Isolation Forest Based Efficient Unsupervised Machine Learning Algorithms

Thesis submitted in partial fulfilment of the requirements

for the award of the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Nidhi Ahlawat

Under the supervision of

Dr. Amit Awekar

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

Guwahati - 781039 Assam India

March, 2025

Dedicated to

My Parents

My Sister and Brother

.....

It is from their love, care and support I derive my strengths...

Acknowledgements

I express my heartfelt gratitude to Dr. Amit Awekar, my Ph.D. advisor. He is an excellent mentor and has always provided an atmosphere to pursue research problems that excite me the most. At the same time, his guidance helped in shaping ideas to concrete objectives while aiming at the ultimate goal. He always encouraged me to participate in various seminars, tutorials, meetings, workshops, and conferences, often by providing travel support, and I am grateful to him. I am also thankful to my doctoral committee members, Dr. Pradip Krumar Das, Dr. Vijaya Saradhi, and Dr. Hanumant Singh Shekhawat, for providing valuable feedback during the research. I will thank several interns and collaborators: Suryansh Singh and Sandeep Chatterjee.

I am thankful to ACM India for travel support to attend the conferences and workshops. Also, I thank MHRD, Government of India, for providing financial assistantship throughout the Ph.D. program. I am thankful to the Dept. of CSE for providing the necessary computing resources used during the work.

I am fortunate to have several wonderful friends who were there to support and celebrate with me during various events of the grad school experience. Especially I would like to thank Sonia, Vanshali Sharma, Abhishek, Nilotpal Biswas, Debanjan Roy Chowdhury, Menaxi Bagchi, Maithilee Patawar, Suraj Kumar Pandey, Hemraj Raikwar, Rohit Raj Rai, Saipriya Karnati, Price Yadav. I am thankful to my juniors Soumya Bhardwaj, Laxita Aggarwal, and Shifali, who made me feel good with their warm gestures.

It goes without saying that this journey would not have been possible without the persistent support, unconditional love and profound encouragement of my entire family. They never doubted my intentions and wholeheartedly supported me in all my endeavors. I fall short of words to express my gratitude to my husband, Arvind. His support and insightful suggestions have shaped my journey and prepared me for life's challenges.

March 24, 2025 Nidhi Ahlawat

Declaration

I certify that

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisor(s).
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the references. Elaborate sentences used verbatim from published work have been clearly identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge and understanding and take complete responsibility if any complaint arises.
- I am fully aware that my thesis supervisor(s) are not in a position to check for any possible instance of plagiarism within this submitted work.

March 24, 2025 Nidhi Ahlawat

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati - 781039 Assam India

Dr. Amit Awekar
Associate Professor
Email : awekar@iitg.ac.in
Phone : +91-361-258-2373

Certificate

This is to certify that this thesis entitled "**Isolation Forest Based Efficient Unsupervised Machine Learning Algorithms** " submitted by **Nidhi Ahlawat**, in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, to the Indian Institute of Technology Guwahati, Assam, India, is a record of the bonafide research work carried out by the candidate under my guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India. To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date: March 24, 2025
Place: Guwahati

Dr. Amit Awekar
(Thesis Supervisor)

Abstract

Many ML algorithms have common redundancies that make them impractical for large datasets. The overarching goal of this thesis is to prune the redundant computations with minimal loss in the quality of the downstream tasks. This dissertation focuses on three unsupervised machine-learning tasks: clustering, anomaly detection, and model update. We utilize the Isolation Forest data structure as a tool to improve efficiency for all three tasks. This data structure was initially developed to perform anomaly detection task in an unsupervised manner.

Specifically, we focus on the following three scenarios: 1. When an application needs all-pair distances: How to compute all-pair distances faster by optimizing the order of distance computation? 2. When an application needs only a subset of all-pair distances: How do we quickly identify the required subset of all pairs? 3. When new data causes concept drift: How to update the model quickly?

For the **first** scenario, we develop an algorithm: fast MBD ($fMBD$) that computes all-pair distances with up to 5X speed-up. Our $fMBD$ algorithm has no approximation or heuristic, and it computes the exact distance for each data point pair. We demonstrate the effectiveness of the $fMBD$ algorithm with clustering and anomaly detection applications. For the **second** scenario, we develop a scalable $MBScan$ ($sMBScan$) clustering algorithm that selectively computes distances between data point pairs. Our algorithm achieves up to 53X speed up with up to 96% reduction in the memory footprint and no loss in the clustering quality. For the **third** scenario, we develop an incremental Isolation Forest ($I^2Forest$) that quickly updates the Isolation Forest data structure in response to the arrival of new data. $I^2Forest$ is particularly effective when the new data causes concept drift. $I^2Forest$ has significantly lower training time than retraining the model from scratch. $I^2Forest$ also performs better than other incremental approaches for model update.

Contents

List of Figures

List of Algorithms

List of Tables

1

Introduction

1.1 Overview

This dissertation presents efficient algorithms for mass-based clustering [1], anomaly detection [2], and model update in response to new data [3]. Our primary research contribution is to effectively identify and avoid redundant computation without impacting the quality of output. All our methods utilize Isolation Forest (*iForest*) [4] as the underlying data structure and tool to enhance efficiency.

Machine learning algorithms are used across various real-world applications, including healthcare, finance, marketing, manufacturing, transportation, cybersecurity, agriculture, retail, education, entertainment, and smart cities. However, efficiently scaling these algorithms for large-scale data presents a significant challenge. Additionally, the emergence of new data can introduce concept drift, causing model performance to degrade and necessitating frequent retraining. Redundant computation can be identified and minimized to enhance efficiency while maintaining the effectiveness of the algorithms [5],[6],[7]. Researchers have long focused on bridging the gap between effectiveness and efficiency through various approaches. Techniques such as model simplification, computational optimization, and computational parallelization [5] have been widely adopted for the efficiency of machine learning algorithms in handling large-scale

and dynamic data. These solutions have been applied across numerous tasks, including classification [8],[9],[10] clustering [11],[12], anomaly detection, and incremental model updates to manage concept drift [13].

1.2 Problem Definition

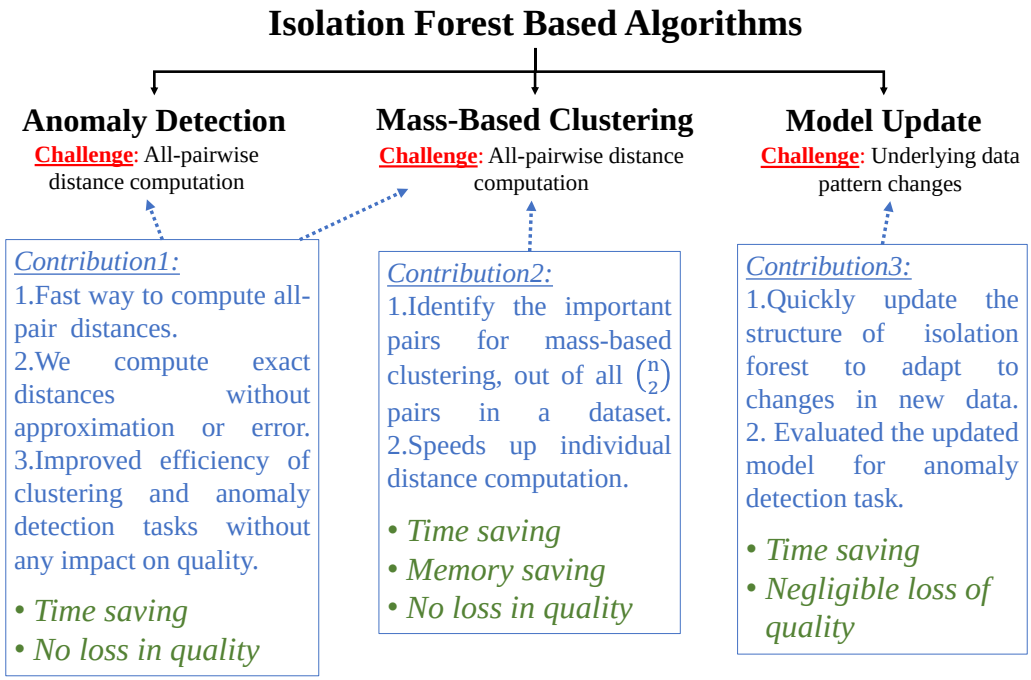


Figure 1.1: An Overview of thesis contributions.

Redundancies in ML algorithms make them impractical for large-scale datasets. Pruning these redundant computations with minimal loss in the quality of the downstream tasks improves the computational efficiency of the task. This dissertation focuses on three unsupervised machine-learning tasks: clustering [14], [15], anomaly detection [2], and model update, refer to figure 1.1 for details. These algorithms have Isolation Forest working at the core of these tasks. All the thesis contributions optimizations focus on the Isolation Forest ($iForest$) structure. In particular, the work focuses on the following three scenarios.

1. When an application needs all-pair distances: How to compute all-pair distances

faster by optimizing the order of distance computation?

2. When an application needs only a subset of all-pair distances: How to identify the required subset of all pairs?

3. When new data causes concept drift: How to quickly update the model?

We focus on the redundancies in Mass-Based Distance (MBD) computations using Isolation forest [4]. MBD is used for downstream tasks of Mass-based clustering and nearest neighbor-based anomaly detection [1]. Model performance degrades when new data causes concept drift. We also explore how to update the model efficiently to deal with this concept drift. Please refer to the summary of the best existing solutions and their bottleneck in table 1.1.

Problem	Best Existing Solution	BottleNeck
ML application needs all-pair MBD	Naive all-pair MBD [1] computation using Isolation Forest	Pair-major order MBD computation requires 2 passes over iTree for each pair.
ML applications needs a subset of all-pairs MBD	Naive all-pair MBD [1] computation using Isolation Forest	Compute Unnecessary MBD.
ML applications require model updates.	Incremental Algorithms using iForest [16], [17], [18], [19]	Make coarse grained updates using new data.

Table 1.1: Summary of existing approaches and their bottlenecks.

1.3 Research Contributions

In this dissertation, we make three contributions to improve the efficiency of Isolation Forest-based unsupervised ML algorithms. Please refer to Table 1.2 for a summary of our solutions to the bottlenecks mentioned in Table 1.1

1.3.1 Contribution 1: Fast Computation of All-Pair Mass-based Distances ($fMBD$)

Our first contribution focuses on a specific data-dependent distance measure: Mass-Based Distance (MBD). MBD computation is based on a well-known data structure,

Problem	Our solution [**PUBLISHED AT**]	Downstream Task	Results
ML Application needs all-pair MBD	$fMBD$ [**UNDER REVIEW**]: Tree-major order of MBD computation requires a single pass over each iTree for all pairs.	Clustering & Anomaly Detection	Upto 5X speedup. No loss in quality.
ML Application needs a subset of all pairs MBD	$sMBSCAN$ [**CIKM 2022**]: 1. Identify important pairs 2. Speed up each MBD computation	Clustering	Upto 53X speedup. Upto 22X memory saving. No loss in quality.
ML Applications require model updates.	$I^2Forest$ [**CoDS-COMAD24**]: Make fine grained updates to the model.	Anomaly Detection	Upto 85X speedup. Minimal loss in quality.

Table 1.2: Summary of thesis contributions.

Isolation Forest. We observe that clustering and anomaly detection methods that use MBD require all-pair distance computation. This distance computation accounts for more than 93% of the running time. We develop a method fast MBD ($fMBD$) to speed up all-pair MBD computations. Our method computes the same MBD without any error or approximation. We have performed experiments using popular real-world datasets (12 for clustering and 5 for anomaly detection). With $fMBD$, we achieve a speedup of 2X to 5X without any loss of performance over both tasks.

1.3.2 Contribution 2: Scaling Up Mass-Based Clustering ($sMBScan$)

Our second contribution addresses the problem of scaling up the mass-based clustering paradigm to handle large datasets. The existing algorithm MBScan computes and stores all pairwise distances, resulting in quadratic time and space complexity. However, we observe that mass-based clustering requires information about only a tiny fraction of all possible data point pairs. We propose three optimizations to MBScan for quickly finding such pairs and computing their distances. We empirically evaluate our work on ten real-world and synthetic datasets. Our experiments show that our approach results in fast and memory-efficient clustering with no loss in the quality of clusters.

