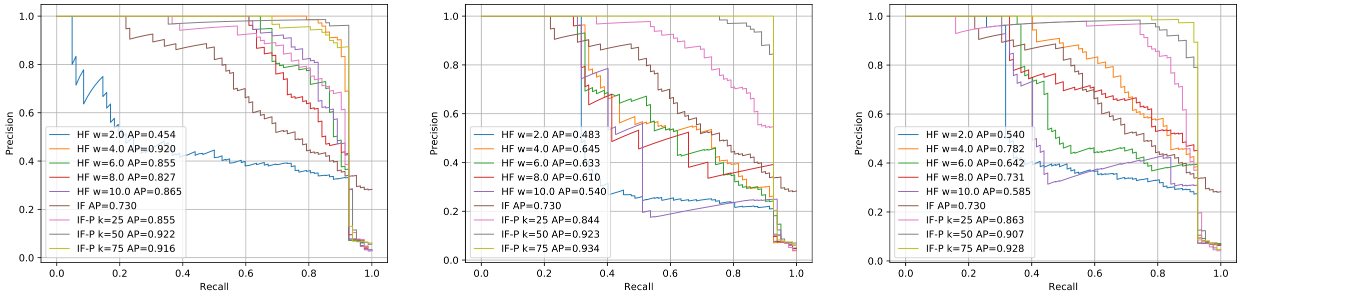
1.1 HF Results

Methods compared in Figure 1 (on the synthetic data with noise):

- **HF:** Histogram forest with varying bin width w, number of projected dimensions K and number of histograms T.
- **IF:** iForest with 100 trees.
- **IF-P:** iForest with 100 trees and data projected to k dimensions using sparse random projections, with varying k.

Observations from Figure 1:

1. HF with $K = 25, T = 100, w = 4.0$ is on par with iForest with 100 trees on the data projected to $k = 50$ or $k = 75$ dimensions, and better than iForest for $k = 25$.
2. HF with $K = 50, T = 100$ has poor performance for all w. This was our configuration setting for the previous histogram experiments (with some minor variations): we considered them a failure without trying a *lower* value of K. This suggests that K as a measure of data approximation quality may not be the right interpretation.
3. HF with $K = 50, T = 200$ improves performance over $T = 100$ slightly, for all w.



(a) $K = 25, T = 100$

(b) $K = 50, T = 100$

(c) $K = 50, T = 200$

Figure 1: Precision-recall curves on synthetic data with 100 noisy dimensions.

1.2 Interpretation — Approximate Nearest Neighbors

The binned data point in our histogram is identical to a point "hashed" using 2-stable distributions (**datar2004locality** 3.1), which extends LSH to solve the approximate nearest neighbors problem in L_2. In this method, each point x is "hashed" using the function $h(x) = \lfloor \frac{x^T a + b}{w} \rfloor$ where a is a random Gaussian vector, b is a random shift and w is the bin width. The probability that two points x and y hash to the same value is proportional to $\|x - y\|_2$.

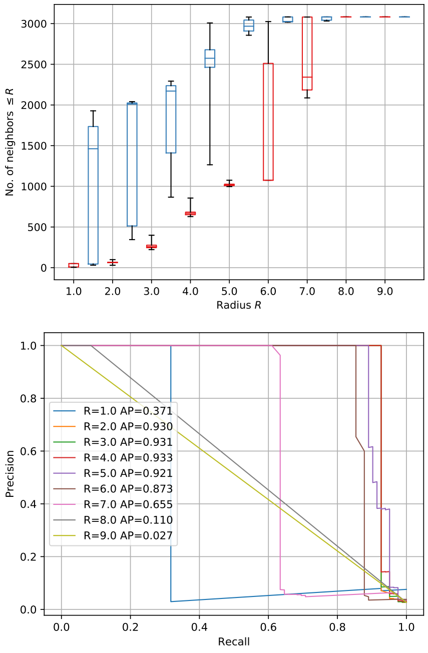


Figure 2: (Top) Number of neighbors within radius R, for anomalous (red) and benign (blue) points. (Bottom) PR curves using exact nearest neighbor counts.

Each point is hashed K times, and points that agree on all K values are considered candidate nearest neighbors. However, as K grows, it becomes less likely that even nearby points agree on all K values; hence, the process is repeated T times, and the points agreeing on all K values at in least one of the T trials are considered candidate nearest neighbors.

Each configuration of K, T and w results in a certain *probability gap* for points to be considered nearest neighbors. Specifically, this probability gap is defined by the 4-tuple (R, p_1, cR, p_2):

- The probability of two points with distance $\leq R$ being hashed to the same values is $\geq p_1$.
- The probability of two points with distance $\geq cR$ being hashed to the same values is $\leq p_2$.

Hence, our anomaly score for a point can be viewed as the negative of the approximate number of points within a sphere of some radius R centered at that point. The optimal R for anomaly detection on a given dataset is data-dependent (Fig. 2). Hence, the optimal K, T, w are also data-dependent. This explains why a $K = 25$ histogram forest may perform better than a $K = 50$ one.

1.3 Data-dependent Parameter Tuning

The anomaly detection performance is sensitive to picking the right parameters K, T and w (equivalent to picking a radius R and the LSH approximation probabilities p_1 and p_2). There have been some approaches to auto-tuning the LSH parameters based on the data. However, they were primarily designed for nearest-neighbor search and do not work for data streams. Could we design a data structure for data streams that is tailored for anomaly detection?