1.3.3 Contribution 3: Incremental Isolation Forest ($I^2Forest$)

Our third contribution addresses the problem of updating Isolation Forest in response to new data. The update becomes even more critical when the new data causes concept drift. A lazy solution is to keep using the old model. However, it will result in inferior performance. An aggressive solution is to rebuild the model from scratch. This solution will improve the model performance at the cost of time spent in retraining the model. We design an incremental solution that quickly updates the existing model to match the performance of the aggressive solution. We have chosen anomaly detection as the downstream task to evaluate the quality of the updated model. Our incremental approach results in a minimal loss in the model's performance for the downstream task while significantly reducing the running time.

1.4 Outline of the Thesis

The thesis comprises six chapters. The Isolation Forest and related work is described in Chapter 2. We also survey the domain-specific applications and variants of Isolation Forest in the same chapter. The following three chapters describe each of our research contributions: $fMBD$ (Chapter 3), $sMBSCAN$ (Chapter 4), and $I^2Forest$ (Chapter 5). In Chapter 6, we conclude the thesis and discuss possible future work.

2

Background and Literature Study

This chapter starts with a brief summary of existing clustering and anomaly detection algorithms. Later it presents details of the Isolation Forest necessary to understand the thesis contributions. Isolation Forest (iForest) is a randomized full binary tree-based data structure introduced by Liu et al. in 2008 for anomaly detection [4]. iForest is an ensemble of full binary trees known as Isolation Trees (iTree) because they can isolate anomalies from the rest of the data. The underlying idea of iForest is to isolate anomalies by recursively partitioning data points. Anomalies being "few and different" tend to isolate in lesser splits than the normal data points. Isolation forest utilizes the concepts of sub-sampling and randomization and provides a near-linear time complexity and a small memory footprint requirement.

2.1 Clustering and Anomaly Detection Literature

This section summarizes the existing literature in the domain of clustering and anomaly detection algorithms. This brief summary helps to understand the domain of clustering and anomaly detection and will lay a foundation for the efficient algorithms focused on Mass(or density) based clustering and Isolation forest-based anomaly detection.

2.1.1 Clustering Algorithms

Clustering algorithms can be broadly categorized into several paradigms based on their approach to grouping data. Partitioning-based methods like k-Means [20] divide data into a fixed number of clusters by optimizing a similarity criterion, whereas hierarchical clustering [21] builds a tree-like structure without requiring a predefined number of clusters. Density-based techniques, such as DBSCAN [15], detect clusters based on dense regions and are effective in handling noise and arbitrary shapes. MBSCAN [1], which introduces mass-based distances, is a mass-based clustering algorithm that extends DBSCAN by handling multiple density levels within the same dataset. It dynamically adjusts the neighborhood radius to detect clusters of varying densities while effectively identifying noise points. Grid-based clustering, exemplified by CLIQUE [21], partitions the data space into grid cells, making it efficient for large and high-dimensional datasets. Model-based approaches, like Gaussian Mixture Models [22], assume an underlying probability distribution and optimize parameters to fit the data. Spectral clustering [23] utilizes graph-based techniques to find complex cluster structures, often performing well on non-linearly separable data. For high-dimensional data, subspace clustering methods such as PROCLUS [24] identify clusters within relevant feature subsets. Recently, deep learning-based clustering has emerged, leveraging neural networks for feature learning and clustering large-scale, high-dimensional data [25]. Each paradigm has strengths and limitations, making it suitable for varying requirements.

2.1.2 Anomaly Detection Algorithms

Anomaly detection algorithms can be broadly categorized into several paradigms based on their underlying approach to identifying deviations from normal patterns. Statistical methods [26] assume a probabilistic distribution of normal data and detect anomalies as low-probability instances, with techniques such as Gaussian models and hypothesis testing. Distance-based methods [27] identify anomalies by measuring their distance from other points, with k-Nearest Neighbors (k-NN) and DBSCAN being common choices. Density-based approaches, such as Local Outlier Factor (LOF) [28], estimate local data

density and flag points in sparse regions as anomalies. Clustering-based techniques [29] utilize clustering algorithms like k-Means and DBSCAN, where anomalies are identified as points that do not belong to any dense cluster or are far from cluster centroids. Classification-based approaches, including One-Class SVM [30] train models to distinguish normal from anomalous instances using labels. Isolation Forest [4], a model-based approach that learns the anomaly structure without using labels, unlike other approaches that learn the structure of normal datapoints. Reconstruction-based methods, particularly autoencoders [31] and PCA [32], rely on the idea that anomalies have higher reconstruction errors when mapped back from a lower-dimensional space. Graph-based approaches [33] detect anomalies in networked data by analyzing structural irregularities. Finally, deep learning-based techniques, such as LSTMs for time-series data [34] and self-supervised learning [35], leverage neural networks to model complex data distributions and detect outliers. The choice of technique depends on data characteristics, interpretability needs, and computational constraints.

2.2 Isolation Forest Construction

$iForest$ is an ensemble of $iTrees$. Each $iTree$ is a proper binary tree. $iForest$ requires two input parameters: the number of $iTrees$ (K) and the sub-sample size ($|S|$). $iForest$ has linear time complexity and thus scales well for large datasets. For a given input dataset D_i at timestamp T_i, to build an $iForest$ F_i consisting of K $iTrees$, each $iTree$ is built independently. For each $iTree$, a random sample of size $|S|$ is selected from D_i. The maximum height of the $iTree$ is restricted to $\lceil log_2(|S|) \rceil$.

Please refer to Figure 2.1. The $iTree$ creation starts with the root node. Initially, all the sampled data points belong to the root node. The root node is partitioned into left and right children. The splitting criteria for the root node consist of two parts: attribute and value. The attribute for splitting is chosen randomly out of all attributes. Let the selected attribute for splitting be A. Amongst all the data points belonging to the root node, let A_{max} and A_{min} be the maximum and minimum values observed for the attribute A. The value for splitting is chosen randomly between A_{max} and A_{min}.

This splitting procedure is carried out recursively till we reach the maximum height of $\lceil log_2(|S|) \rceil$ or the node contains only a single data point. The generated tree structure and each split criteria are used as the $iTree$.

After the tree construction, the whole dataset D_i is inserted into the tree. Each data point is moved to the appropriate leaf node in the tree. Please note that during this phase, the structure of the tree is not altered. A data point belongs to each node on the path from its leaf node to the root node. The mass of any node in the tree is the number of data points that belong to the node. For example, in any Isolation Tree, the mass of the root node is always n as the whole dataset D belongs to the root node. Isolation Forest was initially designed for anomaly detection task. Each node of the $iTree$ is associated with a hyperrectangle in the feature space. Typically, anomalies belong to the shallow leaf nodes of $iTree$. Hyperrectangles corresponding to shallow leaf nodes are considered as anomalous regions. Rest all regions are considered as normal.

2.3 Isolation Forest: A tool for pattern identification

Isolation Forest performs recursive random partitioning of a dataset and arranges the whole dataset such that similar points fall in one partition and otherwise in different partitions. In other words, an isolation forest creates imperfect random groups of points in the datasets. Application of isolation forest to downstream tasks like anomaly detection, density estimation, and (dis)similarity computation shows that these partitions provide insights into the data patterns and become an efficient tool for unsupervised machine learning.

2.3.1 Anomaly Score Computation

Isolation Forest was originally designed for anomaly detection. Liu et al., in the original isolation forest paper [4], call it a model-based anomaly detection method that isolates the anomalies in the shallow leaf nodes of the tree. The authors define an anomaly score to quantify the isolation measure of each point. The anomaly score is normalized between zero and one. The anomaly score of a point is a function of path length from

the root of the iTree to the associated leaf of the point. Please refer to Algorithm 3 in [4] for details about how to compute the anomaly score. Anomalous points have high anomaly scores close to one, and normal points have low anomaly scores close to zero. A threshold is decided using domain knowledge about the dataset to classify the data points. Any point x with an anomaly score higher than the threshold is predicted as an anomaly. The ideal threshold suggested is 0.5, though it varies significantly for different datasets.

2.3.2 Data Dependent Distance Computation

Later, Ting et al. in 2016 utilized isolation forest for data-dependent distance computation [1]. The distance is called Mass-Based Distance (MBD) and changes with a change in the probability mass of the region. MBD and other data-dependent distance measures are closer to the human notion of similarity [36] between two objects. Mass-Based Distance computation using isolation forest requires inserting complete dataset D in each iTree to prepare for distance computation. The total number of points that pass through to a node is known as the size or mass of the node. For a dataset D, consider an Isolation Forest F consisting of K Isolation Trees $T_1, T_2 ... T_K$. For a data point pair consisting of data points x and y, the MBD is calculated as defined in equation 2.1,

$$MBD(x, y) = \frac{1}{K} \sum_{i=1}^{K} MBD_i(x, y) \tag{2.1}$$

$$MBD_i(x, y) = Mass_i(LCA(Leaf_i(x), Leaf_i(y))) \tag{2.2}$$

where $MBD_i(x, y)$ is the mass-based distance computed using only one Isolation Tree T_i, $Leaf_i(x)$ returns the id of the leaf node that data point x belongs to in the Isolation Tree T_i, $LCA(Node1, Node2)$ returns the id of lowest common ancestor of nodes $Node$ and $Node2$, and $Mass_i(node)$ returns the mass of $node$ in Isolation Tree T_i.

2.4 Domain Specific Applications of Isolation Forest

Isolation Forest, known for its simplicity and effectiveness, has many applications across diverse domains. Its capability of isolating *few and different* data points in datasets makes it particularly useful in various scenarios. For instance, Isolation Forest is employed in Network Monitoring and Network Security systems to identify unusual traffic patterns that may indicate network attacks or cybersecurity threats. Similarly, it monitors and detects rare activities in Smart Homes or Industrial Internet of Things (IIoT) applications and identifies compromised activities in smart grid and blockchain-based intelligent systems. Isolation Forest has been extensively explored in the biomedical field to detect rare diseases or anomalies in medical data. Its efficacy extends to the finance and banking sectors, where it has been effectively utilized to detect fraudulent transactions. Modern agricultural practices also benefit from Isolation Forest through disease detection and monitoring. Additionally, it finds application in a wide spectrum of engineering and manufacturing industries, including physics processes, semiconductor manufacturing, wind energy harvesting, power generation plants, and marine engineering processes, for monitoring equipment health and predicting failures by detecting unusual environmental conditions. Furthermore, in the retail and e-commerce sectors, Isolation Forest is employed to observe customer behaviors and monitor sales patterns, thereby detecting unusual patterns that can benefit businesses. Intelligent transportation systems utilize Isolation Forest to detect traffic scene anomalies and identify anomalous trajectories.

Moreover, Isolation Forest has been explored as a sub-component in combination with neural network components for various NLP and vision applications related to anomaly detection. Beyond these applications, Isolation Forest is also used for monitoring computing servers and applications to detect unusual activity or performance issues. Overall, the versatility and robustness of Isolation Forest make it a valuable tool in numerous fields, demonstrating its extensive applicability and impact.

2.4.1 Network Monitoring and Security

Network monitoring and security aspects have been extensively researched using Isolation Forest. Specifically, intrusion detection has been addressed with data from various network layers. For example, a study by Karev et al. [37] utilizes HTTP log data to detect novel threats using Isolation Forest. Nadler et al. [38], and Ahmed et al. [39] in different research employ Isolation Forest to analyze DNS logs, identifying DNS tunneling and low throughput data exfiltration malware and subsequently denying requests to malicious domains. Another study by Siddiqui et al. [40] focuses on detecting cyber attacks and generating explanations with human involvement.

Furthermore, researchers in [41],[42],[43],[44] have explored the application of Isolation Forest for insider threat detection. In the context of cloud computing, Calheiros et al. [45] addresses cloud monitoring and anomaly detection in cloud data centers. Research by Vartouni et al. [46] also leverages Isolation Forest to detect various web attacks by analyzing features extracted from HTTP traffic using autoencoder-based networks. Moreover, Ren et al. investigated isolation forest for data sampling by removing outliers and subsequently performing hybrid data optimization for machine learning applications [47]. All the mentioned research highlights the versatility of Isolation Forest in enhancing network security and monitoring across different contexts.