- LSH-Forest (**bawa2005lsh**). The earliest approach; this is a very simple approache that builds a trie using each bit of the K-element sketch. Demonstrated for the Jaccard and cosine distances (since their LSH values are binary). A theoretical analysis was recently published (**andoni2017lsh**).

- Parameter-free LSH (**ahle2017parameter**). The most recent approach, designed for Euclidean distances. This builds a "multi-level LSH" data structure that tries multiple values of the projection size K, and finds the best level for each nearest-neighbor query.

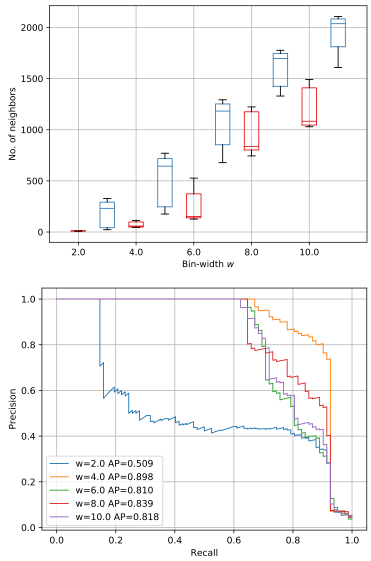


Figure 3: (Top) Distribution of approximate nearest neighbor counts via HF for different w (Bottom) PR curve for HF. $K = 25, T = 100$.

1.4 Relationship to Chains

2 Chains

Pre-initialization. This is done once, and is unchanged for all chains.

1. **Initialize the random projection matrix** A**.**
2. **Project the data.** $Y = XA^T$.
3. **Set** Δ_f = half the range of projected feature f.
4. **Sample a shift** b_{if} for each chain i and feature f.

	Chain Parameters
K	Dimension of projected data
C	Number of chains
D	Chain depth
Δ_f	Initial/maximum bin width
	Other variables
f	A feature index in $1, \ldots, K$
X	$N \times D$ data matrix
Y	$N \times K$ projected data matrix
A	$K \times D$ projection matrix
b_{if}	Shift $\sim$ Uniform$(0, \Delta_f)$
H_{id}	Hashtable/count-min sketch

Table 2: Notation.

Chain fitting. For each chain $i = 1, \ldots, C$:

For each depth $d = 1, \ldots, D$:

1. **Sample a dimension** $f_d \in \{1, \ldots K\}$ and store the dimension sampled at each depth. Let $0 \leq c(f, d) \leq d$ be the number of times dimension f has been sampled in the chain until depth d.
2. **Bin the data.** Let $Y_j = (y_{j1}, \ldots, y_{jK})$ be a projected data point. Denote by the $Z_{jd}[f]$ the "unfloored bin index" of feature f for point j at depth d:

$$Z_{jd}[f] = \frac{y_{jf} + b_{if}/2^{c(f,d)-1}}{\Delta_f/2^{c(f,d)-1}} \quad \forall f = 1, \ldots K \tag{1}$$

At depth d, the bin indices $\bar{Y}_{jd}$ are logically defined as follows, for $f = 1, \ldots, K$:

$$\bar{Y}_{jd}[f] = \begin{cases} 0, & \text{if } c(f,d) = 0 \\ \lfloor Z_{jd}[f] \rfloor & \text{if } c(f,d) > 0 \end{cases} \tag{2}$$

In practice, the bins are computed recursively at each depth as follows:

$$\bar{Y}_{jd}[f] = \begin{cases} 0 & \text{if } c(f,d) = 0 \\ \lfloor Z_{jd}[f] \rfloor & \text{if } c(f,d) = 1 \\ \lfloor 2 \times Z_{jd-1}[f] - \frac{b_{if}}{\Delta_f} \rfloor & \text{if } c(f,d) > 1 \end{cases} \tag{3}$$

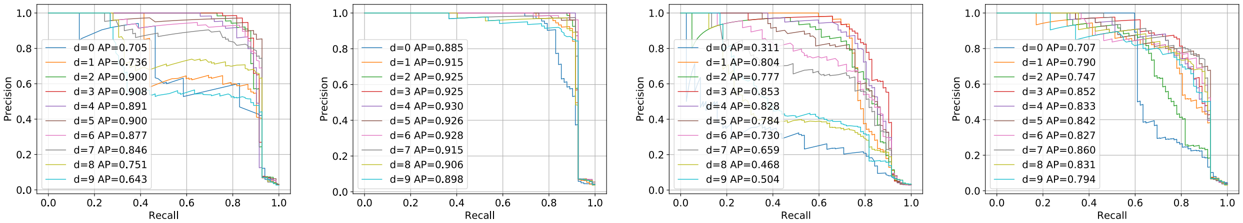
The recurrence is derived as follows:

$$x_k = \frac{y + b/2^k}{\Delta/2^k} = \frac{y + b/2^{k-1}}{\Delta/2^k} - \frac{b/2^k}{\Delta/2^k} = 2\left(\frac{y + b/2^{k-1}}{\Delta/2^{k-1}}\right) - \frac{b}{\Delta} = 2x_{k-1} - \frac{b}{\Delta} \tag{4}$$

3. **Store bin counts.** In an exact hashtable or count-min sketch H_{id}, increment the count $H_{id}(\bar{Y}_j)$ for $j = 1, \ldots, N$.