2.4.2 IoT Applications

The Internet of Things (IoT) and edge devices have increasingly utilized Isolation Forest across various applications. In smart home environments, Isolation Forest analyzes pyroelectric infrared sensor data for detecting abnormal activities and novelties [48], [49]. Additionally, it is employed for security monitoring [50]. In industrial IoT contexts, Isolation Forest has been explored for attack detection within industrial control frameworks [51], predictive maintenance of sensors [52], and noise reduction from sensor data [53]. Furthermore, Isolation Forest has been applied to detect botnets in IoT and edge devices using a one-class classification approach by Bezerra et al. [54]. Researchers have also utilized Isolation Forest to analyze Twitter bot networks for detecting bot

behaviors [55] and for anomaly detection in smart audio sensors deployed in IoT edge devices [56]. Another application of isolation forest in combination with PCA to detect data integrity assault in smart grid communication networks was explored by Ahmed et al. in 2019 [57]. These are a few examples of Isolation Forest enhancing security and monitoring capabilities in IoT and edge computing environments.

2.4.3 Blockchain Security

Researchers have explored security and threat detection in Blockchain networks utilizing Isolation Forest. For instance, Podgorelec et al., in their study [58], explored automated signing and anomaly detection in blockchain transactions within the context of the Ethereum public network. Maskey et al. [59] implement Isolation Forest for outlier detection in a blockchain-based intelligent transportation system, ensuring security and data integrity in smart cities.

2.4.4 Medicine and Bioinformatics

Isolation Forest has been utilized extensively by researchers in the biomedical field for various applications. For instance, it has been employed to observe rare or anomalous patterns to identify specific discrepancies in genome sequence datasets [60],[61]. It has also been used to classify chest X-ray images for identifying COVID-19 cases [62] and as a defense strategy against backdoor attacks in federated GAN networks for medical images [63]. Additionally, Isolation Forest has been applied to detect Medicare fraud [64]. These examples highlight the diverse applications of Isolation Forest in this important domain.

2.4.5 Finance and Banking

Isolation Forest has been utilized in various applications within the financial technology sector. For instance, it has been employed to generate user suspicion rankings for detecting fraudulent activities in fund movements and Ripple network transactions involving digital cryptocurrency [65]. In the fintech industry, isolation forest serves as an

unsupervised anomaly detection approach in real-time transaction fraud detection systems [66]. Furthermore, in credit card fraud detection, isolation forest combined with supervised machine learning techniques has proven effective in addressing this crucial issue [67].

2.4.6 Agriculture

Isolation Forest has been explored for various agricultural applications. For instance, Deng et al. [68] employed Isolation Forest to detect diseases in citrus orchards using high-dimensional data captured by a UAV monitoring system. A study by Cejrowski et al. [69] utilized Isolation Forest with contrastive autoencoders to identify hazardous situations in honey-bee colonies. Additionally, Kansara et al. [70] demonstrated that Isolation Forest was the most effective outlier detection algorithm for data cleaning in the Indian Ayurvedic plant organ image dataset.

2.4.7 Engineering Applications

Isolation Forest has been widely explored in different engineering applications. In physics research, it is employed to identify new physics events and anomalies at Large Hadron Colliders [71],[72],[73]. In power engineering, isolation forest methods have been utilized to segregate multi-source particle discharge signals in power equipment [74], detect early-stage malfunctions in combined cycle power plants [75], and enhance deep learning approaches for wind power prediction systems[76]. Monitoring and predicting faulty conditions in marine machinery systems have also benefited from isolation forest techniques [77]. Isolation forest algorithm has been applied to preprocess noisy data for generating wind power curves by Wang et al. [78] to achieve effective anomaly detection and fault discrimination in wind turbine gearboxes by Du et al. [79]. It detects anomalies in optical emission spectroscopy data from semiconductor manufacturing processes [80], thereby improving interpretability in high-dimensional datasets [81]. In the domain of automated power consumption systems, isolation forest has been employed for outlier removal before electricity price prediction [82] and for detecting anomalies in household power consumption trends [83].

2.4.8 Miscellaneous

Isolation forest has been applied across various domains beyond the scenarios mentioned above. For instance, it detects fake reviews based on temporal patterns in product review records on e-commerce platforms [84]. Anomalous user behavior over enterprise datasets is also identified using isolation forest techniques . Furthermore, it has proven effective in detecting disorientation in GPS trajectories of elderly individuals with cognitive disorders [85]. In transportation systems, isolation forest is utilized to identify abnormal events in intelligent driver assistance systems [86]. Additionally, it is studied for anomaly detection in High-Performance Computing (HPC) systems, highlighting its versatility and efficacy across various fields [87]. Sarria et al. [88] present a remote sensing application of isolation forest for evaluating class separability for land cover classification approaches.

2.5 Variants of Isolation Forest for Anomaly Detection

The original Isolation Forest algorithm has multiple components that introduce randomization. It performs random sampling to choose a sample from the data. It also randomly chooses the splitting attribute followed by a random choice of splitting value. This randomization makes it resource-efficient. Researchers have explored different directions to improve the effectiveness of Isolation Forest and make changes to this randomized mechanism or modify the anomaly score computation.

Literature has variants of Isolation Forest where the splitting criterion of a node is modified by adopting different concepts. Hariri et al. [89] have replaced the random axis-parallel splits with random non-axis parallel splits that include more than one attribute in the splitting decision. Another Isolation Forest variant by Liu et al. [90] optimizes the non-axis parallel splits and proves to be effective for both scattered and clustered anomalies. Tokovarov et al. [91] presented a probabilistic choice of split criteria to generalize the random split criteria of Isolation Forest. Their Isolation Forest variant reduces the possibility of poor-quality isolation trees in the ensemble.

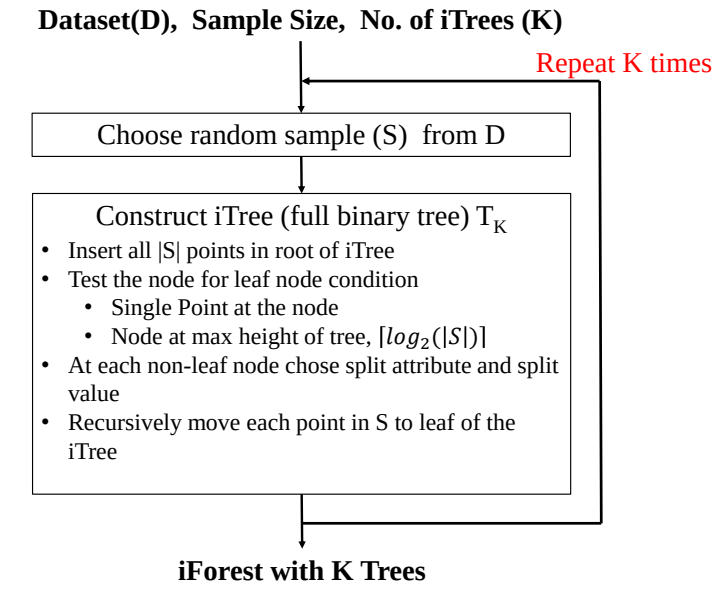
Another set of Isolation Forest variants choose the split criteria in a more informed way

using the distance between data points. These modifications are time-consuming due to the requirement of the underlying distance computations. For Instance, Karczmarek et al. [92] proposed a k-means-based Isolation Forest that predicts the number of divisions at each node of the tree using k-means clustering. This variant performs better splits in comparison to the vanilla Isolation Forest. Galka et al. [93] used the Minimal Spanning Tree (MST) concept to merge the set of points and construct the Isolation Trees in a bottom-up manner.

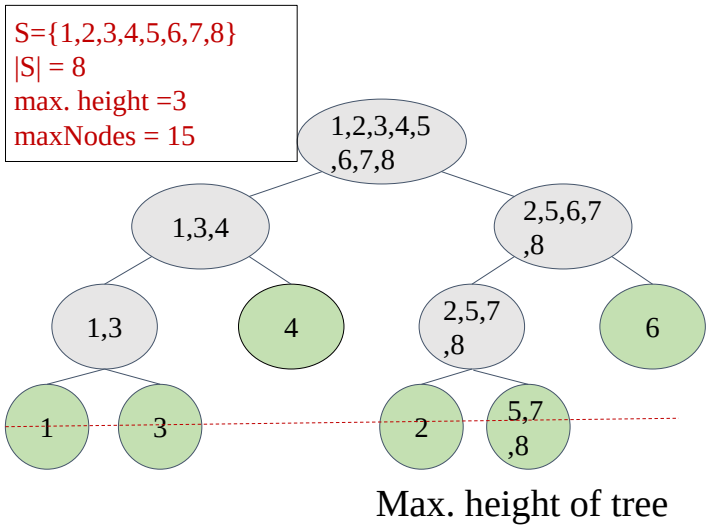
Some variants of Isolation Forest utilize hashing concepts. They use hashing to decide the number of splits in every node of the tree. Zhang et al. [94] explored Locality Sensitive Hashing (LSH) for splitting a node. Later, Xiang et al. [95] used order-preserving hashing for splitting the nodes and providing robust anomaly detection.

Instead of changing the Isolation Forest construction, some variants alter the anomaly score computation. For instance, Aryal et al. [96] used relative mass between points as an anomaly score associated with the points and tried to improve the anomaly detection quality. Similarly, Mensi et al. [97] explored a variety of neighborhood-based weighted scores to points for anomaly detection using Isolation Forest.

To summarize, Isolation Forest is a useful and efficient data structure for anomaly detection. It is well-adopted in various real-world applications. Isolation Forest is an active topic of research.



((a)) iForest construction steps



((b)) iTree with an example sample set.

Figure 2.1: Steps of iForest Construction and an example iTree.

3

Fast Computation of All-Pairs Mass-based Distance

Chapter Highlights

- Computing Mass Based Distance (MBD) for all data point pairs is an operation required for many machine learning algorithms.
- This work focuses on how to speed up the distance computation without introducing any approximation or error.
- We propose fast MBD ($fMBD$) algorithm to speed up the all-pairs MBD computation.
- We evaluate $fMBD$ for clustering and anomaly detection tasks.
- This research is under review in a peer-reviewed conference.

3.1 Abstract

Given a dataset with n data points, there are $\binom{n}{2}$ possible data point pairs. Many ML tasks require distance computation for all these data point pairs. This chapter focuses on a specific data-dependent distance measure: Mass-Based Distance (MBD). MBD

computation is based on a well-known data structure, Isolation Forest. We observe that clustering and anomaly detection methods that use MBD require all-pairs distance computation. This distance computation accounts for more than 93% of the running time. To speed up the all-pairs MBD computations, we develop a method fast MBD ($fMBD$). Our method computes the exact same MBD without any error or approximation. We have performed experiments using popular real-world datasets (12 for clustering and 5 for anomaly detection). With $fMBD$, we achieve a speedup of 2X to 5X without any loss of performance over both tasks.

3.2 Introduction

This chapter presents a fast and exact algorithm for the all-pairs distance computation problem, using the Mass-Based Distance (MBD) [1] as the distance measure. Given a dataset of n data points, the all-pairs distance computation is the problem of computing distance for all $\binom{n}{2}$ data point pairs. The problem is compute intensive as it requires distance computation for $O(n^2)$ pairs. The all-pairs distance computation can become the bottleneck for the downstream task if it consumes the majority of the running time. One possible solution is to approximate the distance computation quickly. However, such approximations introduce errors and affect the quality of downstream tasks. Our goal is to compute the exact distance while reducing the running time of the algorithm. MBD computation is based on a well-known data structure, Isolation Forest (iForest) [4]. iForest was initially designed for the task of anomaly detection. iForest is a collection of Isolation Trees (iTree). Each iTree computes the distance for a data point pair independently. MBD is the average distance across all iTrees. Each iTree is a full binary tree built using a random sample of data. Within an iTree, each data point belongs to a leaf node and all other nodes along the path from the root node to that particular leaf node. The mass of a node in an iTree is the number of data points that belong to the node. Distance between any two points in an iTree is the mass of the lowest common ancestor (LCA) node of the two leaf nodes corresponding to the two data points. MBD is a data-dependent distance measure. The distance between

any two data points depends on the other data points in the dataset. MBD is shown to have better quality than data-independent distance measures such as the Euclidean distance[1].