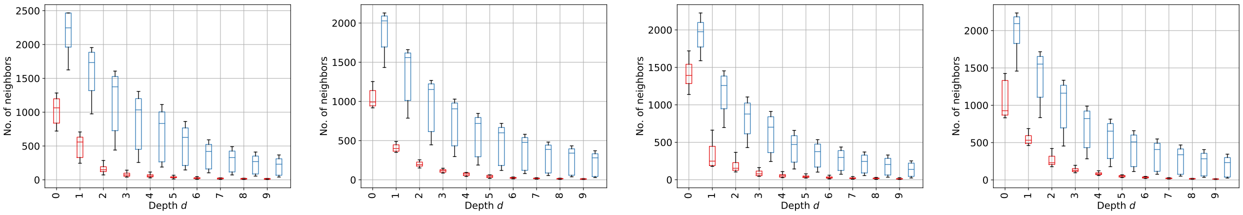
Scoring at depth d**.** The anomaly score of a point X_j at depth d is $-\text{average}_i(H_{id}(\bar{Y}_j))$.

2.1 Results with bin-counts as anomaly scores



(a) $K = 50, C = 10$ (b) $K = 50, C = 50$ (c) $K = 25, C = 10$ (d) $K = 25, C = 50$

Figure 4: Precision-recall curves on synthetic data with 100 noisy dimensions, $D = 10$.



(a) $K = 50, C = 10$ (b) $K = 50, C = 50$ (c) $K = 25, C = 10$ (d) $K = 25, C = 50$

Figure 5: No. of neighbors in the same bin for anomalous (red) and benign (blue) points.

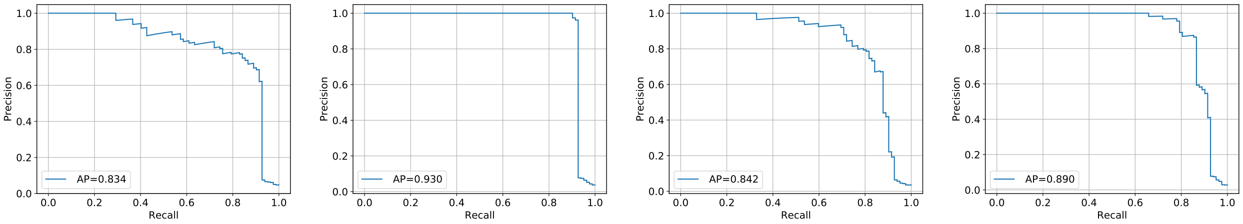
2.2 Results with LOCI scores as anomaly scores

Previously, we scored each point X_j using chain i at depth d using the number of neighbors lying in the same bin, $H_{id}(\bar{Y}_j)$. We modify this to obtain a LOCI score $L_{id}(\bar{Y}_j)$ as:

$$\bar{H}_{id}(\bar{Y}_j) = H_{id}(\bar{Y}_j) \times 2^d \tag{5}$$

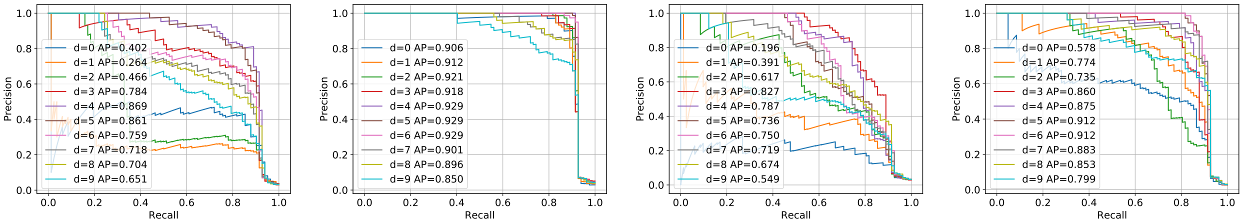
$$L_{id}(\bar{Y}_j) = \bar{H}_{id}(\bar{Y}_j) - \frac{1}{N-1} \sum_{k \neq j} \bar{H}_{id}(\bar{Y}_k) \tag{6}$$

Then the anomaly score of a point X_j is $-\text{average}_i \left(\min_d L_{id}(\bar{Y}_j) \right)$.



(a) $K = 50, C = 10$ (b) $K = 50, C = 50$ (c) $K = 25, C = 10$ (d) $K = 25, C = 50$

Figure 6: Precision-recall curves on synthetic data with 100 noisy dimensions, $D = 10$.



(a) $K = 50, C = 10$ (b) $K = 50, C = 50$ (c) $K = 25, C = 10$ (d) $K = 25, C = 50$

Figure 7: Precision-recall curves with LOCI scores at different depths d.

2.3 Stability across runs

Chain Parameters	Mean AP	Standard Deviation
$K = 50, C = 10$	0.824	0.064
$K = 50, C = 50$	0.930	0.001
$K = 25, C = 10$	0.742	0.123
$K = 25, C = 50$	0.901	0.019

Table 3: Chain AP stability across 5 runs, $D = 10$.