The naive method for the all-pairs distance problem computes the distance for each data point pair. Please refer to Algorithm 1. We have to visit each iTree $\binom{n}{2}$ times. For MBD, it involves two steps. First, locate the leaf nodes of each data point. Second, compute the LCA of two leaf nodes. We cannot speed up the individual MBD computation. However, while computing all-pairs distance, there are two possible opportunities for optimization. First, we can visit each iTree only once and compute the MBD component for all data point pairs. Second, we can avoid the overhead of repeatedly locating the leaf node for all data points. We propose an algorithm fast MBD (fMBD) that facilitates both these optimizations to speed up the all-pairs distance computation. For complete reproducibility, all our code and datasets are available publicly on the Web.[1]

Speeding up all-pairs distance computation will matter only if it accounts for a significant part of the total running time of a task. We experiment with two such tasks: Density-based clustering and K-nearest neighbor-based anomaly detection. For both tasks, all-pairs distance computation accounts for more than 93% of the running time. With fMBD, we demonstrate that both tasks can achieve a speed-up of up to 5X. We have experimented with a variety of real-world popular datasets for these tasks. There is no loss in output quality with fMBD as the distance computation is exact. The main research contribution of our work is to provide a fast algorithm for all-pairs distance computation problem.

3.3 Related Work

Density Peak Clustering (DPC) [14] is a well-known and popular clustering algorithm. It works in five steps. First, it computes all-pairs distances. Second, it computes the density of each data point as the number of other data points having a distance below a threshold to it. Third, it locates the nearest neighbor with a higher density (NNHD)

[1] https://github.com/nidhiahl/fMBD

for each data point. For each point, it also records the distance to NNHD. The point with the highest density will not have any NNHD. For such a data point, the distance to NNHD is considered infinity. Number of clusters K is a parameter to the algorithm. Fourth, K data points are chosen as seeds that represent the K cluster. Seeds are the points that have high density and high distance to NNHD. Fifth, each data point is assigned to the cluster of its NNHD. The original DPC algorithm uses Euclidean distance for density computation. In our work, we have replaced the Euclidean distance with MBD.

K nearest neighbor (KNN) based anomaly detection [2] is a three step process. First, it computes all-pairs distances. Second, it records the distance to the Kth nearest neighbor for each data point. Third, a data point is considered as an anomaly if its KNN distance is above a threshold. Defining an appropriate threshold is tricky and requires domain knowledge.

There are three steps in the naive solution for the all-pairs distance problem while using the MBD [1]. Please refer to Algorithm 1.

Algorithm 1 naive All-Pairs MBD(iForest, M, D)

```
1: //We assume that the iForest is already constructed.
2: //Maxtrix M is an nXn matrix that stores all-pairs distances.
3: Insert all data points in D into each iTree of iForest.
4: Initialize all elements of M with zero.
5: for each data point x_i (i=1 to n) in D do
6:    for each data point x_j (j=i+1 to n) in D do
7:       for each iTree t in iForest do
8:          leaf_{x_i} = getLeaf(x_i,t)
9:          leaf_{x_j} = getLeaf(x_j,t)
10:         lcaNode = getLCA(leaf_{x_i},leaf_{x_j})
11:         M[x_i][x_j] += lcaNode.mass/(n*T)
12:         M[x_j][x_i] += lcaNode.mass/(n*T)
13:      end for
14:   end for
15: end for
16: Output: M
```

Construct iForest: Please refer to Chapter 2 for the details about the iForest construction process.

Insert all the points: After iForest construction, all the data points are inserted

into each iTree to prepare for MBD computation. The preparation includes constructing a point-to-leaf mapping separately for every tree, which will be used in the MBD computation.

MBD computation(One pair at a time): MBD of a pair of points x and y is defined as the average of the individual $MBD_t(x, y)$ from all K trees; refer to equation 3.1.

$$MBD(x, y) = \frac{1}{K} \sum_{t=1}^{K} MBD_t(x, y) \tag{3.1}$$

Given an isolation tree t, $MBD_t(x, y)$ computation is a three-step process. First, map both x and y to their respective leaf nodes $leaf_x$ and $leaf_y$ using point-to-leaf mapping. Second, locate the Lowest Common Ancestor (LCA) node for both the leaf nodes. Third, the mass of the LCA node is the MBD for the given data point pair. MBD for the data point pair is the average MBD across all iTrees. MBD is normalized between 0 and 1 by dividing the mass of every node by the number of data points (n) in the dataset.

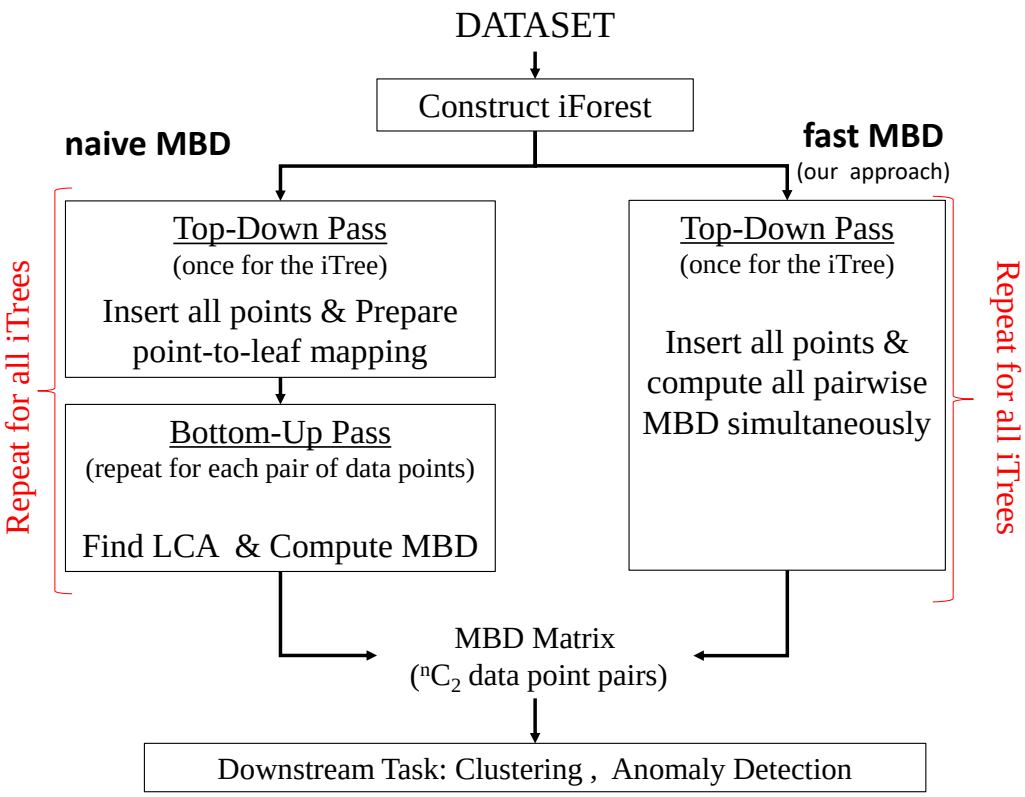


Figure 3.1: Overview of our approach fast MBD and Comparison with naive solution

3.4 fast MBD

Two opportunities exist to improve the naive solution for the all-pairs MBD distance problem. First, the naive solution computes the MBD for each data point pair separately. Please refer to the two nested loops on lines 5 and 6 of Algorithm 1. As a result, we end up visiting each iTree $\binom{n}{2}$ times (third nested loop on line 7 of Algorithm 1). We do not have to carry out the distance computation for each data point pair separately. We can visit each iTree only once and compute the MBD component for all the pairs in a single pass.

Second, the naive solution must repeatedly locate the leaf node for each data point (lines 8 and 9 of Algorithm 1). The naive solution pre-computes the leaf node for each data point in every iTree. The naive solution has to insert the whole dataset into every iTree to pre-compute the leaf node for each data point. We can compute the MBD for a pair only in a single top-down pass. It will avoid leaf node computation, and as a result, we do not have to insert the whole dataset into the iForest.

3.4.1 fastMBD Algorithm

Please refer to Algorithm 2. Our fastMBD algorithm visits each iTree only once and computes MBD for all data point pairs.

Algorithm 2 fMBD(iForest, M, D)

```
1: //We assume that the iForest is already constructed.
2: //Maxtrix M is an nXn matrix that stores all-pairs distances.
3: Initialize all elements of M with zero.
4: for (each iTree t in iForest) do
5:    fMBD_Tree(t,M,D)
6: end for
7: Output: M
```

When it visits each iTree, it performs a single breadth-first traversal of the tree (Algorithm 3). Initially, all data points are inserted into the root node of the tree (Line 1, Algorithm 3).

While performing the breadth-first traversal, fMBD distinguishes between the internal nodes (Algorithm 4) and lead nodes (Algorithm 5). While processing an internal node,

Algorithm 3 fMBD_Tree(t, M, D)

```
1: t.root.data = D // Insert all points in D to root of iTree t.
2: Create empty BFT //Queue for breadth-first traversal of nodes in iTree t.
3: BFT.enqueue(t.root)
4: while (BFT is notEmpty) do
5:    node = BFT.dequeue
6:    if (node is Internal Node) then
7:       fMBD_InternalNode(node,M,BFT)
8:    else
9:       fMBD_LeafNode(node,M)
10:   end if
11: end while
```

we first distribute the data points of the current node into the left and right children using the split criteria (Line 1 to 7, Algorithm 4). Now consider a data point pair (x, y) such that x satisfies the split criteria and y does not satisfy the split criteria at the current node. As a result, x will be assigned to the left child, and y will be assigned to the right child. Eventually, both x and y will end up in their respective leaf nodes $leaf_x$ and $leaf_y$. The current node will be the LCA for these two leaf nodes. Therefore, the MBD of pair (x, y) will be the mass of the current node (Lines 10 and 11, Algorithm 4).

Algorithm 4 fMBD_InternalNode(node,M,BFT)

```
1: for each data point x in node.data do
2:    if x satisfies node.splitCriteria then
3:       add x to node.left.data
4:    else
5:       add x to node.right.data
6:    end if
7: end for
8: for each x in node.left.data do
9:    for each y in node.data.right do
10:      M[x][y] += node.mass/(n * T)
11:      M[y][x] += node.mass/(n * T)
12:   end for
13: end for
14: BFT.enqueue(node.left)
15: BFT.enqueue(node.right)
```

However, we do not have to locate the leaf nodes explicitly, and we do not have to perform the LCA computation. This is the advantage of fMBD over the naive approach. While processing a leaf node, there are no children on the left or right. In such a scenario,

the MBD between data points belonging to the leaf node is simply the mass of the node (Lines 3 and 4, Algorithm 5).

Algorithm 5 fMBD_LeafNode(node,M)

```
1: for each data point x_i (i=1 to node.mass) do
2:    for each data point x_i (j=i+1 to node.mass) do
3:       M[x_i][x_j] += node.mass/(n * T)
4:       M[x_j][x_i] += node.mass/(n * T)
5:    end for
6: end for
```

3.4.2 Time Complexity Analysis

In this section, we discuss asymptotic time complexity for an individual iTree. For the iForest, the running time will be scaled by a factor of T (number of iTrees). For all-pairs MBD computation, the $nMBD$ method will first build the iTree in $S.logS$ time. Then, it will insert all the n datapoints into the iTree to locate the appropriate leaf node for each datapoint. The maximum height of any iTree is $logS$. Therefore, this step requires $n.logS$ time. Given a datapoint pair, the MBD computation in $nMBD$ requires two operations. First, locate leaf nodes for both datapoints. Second, compute LCA for both the leaf nodes. Both these operations, in total, require $2logS$ time. Hence the MBD computation for all $\binom{n}{2}$ datapoint pairs requires $O(2.n^2.logS)$ time. The total time taken by $nMBD$ is given in the equation 3.2. However, the asymptotic time complexity of $nMBD$ is $O(n^2.logS)$.

$$S.logS + n.logS + 2.n^2.logS \tag{3.2}$$

Similar to $nMBD$, our method $fastMBD$ requires an iTree constructed over sample S. This step requires $S.logS$ time. In the next step, $fastMBD$ passes the whole dataset through the iTree. Eventually, each datapoint reaches a leaf node. Any iTree has a maximum height of $logS$. This step requires $n.logS$ time. While passing the dataset through the iTree, $fastMBD$ also computes MBD for all $\binom{n}{2}$ pairs. In contrast to the $nMBD$, our method $fastMBD$ does not have to locate leaf nodes and LCA. Hence the MBD computation can be done in the constant time (Lines 10 and 11 of

Algorithm 4, Lines 3 and 4 of Algorithm 5). This step requires n^2 time. The total time taken by $fastMBD$ is given in the equation 3.3. However, the asymptotic running time of $fastMBD$ is $O(n^2)$.

$$S.logS + n.logS + n^2 \tag{3.3}$$

Comparing the total running time of $nMBD$ and the proposed algorithm $fastMBD$ given in equation 3.2, 3.3 respectively, $nMBD$ is slower by a factor of $2.logS$. However, asymptotically, the improvement is of the order $logS$, but the constant 2 is responsible for a speedup of 2X achieved by $fMBD$. Please note that for larger n values $logS$ is very small and does not contribute significantly to the speedup. Speedup given in table 3.2 also shows that above 10K datapoints, the speedup plateaus closer to $\sim 2X$, neglecting the contribution from $logS$, due to $logS << N^2$.

3.5 Experimental Evaluation

Fast computation of all-pairs distance is important only if it accounts for a significant chunk of the total running time of a task. We have considered two such tasks for our experimental evaluation: Clustering and Anomaly Detection. We have chosen the Density Peak Clustering (DPC) algorithm for the clustering task. Please refer to Section 3 for a summary of the DPC algorithm. We have replaced the Euclidean distance in the original DPC algorithm with the MBD. We refer to this clustering algorithm as Mass Peak Clustering (MPC). For the anomaly detection task, we have to use the MBD to calculate the K nearest neighbors. We refer to this anomaly detection algorithm as Mass-based Anomaly Detection (MAD).

We have experimented with twelve datasets for the clustering task. Please refer to table 3.1 for a summary of these datasets. These datasets are frequently used in clustering research. The dimensions of these datasets vary from 6 to 617. The number of data points in the dataset also varies from 351 to 70000. The number of clusters varies from 2 to 26. When we run MPC with the naive solution for all-pairs MBD distance computation, the MBD computation accounts for at least 98% of the total running time.

Table 3.1: Dataset details

dataset	Points	Dimensions	Clusters	% Time required for MBD computation
ionosphere [98]	351	33	2	98
wdbc [1]	569	30	2	99
control [98]	600	60	6	98
madelon [98]	2,600	500	2	99
satelite [99]	6,435	36	7	99
muskv1 [99]	7,074	166	2	99
thyroid [1]	7,200	6	3	98
isolet [99]	7,797	617	26	98
smartphone [98]	10,299	561	6	98
pendigits [1]	10,992	16	10	99
shuttle [99]	57,999	9	7	98
mnist [99]	70,000	16	10	98
dataset	Points	Dimensions	% Anomaly	% Time required for MBD computation
velocity [1]	229	20	34.06	96
mfeat [1]	410	649	2.44	94
parkinson [99]	756	754	25.3	93
tuandromd [99]	4,464	241	20.13	97
mutantp53 [1]	16,592	5,408	0.51	93

The ionosphere consists of radar data collected from a system of 16 antennas. The data is used to classify the good and bad radar returned from the ionosphere. WDBC is a breast cancer dataset with 30 features computed from 569 digitized images. The control dataset has 600 synthetically generated time-series images that are classified into six classes based on the time-series trends. The Satellite dataset is about multispectral satellite images classified into seven classes based on 36 features extracted from 3X3 images. Muskv1 is a 166-feature human-annotated dataset used for the classification of a molecule as musk or non-musk. The Smartphone is a human activity classification dataset recorded for 30 subjects with embedded inertial sensors. The Shuttle is a statlog dataset used to predict the space shuttle. It has seven different categories of the position of the shuttle recorded on the basis of sensor readings. MNIST is a digit dataset consisting of 10 different classes.

We have experimented with five datasets summarized in table 3.1. These datasets are

popular in the anomaly detection literature. The number of anomaly points in these datasets is expressed as a percentage of dataset size. It varies from 0.51% to over 34%. We can observe that all-pairs MBD computation is the main bottleneck in the running time for this task as well. MBD computation accounts for at least 93% of the running time. Both these tasks and datasets are an appropriate target for applying our fMBD algorithm.

The Parkinson's speech dataset consists of 754 distinct extracted features, which are utilized to determine whether a subject has Parkinson's disease or is healthy. Tuandromd is a malware detection dataset with 241 features to differentiate between malware and goodware. The Mutantp53 dataset is extracted from biophysical simulations of mutantp53 proteins having 5408 features used to predict transcriptional activity.

For all experiments, isolation forest construction is done for a sample size of 256 and the number of trees as 100, referring to [4]. The cut-off distance for MPC is chosen between 0.1% to 50% of the smallest distances for every dataset. Similarly, parameter K for anomaly detection is chosen between 1 to 50% of n, where n is the number of data points.

Please refer to table 3.2 for the comparison of running time. For both tasks, we can perform the all-pairs MBD computation in two ways: naive solution (nMBD) and our solution (fMBD). The table shows the running time of the task with each option for all-pairs MBD computation. Compute the speed of fMBD up over nMBD as the ratio of running time. For all the datasets, fMBD is faster than the nMBD. Even for large datasets, we achieve speed up close to 2X.

We compare the clustering quality using three evaluation measures: F-measure(higher the better), RandIndex(higher the better), and Entropy(lower the better). Anomaly Detection quality is compared using two evaluation measures: AUC(higher the better) and F1-score(higher the better). Please refer to table 3.3 for details. We observed that the quality of both the downstream tasks using $fMBD$ is precisely the same as that of using $nMBD$. This result is expected as our $fMBD$ algorithm computes the exact distance, and it does not affect the quality of downstream tasks.

Please refer to Figure 3.2. It shows variation in the speed-up of $fMBD$ with respect

Table 3.2: Running Time and Speedup results

dataset	Clustering Task		
	Running Time (in seconds)		Speedup over
	MPC_{nMBD}	MPC_{fMBD}	MPC_{nMBD}
ionosphere	0.95	0.24	**3.9**
wdbc	2.20	0.56	**3.89**
control	2.76	0.64	**4.31**
madelon	25.3	8.01	**3.16**
satelite	85.7	40.10	**2.14**
muskv1	96.39	45.17	**2.13**
thyroid	98.13	51.2	**1.92**
isolet	151.25	64.65	**2.34**
smartphone	203.26	99.14	**2.05**
pendigits	336.56	166.6	**2.03**
shuttle	7,745.14	4,139	**1.87**
mnist	12,826.25	6,497.3	**1.97**

dataset	Anomlay Detection Task		
	Running Time (in seconds)		Speedup over
	MAD_{nMBD}	MAD_{fMBD}	MAD_{nMBD}
velocity	0.28	0.069	**4.04**
mfeat	1.33	0.26	**5.05**
parkinson	3.53	0.99	**3.58**
tuandromd	107.18	34.86	**3.07**
mutantp53	1,175.39	534.2	**2.2**

Table 3.3: Clustering and Anomaly Detection quality Results

Datasets	Clustering Task		
	F-measure	RandIndex	Entropy
madelon	0.537	0.504	0.934
satelite	0.626	0.852	1.13
muskv1	0.71	0.755	0.408
thyroid	0.638	0.585	0.938
isolet	0.312	0.903	0.96
smartphone	0.561	0.808	1.001
pendigits	0.814	0.946	0.645
shuttle	0.559	0.562	1.587
mnist	0.341	0.74	0.70

Datasets	Anomlay Detection Task	
	AUC	F1 score
velocity	0.72	0.538
mfeat	0.997	0.8
parkinson	0.586	0.762
tuandromd	0.912	0.621
mutantp53	0.71	0.6

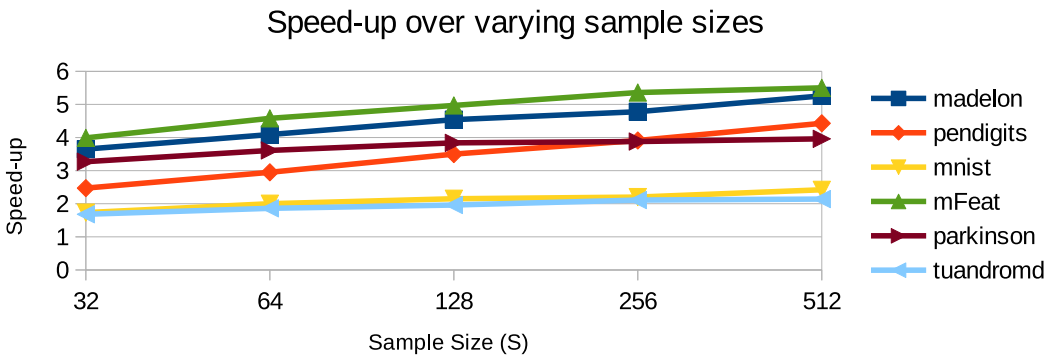


Figure 3.2: Variation in $fastMBD$ speed up over $nMBD$ with respect to sample size (S)

to change in the sample size(S). We have shown results for three datasets for each task. Results for other datasets are similar. We can observe that the speed up of $fMBD$ increases slowly with the sample size for all datasets. This monotonic increase demonstrates the $logS$ speed-up factor in the asymptotic time complexity of $fMBD$ over $nMBD$.

3.6 Conclusion and Future Work

This chapter presents a fast and exact algorithm, fMBD, for all-pairs distance computation. Our algorithm computes the exact MBD without introducing any error or approximation. We have evaluated our algorithm on two downstream tasks: clustering and anomaly detection. We have performed experimental evaluation using 16 popular datasets. Our algorithm fMBD consistently outperforms the naive solution for all-pairs MBD computation. Our work can be further improved by introducing an approximation to MBD computation with minimal loss in the quality of the downstream task.

4

Scaling Up Mass-Based Clustering

Chapter Highlights

- Mass Based Clustering algorithm needs to know the data point pairs that have distance below a threshold value.
- For a dataset with n data points, the number of such data point pairs is a tiny fraction of all possible $\binom{n}{2}$ pairs.
- The objective of this chapter is to compute distance for only a sufficient number of data point pairs, instead of all $\binom{n}{2}$ pairs.
- We have handled 100X larger data sets than existing Mass Based Clustering
- The proposed approach saves significant time and memory without any loss of clustering quality.
- This chapter is based on the publication "Scaling Up Mass-Based Clustering" presented in CIKM 2022.

4.1 Abstract

This chapter addresses the problem of scaling up the mass-based clustering paradigm to handle large datasets. The existing algorithm MBScan computes and stores all pairwise distances, resulting in quadratic time and space complexity. However, we observe that mass-based clustering requires information about only a tiny fraction of all possible data point pairs. We propose three optimizations to MBScan for quickly finding such pairs and computing their distances. We empirically evaluate our work on ten real-world and synthetic datasets. Our experiments show that our approach results in fast and memory-efficient clustering with no loss in the quality of clusters.

4.2 Introduction

Mass-based dissimilarity (MBD) is a data-dependent dissimilarity measure [1]. Its intuition is that a data point pair in a dense region has a lower similarity score than other data point pairs with the same inter-point distance in a sparse region. This similarity computation correlates well with the notion of similarity as judged by humans [36]. Using such a similarity measure, we can improve the performance of various important tasks such as clustering, anomaly detection, and classification. This chapter is focused on the clustering task using MBD.