2.4 Number of unique bin IDs with depth

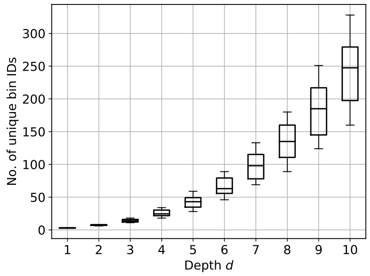


Figure 8: No. of unique bins for $C = 100$ chains, $K = 50, d = 10$.

2.5 Scores for different anomaly types

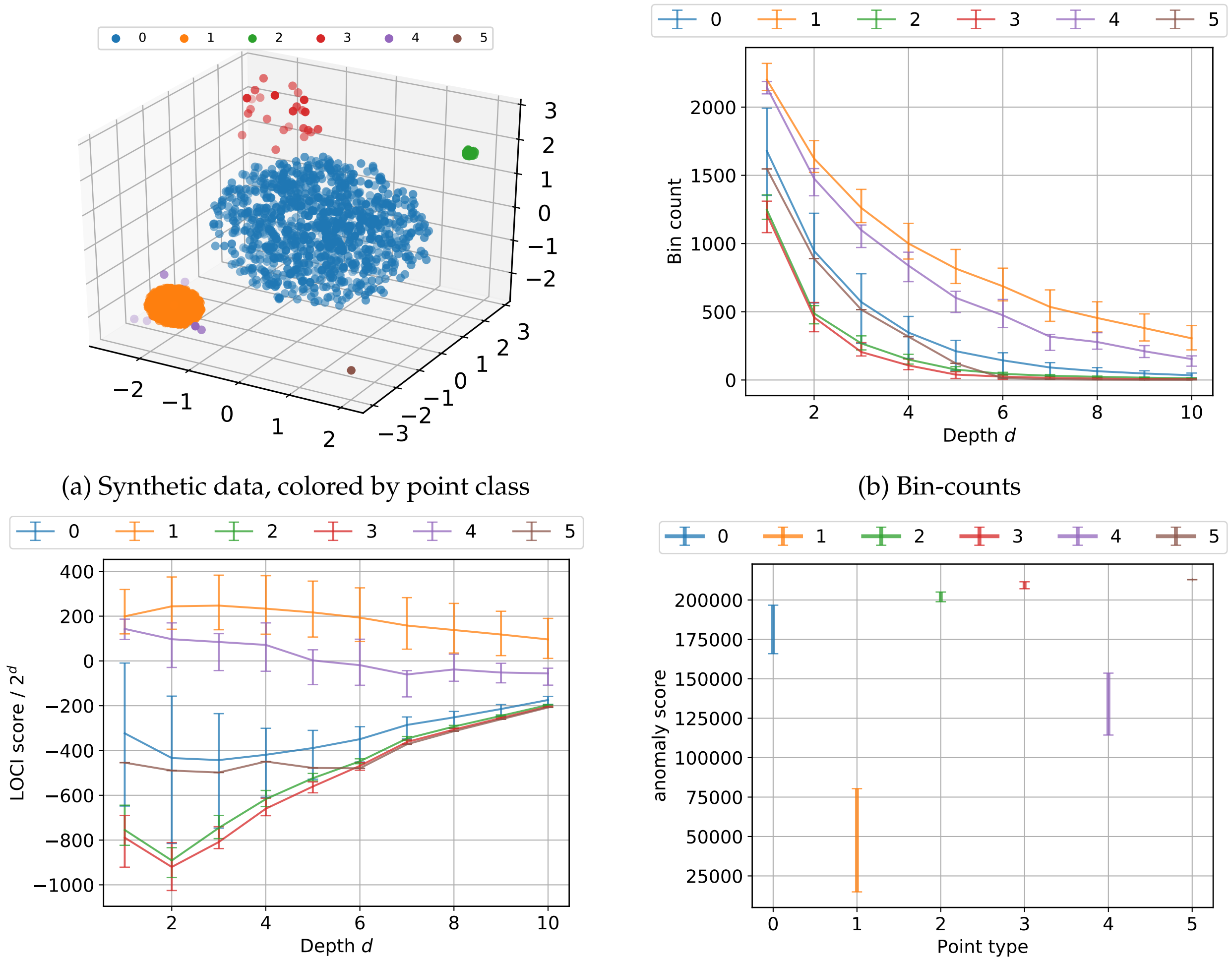
Fig. 9 illustrates how the anomaly scores vary for different classes of points, shown in Fig. 9(a):

Class	No. of points	Description	Color
Class 0	1000	Benign sparse points	Blue
Class 1	2000	Benign dense points	Orange
Class 2	50	Anomalous dense points	Green
Class 3	25	Anomalous sparse points	Red
Class 4	6	Locally anomalous points	Purple
Class 5	1	Single anomalous point	Brown

From Fig. 9(b) conforms to intuition: as the depth in the chain increases, the average number of points lying in the same bin (across chains) decreases for all classes of points. This is due to two factors: (i) increasing the number of bin dimensions, and (ii) decreasing the bin width. Point classes lying in denser regions have higher bin counts.

Fig. 9(c) shows the LOCI scores (eq. 6) at each depth d, scaled up by 2^d for visual clarity. The LOCI score of a point is simply its bin count minus the mean bin count of all other points at that depth. Hence, as d increases and the bin counts of all points tend to 1, the LOCI score tends to 1.