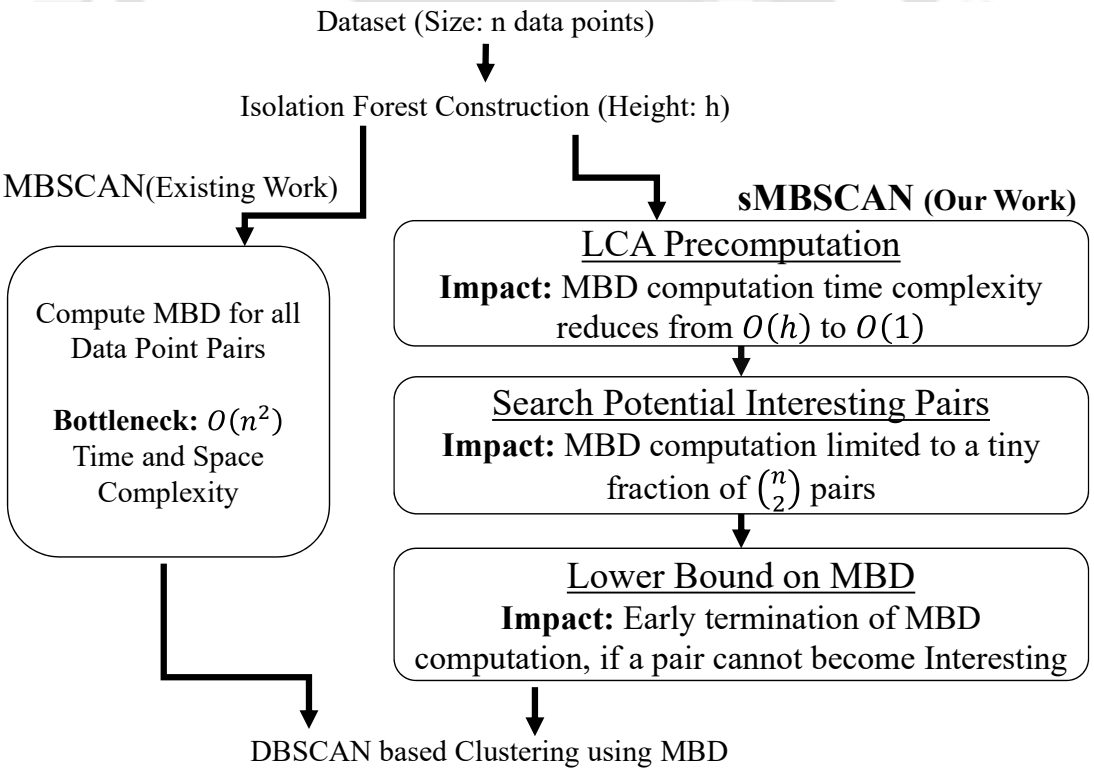
MBScan is a mass-based clustering algorithm with quadratic time and space complexity [1]. It is an improvement over the well-known density-based clustering algorithm DBScan [100]. MBScan replaces the distance measure in DBScan with MBD. MBScan uses a tree-based data structure, Isolation Forest, for computing MBD. An Isolation forest is a collection of independently constructed Isolation Trees [4]. MBScan creates higher quality clustering than DBScan and its variants [101][102]. Consider a dataset D with n data points. MBScan computes and stores MBD for all $\binom{n}{2}$ data point pairs. As a result, MBScan does not scale well to large datasets.

We observe that we do not need information about all $\binom{n}{2}$ data point pairs to perform mass-based clustering. It is sufficient to compute and store MBD only for a subset of pairs called Interesting Pairs. A data point pair is an Interesting Pair if and only if its

dissimilarity score is below a user-defined threshold and at least one of the points in the pair is not an outlier. We observe that only about three percent of data point pairs are Interesting Pairs for a wide variety of datasets. There is a significant opportunity to reduce running time and memory footprint for mass-based clustering.

In this chapter, we present an algorithm, sMBSCAN. It is a scalable algorithm for mass-based clustering. Our work has the following three specific research contributions. First, we propose a constant time method to compute MBD in an Isolation Tree. This contribution accelerates MBD computation for each data point pair. Second, we provide fast filtering criteria for selecting a superset of Interesting Pairs from all $\binom{n}{2}$ pairs. This contribution enables us to compute MBD only for a small subset of all possible $\binom{n}{2}$ pairs. Third, we propose an efficient lower bound on MBD. We use it for early termination of MBD for a data point pair if it cannot become an Interesting Pair. The overview of our work is presented in 4.1.

Figure 4.1: A comparative overview of scalable MBScan with MBScan



We empirically evaluate our optimizations by doing extensive experiments over ten

datasets (six real-world and four synthetic). The size of the datasets ranges from 2×10^3 to 1×10^5 data points. We have three evaluation criteria: running time, memory footprint, and clustering quality. Our optimizations provide a speed-up of up to 50X over MBScan. Memory footprint is up to 22X smaller than MBScan. There is no loss in clustering quality as measured with three measures: F1 measure, Entropy, and Rand Index. All our code and datasets are available publicly on the Web[1].

4.3 Related Work

Isolation Forest is described in Chapter 2.

Various distance (Euclidean, Manhattan, Hamming, Minkowski, and others) and similarity measures (Dot Product, Cosine Similarity, Jaccard Index, and others) are extensively used in various fields such as Information Retrieval, Machine Learning, and Data Mining. However, most of these distance and similarity measures are data-independent. The distance or similarity between a pair of data points does not depend on other data points in the dataset[103]. In contrast, MBD is a data-dependent dissimilarity measure. Consider two different data point pairs $P1$ and $P2$. The inter-point distance measured with data-independent distance measures such as Euclidean distance is the same for both the pairs $P1$ and $P2$. However, $P1$ is located in a dense region, and $P2$ is located in a sparse region. In such a scenario, the MBD will assign a higher dissimilarity score to $P1$ than $P2$. This intuition of dissimilarity computation correlates well with the human judgment of similarity[36]. For example, consider two shirts with blue color, but each has a different shade of blue color such as navy blue and sky blue. These two shirts will be considered more similar in a collection of clothes having a variety of colors. However, these two shirts will be perceived as less similar in a collection of only blue-colored clothes.

MBD computation can be done efficiently using Isolation Forest. For a dataset D, consider an Isolation Forest F consisting of k Isolation Trees $T_1, T_2 ... T_k$. For a data

[1] https://github.com/nidhiahl/sMBSCAN

point pair $P1$ consisting of data points x and y, the MBD is calculated as

$$MBD(P1) = \frac{1}{k}\sum_{i=1}^{k} MBD_i(P1) \tag{4.1}$$

$$MBD_i(P1) = MASS_i(LCA(LEAF_i(x), LEAF_i(y))) \tag{4.2}$$

where $MBD_i(P1)$ is the mass-based dissimilarity computed using only one Isolation Tree T_i, $LEAF_i(x)$ returns the id of the leaf node that data point x belongs to in the Isolation Tree T_i, $LCA(NODE1, NODE2)$ returns the id of the lowest common ancestor of nodes $NODE1$ and $NODE2$, and $MASS_i(NODE)$ returns the mass of $NODE$ in Isolation Tree T_i.

Density-based clustering is a popular clustering paradigm because of its ability to find clusters of arbitrary shapes. DBScan[100] is the most well-known clustering algorithm in this paradigm. It works in three steps. The first step computes $\epsilon - neighborhood$ of each data point. In other words, for each data point, it finds other data points within the distance of ϵ. The second step marks each data point with one of the three labels: CORE, OUTLIER, and NON-CORE. A data point is CORE if its $\epsilon - neighborhood$ has more than $minPts$ data points. Both ϵ and $minPts$ are parameters to the DBScan algorithm. A data point is an OUTLIER if it is not CORE, and there is no CORE data point in its own $\epsilon - neighborhood$. The remaining data points are labeled as NON-CORE. In the third step, DBScan begins by considering each CORE data point as a separate cluster. It merges clusters using the density reachability property. NON-CORE data points are assigned to the nearest cluster. A data point pair can affect the DBScan clustering only if its distance is less than ϵ and at least one of the points in the pair is CORE or NON-CORE. OUTLIERS are discarded, and they are not part of any cluster.

MBScan[1] adopts DBScan to utilize MBD. It works simply by replacing the distance definition in DBScan with MBD. MBScan is shown to have superior quality of clustering than DBScan and its variants such as OPTICS[102] and SNN[101] clustering. However,

MBScan computes and stores MBD for all data point pairs. As a result, MBScan has $O(n^2)$ time and space complexity while handling a dataset with n data points. It severely limits the scalability of MBScan. For example, for the WORMS2d dataset with 105600 data points, MBScan requires approximately 16 hours and 43 GB of memory to perform clustering.

4.4 scalable MBScan

We can scale up the mass-based clustering in two ways. First, we can accelerate the MBD computation for each data point pair. Our LCA Pre-computation optimization achieves this goal. Second, we can reduce the number of data point pairs for which we compute MBD. Ideally, we should compute MBD only for Interesting Pairs. Our other two optimizations: Potential Interesting Pairs and Lower Bound on MBD, achieve this goal.

4.4.1 LCA Pre-computation

Please refer to equations 1 and 2. Each MBD computation requires us to find the lowest common ancestor (LCA) of two leaf nodes. It requires $O(h)$ time, where h is the height of the tree. For a given data point pair, we have to repeat the LCA computation across all k Isolation Trees. Thus, each MBD computation requires $O(k.log_2(|S|))$ time. If we can compute LCA in constant time, then the time complexity of MBD computation will reduce to $O(k)$.

Height h of any Isolation Tree cannot exceed $log_2(|S|)$. It also limits the number of nodes in any Isolation Tree to $2^{(h+1)}$. Please refer to Figure 2. It shows an example of such a tree with a height of three and the number of nodes fifteen. This tree is a perfect binary tree. We number the nodes of such a perfect binary tree by performing a breadth-first traversal. After this particular method of numbering the nodes, we call the perfect binary tree a maxTree. Any Isolation Tree that we construct will be a subtree of the maxTree of height $log_2(|S|)$. To number the nodes of any Isolation Tree in our Isolation Forest, we map that tree to the corresponding maxTree. This mapping results

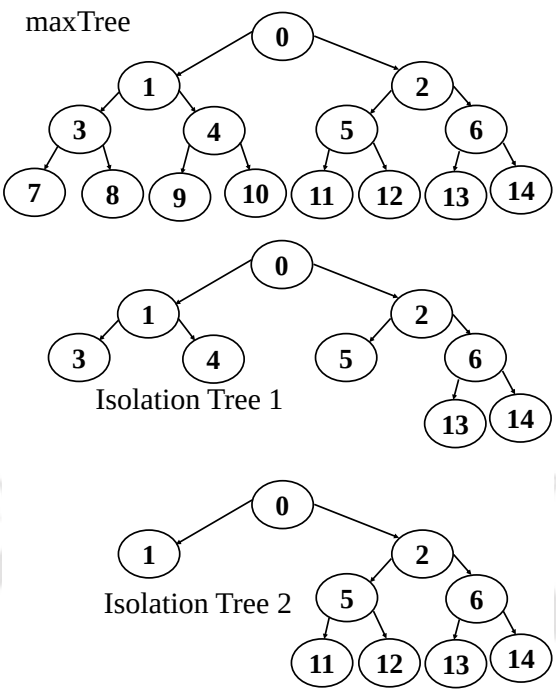


Figure 4.2: Example of maxTree with height 3 and consistent numbering of nodes across two Isolation Trees.

in the consistent numbering of nodes across all Isolation Trees. Please refer to Figure 2 for an example of such mapping and numbering. Therefore, LCA computation across all Isolation Trees also becomes identical. For example, node 2 is the LCA of nodes 5 and 14 across both Isolation Trees in Figure 2. We pre-compute LCA for all node pairs in the maxTree only once after the Isolation Forest construction and store it in a matrix. While performing LCA computation for MBD, we just read the corresponding value from the pre-computed LCA matrix in a constant time.

4.4.2 Potential Interesting Pairs

Only Interesting Pairs affect the outcome of mass-based clustering. A data point pair is an Interesting Pair if and only if:

- Its MBD is less than μ, where μ is a user-supplied parameter to the MBScan algorithm. (It is similar to parameter ϵ in DBScan) and
- At least one of the points in the data point pair is not labeled as OUTLIER.

In our experiments, we have observed that only about 3% of total pairs are Interesting Pairs, indicating that there is dramatic scope for improving the running time and memory footprint of MBScan. Consider a data point pair $P1$ consisting of data points x and y. If $P1$ is an Interesting Pair, its MBD should be less than μ in at least one of the Isolation Trees. Also, if at least one of the data points in $P1$ is not labeled as OUTLIER, then without loss of generality, let us assume that it is the data point x. The $\mu - neighborhood$ of x should have more than $minPts$ data points in at least one Isolation Tree. Now, let us put both these requirements together. For a data point pair to be an Interesting Pair, it must be found together in at least one node with mass m such that $minPts < m < \mu$. This condition is necessary but not sufficient.

We perform a depth-first traversal of each Isolation Tree to locate the largest nodes that satisfy this necessary condition on node mass. We call such nodes as marked nodes. Now, we can limit our MBD computation to only those pairs that are found in at least one marked node. We can further strengthen this filtering criterion by requiring that a data point pair should be found in at least q marked nodes. With a value of q greater than one, we might miss some Interesting Pairs. However, we have observed that we do not miss any Interesting Pairs even with the q value set to $0.4 * k$. Here, k is the number of Isolation Trees in the Isolation Forest. Data points that are similar will end up together in any randomly created Isolation Tree. That is why we do not miss any Interesting Pair even with a high value of q. Only with a value of q higher than $0.5 * k$, we start missing some of the Interesting Pairs. Now, our MBD computation is limited to only Potential Interesting Pairs found in at least q marked nodes.

4.4.3 Lower Bound on MBD

While computing MBD for a data point pair, we have to go across k Isolation Trees. However, we can terminate the MBD computation early if we can conclude that MBD for a given pair will exceed the threshold μ. Consider the partial MBD computed for a data point pair $P1$ using only the first r Isolation Trees, where $1 \leq r < k$. The cumulative score (CS) contributed by these first r Isolation Trees is

$$CS_r(P1) = \sum_{i=1}^{r} MBD_i(P1) \tag{4.3}$$

If the cumulative score for any pair exceeds $\mu * k$, then its final MBD will be greater than μ. Hence, we can terminate the MBD computation for a pair if its cumulative score exceeds $\mu * k$.

4.5 Experimental Evaluations

We evaluated our work empirically by performing extensive experiments over ten datasets (four synthetic and six real-world). Please refer to table 4.1 for information about the datasets[2]. These datasets were chosen because these are some of the most popular datasets used in the clustering research community[1][104][105] [106][107][108][109][110]. All the experiments are run on a server having Ubuntu Linux operating system version 18.04 with 128GB RAM. All the algorithms were implemented in C++ and compiled using the GNU C++ compiler.

Table 4.1: Datasets details

Dataset	Data Points	Dimensions	Clusters	Type
segment[1]	2.3×10^3	19	7	Real
D31[110]	3.1×10^3	2	31	Synthetic
S4(gaussian)[106]	5×10^3	2	15	Synthetic
unbalanced[104]	6.5×10^3	2	8	Synthetic
sattelite[4]	6.4×10^3	36	7	Real
pendigits[108]	1×10^4	16	10	Real
letter[109]	2×10^4	16	26	Real
shuttle[4]	5.7×10^4	9	7	Real
mnist[107]	7×10^4	784	10	Real
worms2d[105]	1×10^5	2	35	Synthetic

By incorporating our work into the MBScan algorithm, we get an optimized version of MBScan. We call it sMBScan (Scalable MBScan). We compare sMBScan against the original MBScan using three evaluation criteria: running time, memory footprint, and clustering quality. As Isolation Forest construction involves randomization, we

[2] https://github.com/nidhiahl/sMBSCAN

report numbers after running each algorithm ten times and then averaging across all ten runs. For Isolation Forest construction, the sample size ($|S|$) was set to 256 or 1% of the dataset size, whichever maximum. The MBScan paper suggested a sample size of 256. However, they performed experiments only with small datasets of up to 10,992 data points. For larger datasets with tens or hundreds of thousands of data points, we need a larger sample size to construct a reliable Isolation Forest. The Isolation Forest consisted of hundred Isolation Trees ($k = 100$). For our second optimization on searching Potential Interesting Pairs, the value of q was set to $0.5 * k$.

Table 4.2: Running Time and Speedup results

Dataset	Time & Speedup		
	MBScan (in seconds)	sMBScan (in seconds)	Speedup
segment	25.06	2.77	9.03
D31	49.85	2.77	17.95
S4(gaussian)	97.32	6.81	14.27
unbalanced	164.62	7.62	21.59
sattelite	151	11	13.38
pendigits	1,082	63	17.01
letter	1,890	112	16.78
shuttle	18,115	846	21.39
mnist	35,698	668	53.38
worms2d	56,882	4,906	11.59

For all datasets, we can observe that the number of Interesting Pairs (IP) is a tiny fraction of the Total Pairs (TP) (Table 4.5). Mass-based clustering needs MBD values only for these Interesting Pairs. Our sMBScan algorithm computes and stores MBD only for Potential Interesting Pairs (PIP). Therefore, we expect significant speed-up and memory savings for the sMBScan algorithm.

Please refer to table 4.2 for the details of the experimental results. We can observe that the running time of MBScan is impractical for larger datasets. For the largest dataset in our experiments (WORMS2d), the running time of MBScan is over 15 hours. However, our algorithm can perform the clustering for the same dataset in just 82 minutes. Across all datasets, sMBScan consistently maintains a significant speed-up over the MBScan.

Similarly, for memory footprint, the memory requirement of MBScan is impractical

Table 4.3: Memory footprint results

Dataset	Memory		
	MBScan (in MB)	sMBScan (in MB)	Memory Saving (in Percentage)
segment	58	27	53
D31	80	32	59
S4(gaussian)	154	47	69
unbalanced	224	51	77
sattelite	227	55	75
pendigits	550	78	85
letter	1,682	113	93
shuttle	7,356	361	95
mnist	16,626	842	94
worms2d	43,802	1,997	95

(Table 4.3). For the largest dataset, WORMS2d, MBScan requires more than 42 Gigabytes of RAM. For the same dataset, sMBScan needs less than 2 Gigabytes of RAM. Across all datasets, sMBScan provides significant memory savings. Especially for large datasets, the memory saving is even more than 90%.

Table 4.4: Clustering Quality results

Dataset	Quality		
	F-meaure	RandIndex	Entropy
segment	0.63	0.86	0.34
D31	0.72	0.91	0.31
S4(gaussian)	0.71	0.76	0.39
unbalanced	0.73	0.79	0.30
sattelite	0.83	0.99	0.05
pendigits	0.68	0.73	0.39
letter	0.60	0.78	0.35
shuttle	0.59	0.69	0.27
mnist	0.55	0.63	0.47
worms2d	0.70	0.73	0.39

Refer to table 4.4, to measure the quality of clustering, we have used three standard measures: F-measure, Rand Index, and entropy [111]. We know the ground truth label for each data point in all our datasets. The F-measure of the clustering is the average of the F-measure of each cluster. The entropy for the clustering is calculated as the weighted average of the entropy of each cluster. The weight of each cluster is

the number of data points in it. Rand Index for clustering is computed by calculating true positive and true negative data point pairs. The absolute values of these quality measures are not crucial to our work. However, the important point is that our sMBScan algorithm achieves the same value as the MBScan algorithm for all quality measures.We can conclude that the set of Potential Interesting Pairs computed by sMBScan is a superset of the set of Interesting Pairs for each dataset. None of our optimizations filter out any Interesting Pair. As a result, we can scale up the mass-based clustering without any loss in clustering quality.

Refer to table 4.5, for a discussion on the effectiveness of $sMBScan$ and how much scope for further improvement to obtain an ideal solution. The table highlights the effectiveness of scalable MBSCAN ($sMBSCAN$) in significantly reducing the number of unnecessary distance computations across various datasets. The TP/PIP ratio in the second-last column quantifies the reduction achieved by $sMBSCAN$. This indicates that $sMBSCAN$ effectively filters out the majority of irrelevant pairs. The PIP/IP ratio in the last column shows the remaining scope for further reduction, which is minimal for most datasets, confirming that $sMBSCAN$ already performs near-optimal reduction. Datasets like "letter" show no additional scope for reduction, showing that 100% unnecessary pairs have already been eliminated by $sMBSCAN$. Overall, the results validate $sMBSCAN$ ability to minimize the unnecessary distance computations and compute only the essential pairs needed for clustering, leaving a low margin for further improvement.

4.6 Conclusion and Future Work

The quadratic time and space complexity of the MBScan algorithm is the major bottleneck in scaling up the mass-based clustering approach to larger datasets. In this chapter, we have presented three optimizations to the MBScan algorithm that attack this bottleneck. We have achieved significant speed-up with dramatic saving in memory without any loss in clustering quality. With our work, mass-based clustering will now become practical for larger datasets.

Table 4.5: Scope of further reduction in number of pairwise distance computations in scalable MBScan. TP/PIP depicts the reduction provided by sMBScan, PIP/IP depicts the further scope of reduction.

Dataset	Scope for Improvement				
	Total Pairs (TP)	Potential Interesting Pairs (PIP)	Interesting Pairs (IP)	TP/ PIP	PIP/IP
segment	2.6×10^6	2.3×10^5	8.5×10^4	11	2
D31	4.8×10^6	2.2×10^5	7.3×10^4	21	3
S4(gaussian)	1.2×10^7	7.8×10^5	2.7×10^5	15	2
unbalanced	2.0×10^7	5.5×10^5	4.6×10^4	36	11
sattelite	2.1×10^7	1.5×10^6	6.6×10^5	14	2
pendigits	6.0×10^7	2.8×10^6	5.6×10^5	21	5
letter	1.9×10^8	1.9×10^6	3.3×10^5	100	-
shuttle	1.6×10^9	3.5×10^7	9.5×10^6	45	3
mnist	2.4×10^9	3.3×10^6	1.4×10^4	727	235
worms2d	5.5×10^9	5.0×10^8	1.7×10^8	11	2

There is only a limited scope to improve the performance of our algorithm sMBScan. Any mass-based clustering algorithm needs to compute MBD at least for all the Interesting Pairs. The key ratio to look at is PIP/IP in table 4.5. This ratio is the upper bound on the further speed-up that can be achieved over our algorithm sMBScan. Our work can be further improved by designing algorithms to scale out the mass-based clustering to utilize multiple processors and systems.

5

Incremental Isolation Forest

Chapter Highlights

- We observe that concept drift in new incoming data degrades the performance of Isolation Forest.
- We target three types of concept drifts.
- Our approach Incremental Isolation Forest ($I^2Forest$) quickly updates the model and adapts to the concept drift caused by new data.
- We examine $I^2Forest$ for efficiency and quality of anomaly detection in complete data after changes.
- This chapter is based on the publication "Incremental Isolation Forest for Handling Concept Drift in Anomaly Detection" presented in CoDS-COMAD 2024, and the extended version is submitted to a peer-reviewed journal.

5.1 Abstract

Isolation Forest is a well-known model designed for anomaly detection task. It identifies regions corresponding to anomalies in the training data and defines anomalies as "few

and different". With the arrival of new data after training the model, concept drift can occur in three ways. First, anomalies can occur in the new regions of the feature space. Second, existing anomalies can become normal with the addition of new data. Third, a new normal region is introduced after adding new data. We observe that the performance of Isolation Forest severely degrades in all these scenarios. Current works fail to tune the existing Isolation Forest to adapt to all three types of concept drifts. We propose an algorithm, Incremental Isolation Forest, to quickly update the existing Isolation Forest in response to the arrival of new data. We perform extensive experiments using synthetic and real-world datasets. Experimental results show that our approach achieves significant time savings with minimal or no loss in anomaly detection performance. Our method is more robust to catastrophic forgetting than incremental baselines that forget the old data.

5.2 Introduction

The question central to this chapter is: How to update a Machine Learning model in response to new data? This question becomes even more critical when the new data causes concept drift. A lazy solution is to keep using the old model. However, it will result in inferior performance. An aggressive solution is to rebuild the model from scratch. This solution will improve the model performance at the cost of time spent in retraining the model. Instead of choosing any of these two extremes, we can design an incremental solution that quickly updates the existing model to match the performance of the aggressive solution. This chapter focuses on the anomaly detection task and the Isolation Forest model to design an incremental solution. Our primary research contribution is to develop an efficient update scheme to incrementally modify the Isolation Forest model in response to the addition of new data.

Anomaly detection is the task of identifying the outliers that do not conform to the normal patterns in the data [112]. It is an important task with applications in diverse domains such as network security, medical diagnosis, and many others [113]. The definition of anomaly heavily depends on the specific application and model used for

detection. Isolation Forest (iForest) is a popular model for anomaly detection [4] that describes anomalies as "few and different" from the rest of the data. iForest is a collection of independently constructed Isolation Trees (iTrees). Each iTree identifies regions corresponding to anomalies in the existing training data. Rest all parts of the feature space are considered normal. Each iTree is constructed using a small random sample from the training data. iForest performs prediction by averaging across all constituent iTrees.

Concept drift occurs when the relation between the input data and the target variable changes [3]. This change results in the degradation of the machine learning model's performance. The iForest defines anomalies as "few and different" and models them by defining regions corresponding to those anomalies in the training data. However, with the arrival of new data, the regions modeled by iForest can change in three ways as follows (Please refer to Figure 5.1.):

1. **CDI (New anomalies)**: anomalies occur in new regions of the feature space. Existing iForest will fail to detect these anomalies, resulting in false negatives.

2. **CDII (Anomalies change to Normal)**: existing anomalies can become normal. Existing iForest will still identify data points in such regions as anomalies, resulting in false positives.

3. **CDIII (New Normal)**: normal points occur in new regions of the feature space. Existing iForest will identify data points in such regions as anomalies, resulting in false positives.

To overcome these problems, we propose the Incremental Isolation Forest ($I^2Forest$) model. It quickly updates the existing iForest in response to the arrival of new data. The intuition of our approach is to update the definition of anomalous regions in each constituent iTree using a small sample from the new data. We perform the breadth-first traversal of each iTree to carry out updates efficiently. Our updates are focused on answering three questions. First, have anomalies occurred in any new region of the feature space? Second, has any of the existing anomalous regions become normal?