Fig. 9(d) shows the anomaly scores. The order of anomalousness across point classes is inversely proportional to their global density.



(a) Synthetic data, colored by point class

(b) Bin-counts

(c) LOCI scores (mean-adjusted bin-counts)

(d) Anomaly scores (negative of the minimum LOCI score across all depths)

Figure 9: Score distributions for different point classes (see §2.3 for details). Error bars indicate the median, 5th and 95th percentile scores across all points. Chains with $K = 50, C = 100, d = 10$.

2.6 Results on other high-dimensional datasets

See §3 for details on the datasets, and results with baseline methods.

Dataset	AP	AUC
	High-dimensional Datasets	
Madelone	0.5174 ± 0.0056	0.5127 ± 0.0078
Madelone-Noisy (5.0, 0.1, 10.0)	0.8271 ± 0.0360	0.9749 ± 0.0059
Madelone-Noisy (5.0, 0.1, 20.0)	0.1485 ± 0.0178	0.6587 ± 0.0215
Madelone-Noisy (5.0, 0.1, 50.0)	0.1161 ± 0.0044	0.5002 ± 0.0106
Letter-Recognition	0.4688 ± 0.0087	0.5274 ± 0.0044
Letter-Recognition-Noisy (5.0, 0.1, 10.0)	0.1019 ± 0.0018	0.5005 ± 0.0046
Letter-Recognition-Noisy (5.0, 0.1, 20.0)	0.1088 ± 0.0022	0.5046 ± 0.0056
Letter-Recognition-Noisy (5.0, 0.1, 50.0)	0.1068 ± 0.0016	0.5010 ± 0.0062
Gisette	0.4633 ± 0.0081	0.4598 ± 0.0137
Gisette-Noisy (5.0, 0.1, 10.0)	0.0934 ± 0.0012	0.4869 ± 0.0069
Gisette-Noisy (5.0, 0.1, 20.0)	0.1090 ± 0.0025	0.5106 ± 0.0082
Gisette-Noisy (5.0, 0.1, 50.0)	0.0925 ± 0.0013	0.4933 ± 0.0056
Isolette	0.4402 ± 0.0131	0.5145 ± 0.0109
Isolette-Noisy (5.0, 0.1, 10.0)	0.1098 ± 0.0023	0.5338 ± 0.0042
Isolette-Noisy (5.0, 0.1, 20.0)	0.0895 ± 0.0017	0.4894 ± 0.0064
Isolette-Noisy (5.0, 0.1, 50.0)	0.0968 ± 0.0018	0.4929 ± 0.0041

Table 4: AP/AUC over 10 runs, $K = 50, C = 100, D = 10$.

Note on results on Madelone-Noisy (5.0, 0.1, 10.0) vs. LODA. LODA selects the optimal number histograms and bins (equivalently, bin width) based on the data. In the streaming variant, a sample of the data is used to select these hyperparameters. Once selected, they remain fixed for the entire duration of the LODA run.

XStream, on the other hand, relies on ensembling instead of finding the "right" bin-width. Increasing K for XStream to the optimal selected by LODA ($K = 125$) brings the AP up to ~ 0.90.

There is an additional factor that makes LODA's performance superior in the static case: LODA uses sparse *Gaussian* random projections, which are a better approximation than the sparse *binary* random projections used by XStream.

Note that increasing K up to 200 makes the AP comparable to that of LODA.

2.7 Results on other low/medium dimensional datasets

See §3 for details on the datasets, and results with baseline methods.

Dataset	**AP**	**AUC**
	Low/medium-dimensional Datasets	
Breast-Cancer-Wisconsin	0.9007 ± 0.0083	0.9231 ± 0.0074
Breast-Cancer-Wisconsin-Noisy (100, 0.1)	0.8514 ± 0.0180	0.8625 ± 0.0186
Breast-Cancer-Wisconsin-Noisy (1000, 0.1)	0.8514 ± 0.0180	0.8625 ± 0.0186
Breast-Cancer-Wisconsin-Noisy (2000, 0.1)	0.8514 ± 0.0180	0.8625 ± 0.0186
Breast-Cancer-Wisconsin-Noisy (5000, 0.1)	0.8514 ± 0.0180	0.8625 ± 0.0186
Ionosphere	0.8425 ± 0.0056	0.8855 ± 0.0046
Magic-Telescope	0.4633 ± 0.0081	0.4598 ± 0.0137
Pima-Indians	0.5458 ± 0.0075	0.7041 ± 0.0096

Table 5: AP/AUC over 10 runs, $K = 50, C = 100, D = 10$.

3 Baselines

3.1 Datasets

Datasets shown in Table **??**.

3.2 Adding Noise To Datasets

We used 2 different methods to generate benchmark datasets. The first method was applicable only to the low-dimensional and mid-dimensional datasets, and the second method was for high-dimensional datasets. We explain both of these methods below: **Method 1**: We add noisy columns to the original dataset. The number of noisy columns is in the percentage of original number of columns that are added, and the noise level governs the mean and standard deviation of the gaussian distribution. We play around with percentage of noisy columns to be added,

Dataset Name	**#Samples**	**#Features**
breast-cancer-wisconsin (Mid)	568	30
ionosphere (Mid)	350	33
magic-telescope (Low)	19020	10
pima-indians (Low)	768	8
gisette (High)	7000	4970
isolet (High)	7797	616
letter-recognition (High)	7797	616
madelon (High)	2600	500

Table 6: List of benchmark datasets used.

choosing values such as 100, 1000, 2000, and 5000. For noise-levels, we played with 0.01, 0.1, 0.2, 0.25. The mean and standard deviation of the Gaussian distribution was just noise-level multiplied with the original mean, or the original standard deviation of the entire dataset.

Method 2: We first discard all the original anomalous points. Now for the remaining nominal points, we first choose 10% of the features and mark them as important features. We further proceed with choosing 5% of the dataset as important samples. Now, in this sub-block, which contains only important samples and important features, we add Gaussian noise to the original data. Again, Gaussian noise is used, parameterized by the mean, and the standard deviation of the entire nominal dataset. Further this noise is scaled according to the user-specified signal to noise ratio, and added to the original sub-block of our dataset and features.

3.3 Baseline Algorithms

- **iForest**: We use it with recommended sample size of 256 or the entire dataset, whichever is minimum. The hlim is set to 15 to be comparable to HS-Trees, and number of components is set to 100.
- **HS-Trees**: Use it with recommended setting of 15 max depth trees, and number of components is 100.
- **LODA**: We set the sparsity factor to $\frac{1}{\sqrt{(d)}}$. We allow LODA to select best width for histograms, but fix the number of histograms to 100 to be comparable to rest of the algorithms.
- **RS-Hash**: For RS-Hash, we use 1000 sampling points, as recommended and 100 components.

3.4 Baseline Results

3.5 Time and Space Complexity

3.6 Results

AUC / AP Results

Time-Analysis iForest is the fastest, RSHash is the slowest (RSHash implementation, however can be optimized)

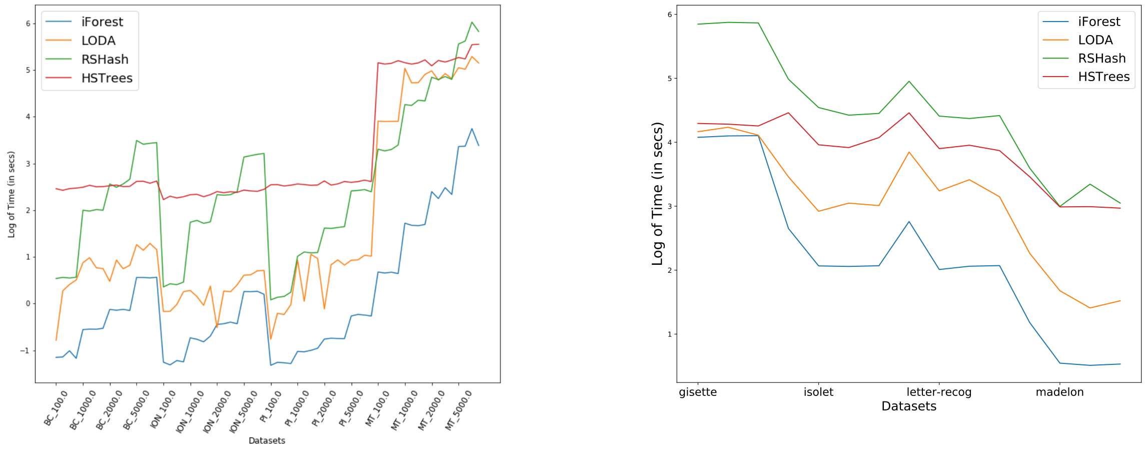
3.7 Streaming Baseline Setup

Datasets:

We use the same datasets as in HighDimension case in static setting. We create 10 shuffled versions for each dataset. The datasets could either be shuffled randomly or it could be clustered in time. For the clustered shuffled, the input also includes a clustering percentage, that is how much percentage of anomalies should be clustered in time. We currently do not experiment with

Dataset	IF	LODA	RSH	HST	XS
High-dimensional Datasets					
gisette (30,0.3,1.2)	0.683 ± 0.045	0.531 ± 0.018	0.46 ± 0.005	0.628 ± 0.021	
gisette (30,0.3,10)	0.541 ± 0.027	0.321 ± 0.004	0.354 ± 0.002	0.471 ± 0.024	
gisette (30,0.3,20)	0.45 ± 0.022	0.29 ± 0.003	0.308 ± 0.003	0.347 ± 0.017	
gisette (30,0.3,30)	0.432 ± 0.016	0.301 ± 0.005	0.31 ± 0.002	0.316 ± 0.006	
isolet (30.0,0.3,1.2)	0.537 ± 0.03	0.58 ± 0.03	0.441 ± 0.01	0.559 ± 0.028	
isolet (30.0,0.3,10)	0.372 ± 0.011	0.35 ± 0.007	0.334 ± 0.004	0.391 ± 0.015	
isolet (30.0,0.3,20)	0.33 ± 0.009	0.313 ± 0.006	0.311 ± 0.001	0.322 ± 0.007	
isolet (30.0,0.3,30)	0.3 ± 0.002	0.292 ± 0.004	0.296 ± 0.001	0.293 ± 0.002	
letter-recognition (30.0,0.3,1.2)	0.49 ± 0.026	0.534 ± 0.03	0.44 ± 0.01	0.519 ± 0.053	
letter-recognition (30.0,0.3,10)	0.374 ± 0.015	0.337 ± 0.003	0.339 ± 0.008	0.375 ± 0.007	
letter-recognition (30.0,0.3,20)	0.309 ± 0.008	0.289 ± 0.002	0.291 ± 0.002	0.31 ± 0.007	
letter-recognition (30.0,0.3,30)	0.319 ± 0.01	0.305 ± 0.003	0.308 ± 0.001	0.307 ± 0.004	
madelon (5.0,0.05,1.2)	0.896 ± 0.051	1.0 ± 0.0	0.92 ± 0.007	0.969 ± 0.039	
madelon (5.0,0.05,10)	0.643 ± 0.13	0.988 ± 0.012	0.784 ± 0.068	0.899 ± 0.094	
madelon (5.0,0.05,20)	0.24 ± 0.093	0.112 ± 0.027	0.16 ± 0.039	0.377 ± 0.143	
madelon (5.0,0.05,30)	0.07 ± 0.013	0.054 ± 0.007	0.083 ± 0.004	0.099 ± 0.059	

Table 7: AP for Baseline algorithms on High Dimensional Dataset. The numbers in bracket indicates the percentage of features, fraction of samples to which noise is added, and the SnR ratio of noise respectively.



(a) Low/Mid - dimensional dataset analysis

(b) High dimensional dataset analysis

Figure 10: RunTimes of algorithms over datasets with different noise levels. X-axis is over different datasets with different noise levels. Y-axis is log of runtime (in seconds).

clustered in time datasets.

Dataset	IF	LODA	RSH	HST	XS
Low/medium-dimensional Datasets					
breast-cancer	0.635 ± 0.024	0.883 ± 0.019	0.758 ± 0.004	0.719 ± 0.019	
breast-cancer (100,0.1)	0.661 ± 0.036	0.839 ± 0.024	0.727 ± 0.01	0.702 ± 0.026	
breast-cancern (1000,0.1)	0.476 ± 0.043	0.686 ± 0.047	0.394 ± 0.007	0.531 ± 0.047	
breast-cancer (2000,0.1)	0.447 ± 0.026	0.643 ± 0.066	0.444 ± 0.015	0.474 ± 0.046	
breast-cancer (5000,0.1)	0.404 ± 0.021	0.508 ± 0.086	0.4 ± 0.013	0.425 ± 0.022	
ionosphere	0.819 ± 0.008	0.79 ± 0.014	0.824 ± 0.003	0.83 ± 0.005	
ionosphere (100,0.1)	0.756 ± 0.031	0.771 ± 0.013	0.773 ± 0.005	0.725 ± 0.018	
ionosphere (1000,0.1)	0.59 ± 0.044	0.742 ± 0.032	0.573 ± 0.008	0.579 ± 0.054	
ionosphere (2000,0.1)	0.441 ± 0.033	0.736 ± 0.029	0.505 ± 0.033	0.446 ± 0.027	
ionosphere (5000,0.1)	0.403 ± 0.022	0.675 ± 0.048	0.376 ± 0.006	0.399 ± 0.028	
magic-telescope	0.655 ± 0.011	0.623 ± 0.004	0.674 ± 0.003	0.678 ± 0.005	
magic-telescope (100,0.1)	0.593 ± 0.01	0.617 ± 0.003	0.576 ± 0.005	0.601 ± 0.015	
magic-telescope (1000,0.1)	0.451 ± 0.018	0.592 ± 0.013	0.419 ± 0.008	0.471 ± 0.021	
magic-telescope (2000,0.1)	0.407 ± 0.021	0.586 ± 0.007	0.407 ± 0.003	0.409 ± 0.018	
magic-telescope (5000,0.1)	0.376 ± 0.011	0.538 ± 0.026	0.38 ± 0.002	0.378 ± 0.005	
pima-indians	0.498 ± 0.015	0.48 ± 0.015	0.502 ± 0.005	0.518 ± 0.006	
pima-indians (100,0.1)	0.477 ± 0.013	0.472 ± 0.01	0.444 ± 0.009	0.478 ± 0.007	
pima-indians (1000,0.1)	0.385 ± 0.017	0.444 ± 0.014	0.379 ± 0.006	0.397 ± 0.024	
pima-indians (2000,0.1)	0.35 ± 0.014	0.41 ± 0.016	0.361 ± 0.005	0.369 ± 0.023	
pima-indians (5000,0.1)	0.343 ± 0.008	0.4 ± 0.028	0.338 ± 0.003	0.349 ± 0.014	

Table 8: AP over 10 runs for Low Dimensional Datasets. The numbers in the brackets alongside the dataset name indicate the percentage of original columns, added as noise, and the relative noise factor, respectively.

Algorithms

- RS-Hash: We use decay constant to be 0.015, and use first 256 samples to setup the 100 components, and then compute AUC/AP over time.
- HS-Trees:
- LODA:

Results

Friedman Statistic	Low-Dim Datasets	High-Dim Datasets	All Datasets
p-value	1.155e-09	8.362e-10	<2.2e-16
Wilcoxon Test		**p-val w/ XST**	
iFor	1.907e-06	0.7271	0.001196
RSH	6.676e-06	0.7392	0.002026
LODA	9.537e-07	0.1744	0.0001096
HSTrees	2.861e-06	0.2375	2.049e-05

Table 9: Friedman's test statistics among methods. And wilcoxon test p-values, indicating that XStream performs better than all the baseline methods for low-dimensional datasets and over the set of all datasets, but no method is more significant than XStream and vice-versa.

	iForest	RSHash	LODA	HS Trees	XStream
iForest		0.9977	0.9999	0.4839	1
RSHash	0.0025		0.9998	0.0752	1
LODA	0.00013	0.00019		0.00010	1
HS Trees	0.5321	0.9299	0.999		1
XStream	1.907e-06	6.676e-06	9.537e-07	2.861e-06	

Table 10: Wilcoxon test p-value result for Low Dimension dataset. XST > LODA > RSH   HST   IF

	iForest	RSHash	LODA	HS Trees	XStream
iForest		0.7432	0.07528	0.00618	0.2853
RSHash	0.2689		0.02964	0.0001	0.2729
LODA	0.9299	0.9728		0.211	0.835
HS Trees	0.9944	0.9999	0.7996		0.7738
XStream	0.7271	0.7392	0.1744	0.2375	

Table 11: Wilcoxon test p-value result for High Dimension dataset. XST   IF   RSH > LODA   HST   XST

	iForest	RSHash	LODA	HS Trees	XStream
iForest		0.987	0.9727	0.01942	0.9989
RSHash	0.0135		0.9542	6.754e-05	0.9981
LODA	0.02815	0.04711		0.0003357	0.9999
HS Trees	0.9812	0.9999	0.9997		1
XStream	0.001196	0.00202	0.000109	2.049e-05	

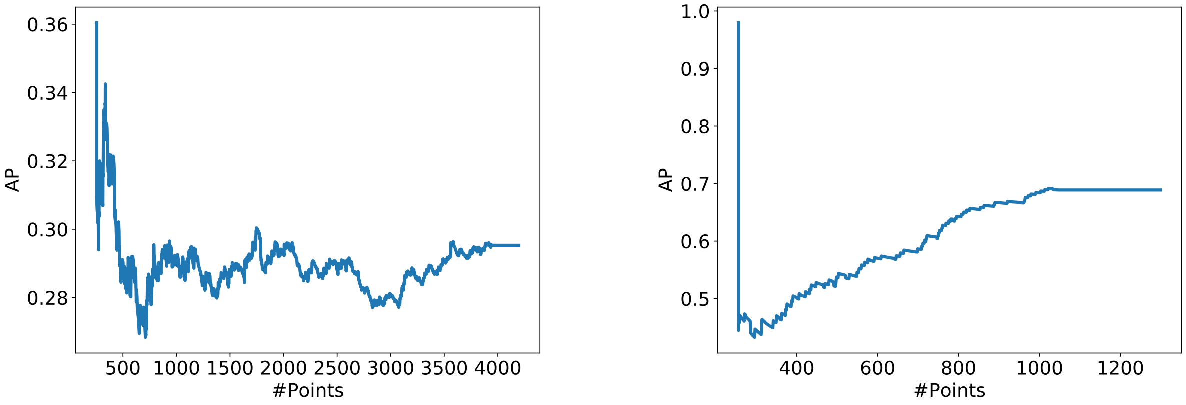
Table 12: Wilcoxon test p-value result for High Dimension dataset. XST   IF   HST > LODA   RSH   XST

Table 13: Symbol List

Symbols	Description
N	Number of data points
D	Number of dimensions/features
C	Number of ensemble components (trees, chains, histograms, etc.)
d	Max depth of trees/chains
ψ	iForest/HS-Trees/RS-Hash: sampling size
r	RS-Hash: Number of sampled dimensions
b	LODA: number of histogram bins
m	X-STREAM/RS-Hash: Number of CMS hash functions
L	X-STREAM/RS-Hash:: CMS hash table size
k	X-STREAM: Number of random projections
M	X-STREAM: Number of subspaces
W	X-STREAM: Number of windows

Table 14: Time and space complexity of the ensemble anomaly detection algorithms compared in this paper. In streaming case, a data point/vector arrives at a time for HS-Stream, LODA, and RS-Hash; whereas an update to a single feature for a data point arrives at a time.

Algorithm	Time Complexity		Space Complexity
	Training	Scoring/Updating	
Batch/Offline			
iForest	$O(C\psi \log \psi)$	$O(C \log \psi)$	$O(C\psi)$
HS-Trees	$O(Cd\psi)$	$O(Cd)$	$O(C2^d)$
LODA	$O(NC\sqrt{D})$	$O(C\sqrt{D})$	$O(C\sqrt{D} + Cb)$
RS-Hash	$O(Crm\psi)$	$O(Crm)$	$O(CLm)$
X-STREAM	$O(M[NDk + Ckmd])$	$O(M[Dk + Ckmd])$	$O(MCLmd)$
Streaming/Online			
HS-Stream	–	$O(Cd + C\psi)$	$O(C2^d)$
LODA	–	$O(C\sqrt{D} + Cb)$	$O(C\sqrt{D} + Cb)$
RS-Hash	–	$O(Crm)$	$O(CLm)$
X-STREAM	–	$O(WMCkmd)$	$O(WMCLmd)$



(a) Low/Mid - dimensional dataset analysis

(b) High dimensional dataset analysis

Figure 11: AP for two of the baseline datasets over time, using RS-Hash Streaming Algorithm.