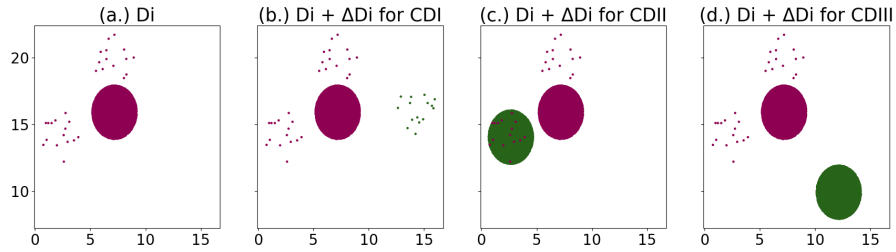


Figure 5.1: New datapoints(ΔD_i) introduce Concept Drift (CD-I: Anomalies occur in a new region; CD-II: Existing anomalies become normal data points; CD-III: New normal regions introduced). Please note drift causing data points are shown in green color.

Third, do normal points exist in any new region of the feature space? For complete reproducibility, all our code and datasets are available publicly on the Web.[1]

To summarize the contributions, $I^2Forest$ is an incremental approach that achieves a speedup of 1.4X to 7842X over incremental methods and 1.7X to 39X over the aggressive approach. This time saving is achieved with no or minimal loss in anomaly detection quality.

5.3 Related Work

Anomaly detection models span multiple paradigms such as supervised, semi-supervised, and unsupervised models [112]. These models work on the intuition that anomalies are sparse and fewer in the count. Compared to the state-of-the-art anomaly detection models, the performance of $iForest$ is slightly inferior but still comparable [114]. However, $iForest$ is one of the fastest models for run-time efficiency.

5.3.1 Isolation Forest

Isolation Forest details are described in Chapter 2. Our work focuses on updating the structure of each $iTree$ in response to the change in data. The data changes from D_i at timestamp T_i to D_{i+1} at timestamp T_{i+1} with addition of ΔD_i to D_i.

[1] https://github.com/nidhiahl/I2F

5.3.2 Baselines

We compare our work with two non-incremental approaches: Lazy and Aggressive. In the Lazy approach, we do not update the model when new data arrives. In the Aggressive approach, we simply rebuild the $iForest$ from scratch using old as well as new data. The Lazy approach will be the most efficient as it does not update the model. However, the Lazy approach will have the worst quality of anomaly detection when the new data causes the concept drift. The aggressive approach will be inefficient. However, it is expected to have high-quality of anomaly detection as it uses all the available data to build new $iForest$.

Lazy and aggressive approaches are two extremes, with the former optimized for efficiency and the latter for quality. Incremental methods try to achieve the best of both worlds by balancing efficiency and quality. A variety of incremental methods exist that handle the concept drift introduced by adding new data for $iForest$ and related methods. We categorize the existing incremental methods based on how they update the model. Methods in the first category replace the whole forest and build every tree of the forest from scratch using the recent batch of data. iF_{ASD} is an example of a method in this category[16]. Methods in the second category identify and replace a few bad-performing trees from the forest. $PCBIF$ is an example method in this category[17]. Methods in the third category modify the existing trees based on the changes in the data. These methods avoid the costly operation of completely replacing any tree. $RRCF$[18] and $SENCForest$[19] are the example methods in this category. Please refer to Table 5.1 for a summary of these methods.

The first incremental approach, iF_{ASD} (isolation Forest for Anomaly detection in Streaming Data), focuses only on new data[16],[115]. This method assumes that the anomaly rate in the old and new data should be almost identical. The anomaly rate for a dataset is the fraction of data labeled as an anomaly by the model. If the anomaly rate for the new data is significantly higher than the old data, then iF_{ASD} builds a new iForest from scratch using the new data. This method uses the increase in the anomaly rate as a signal for concept drift. However, it is not enough to detect all three concepts drifts

Table 5.1: Comparison of existing approaches.

Method	Training Data Type	Update Type	Data required for model update	Risk of catastrophic forgetting
Lazy	Unlabelled	No model update	None	Low
Aggressive	Unlabelled	Build forest from scratch for complete data	D_i+ΔD_i	None
iF_{ASD}	Unlabelled	Replace Forest	ΔD_i	High
$PCBIF$	Unlabelled	Replace few trees	ΔD_i	Moderate
$SENCForest$	Labelled	Modify all trees	ΔD_i	Low
$RRCF$	Unlabelled	Modify all trees	ΔD_i	Moderate
$I^2Forest$ (ours)	Unlabelled	Modify all trees	ΔD_i	Low

we want to address in our work. Also, iF_{ASD} loses knowledge about the old data when it detects the drift. Therefore, iF_{ASD} is prone to catastrophic forgetting. Recently, Togbe et al. tried to improve the iF_{ASD} algorithm by adding new strategies for concept drift detection[116]. However, catastrophic forgetting remains the bottleneck for their approach.

A more recent work, $PCBIF$ (Performance Counter Based Isolation Forest), identifies bad-performing isolation trees in the model[17]. It replaces only such trees in the existing isolation forest. Their experimental results show that $PCBIF$ performs similarly to iF_{ASD}. With only selective replacement of isolation trees, $PCBIF$ is more robust to catastrophic forgetting than iF_{ASD}.

$RRCF$ (Robust Random Cut Forest) [18] is an improved variant of isolation forest. It adapts to the changing data by modifying every tree in the forest. However, it has two main limitations. First, it updates the forest with an update size of one data point. We have to update the forest for every single new data point, leading to a time-consuming forest update process. Second, it targets only one type of concept drift: anomalies occurring in new regions. All the incremental methods we have reviewed till now work with unlabelled data. In contrast, $SENCForest$ (Streaming Emerging New Classes Forest)[19] needs labeled data. It is an improvisation of iForest that uses labels to store information about existing classes in data. It also stores information about the feature

space regions with anomalies in training data. It can detect incoming new anomalies (CDI) and new normal (CDIII) but fails to adapt to the changes where old anomalies become normal (CDII).

Considering various limitations of existing incremental methods, we had the following goals while designing our approach $I^2Forest$.

- Handle all three types of concept drifts (CDI, CDII, and CDIII).
- Maintain a balance between performance on old and new data. In other words, perform well on new data while avoiding catastrophic forgetting over the old data.
- Efficient updates to the iForest in response to new data. In other words, update iForest only once for the whole new data ΔD_i

5.4 Incremental Isolation Forest

When the data changes from D_i at timestamp T_i to D_{i+1} at timestamp T_{i+1} with the addition of ΔD_i to D_i, we have three options. First, a lazy option of using the old iForest F_i built using D_i at timestamp T_i. Second, an aggressive option of building iForest F_{i+1} using D_{i+1}. The first option will have reduced performance for anomaly detection if ΔD_i causes concept drift. The second option will be able to handle the concept drift but with the time delay of building F_{i+1} from scratch. Instead, we provide a third option of quickly updating the F_i to F'_{i+1} using only ΔD_i. We should efficiently change the structure of each iTree in response to the concept drift caused by ΔD_i . If anomalies in ΔD_i occur in new regions of the feature space, then we should insert corresponding new shallow leaf nodes in the iTree. If adding new points in ΔD_i converts existing anomalies to normal, we should replace corresponding shallow leaf nodes with deep subtrees. If a new group of normal points is added in ΔD_i, then we add newly constructed deeper subtrees.

Our method $I^2Forest$ is divided across five algorithms explained in this section. Algorithm 6 represents the overall flow of $I^2Forest$ where we update each isolation tree using Algorithm 7. In Algorithm 7, we modify the structure of a specific isolation tree while

Algorithm 6 $I^2Forest(\Delta D, F_i)$

```
for each iTree T_i^m in F_i do
    Select sample ΔS^m from ΔD_i
    updateITree(ΔS^m, T_i^m)
end for
```

performing the breadth-first traversal of that tree. While performing the breadth-first traversal, we can encounter two types of nodes: leaf and internal. Algorithm 8 explains how we process a leaf node during the breadth-first traversal. Algorithm 9 details how we process an internal node during the breadth-first traversal. While processing an internal node, we might also need to update its left and right subtrees. Algorithm 10 explains how to update the left subtree of an internal node. The process to update the right subtree is described in Algorithm 11.

Please refer to the Algorithm 6. Our $I^2Forest$ updates the F_i to F'_{i+1} using ΔD_i. To build F_{i+1}, the aggressive method would have selected the sample S_{i+1} of size $(|S_i|/|D_i|) * |D_{i+1}|$ for each iTree. To maintain this sampling ratio, we select a random sample ΔS from ΔD_i. We ensure that for each iTree, $|S_{i+1}| = |S_i| + |\Delta S|$. Now, we insert ΔS in the iTree to update its structure. Data points in ΔS help us to estimate if hyperrectangles corresponding to any node in the iTree need to be changed. Please note that ΔS is selected independently for each iTree. With this update to the structure of each iTree, we expect that the incremental iForest F'_{i+1} will perform similarly to the iForest F_{i+1}.

Algorithm 7 $updateITree(\Delta S, T)$

```
T.root.newPoints = ΔS
//Add all points from ΔS to root node of iTree T
Create queue BFT
//Queue used for breadth first traversal of T
BFT.enqueue(T.root)
while BFT.notEmpty() do
    N = BFT.dequeue
    if N is leaf node then
        updateLeaf(N, BFT)
    else
        updateInternalNode(N, BFT)
    end if
end while
```

Please refer to Algorithm 7. While updating the structure of an iTree, we first store new sample points in the root node. Then, we perform the breadth-first traversal (BFT) of the tree. At each node, we have two tasks to perform. First, decide if the split criteria needs to be changed. Second, distribute the new sample points between the two child nodes. While performing these two tasks, we differentiate the nodes into two categories: leaf and internal.

Algorithm 8 $updateLeaf(N, BFT)$

```
1: if N.height == T.maxHeight then
2:     //Height of isolation tree cannot exceed ⌈log2(|Si+1|)⌉
3:     return
4: else
5:     splitNode(N) //Node splitting same as in iForest. It will attach two child nodes
       to N.
6:     BFT.enqueue(N.leftChild)
7:     BFT.enqueue(N.rightChild)
8: end if
```

Please refer to Algorithm 8. The leaf nodes already at the maximum height require no further processing (Lines 1 to 3). There are no split criteria to change, and there are no children. For the leaf nodes with a height less than the maximum, we need to split them if they receive new sample points (Lines 4 to 8). This update will help us to convert existing anomalous regions to normal by introducing deeper subtrees below the existing leaf nodes. To continue the BFT, we need to add the newly created left and right children to the BFT queue.

Figure 5.2 summarizes four possible cases while updating an internal node. Please refer to Algorithm 9. At each internal node, the split criteria depend on minimum and maximum values for the split attribute. With the arrival of new sample points, there is no need to change the split criteria if there is no change in these values. We just have to distribute the received sample points between the left and right child nodes. Suppose we observe that at node N, the minimum value for the split attribute is reduced due to the arrival of new sample points. In that case, we must update its left subtree (Algorithm 10). We create a new node $N_{L'}$. This node becomes the left child of N. It accommodates all the data points that would belong to the node N_L and additional

Algorithm 9 *updateInternalNode*(N, BFT)

```
1: if N.height == T.maxHeight then
2:    Prune left and right subtrees at N.
3:    //N becomes leaf node.
4:    return
5: else
6:    Distribute N.newPoints between N.left.newPoints and
      N.right.newPoints using the split criteria chosen at N.
7:    if N.SA.min ≤ N.newPoints.SA.min then
8:       BFT.enqueue(N.leftChild)
9:    else
10:      updateLeftSubTree(N, BFT)
11:   end if
12:   if N.SA.max ≥ N.newPoints.SA.max then
13:      BFT.enqueue(N.rightChild)
14:   else
15:      updateRightSubTree(N, BFT)
16:   end if
17: end if
```

sample points having split attribute value less than the split criteria. The node $N_{L'}$ is split further into the original N_L and a new node $N_{L''}$. Here, the split attribute is the same as the split attribute chosen for N. However, the split value is the minimum value of the split attribute observed in node N before the arrival of the new sample points. Similarly, if the maximum value of the split attribute at node N changes with the arrival of new sample points, it is handled by updating the right subtree (Algorithm 11). The algorithm is similar to that of updating the left subtree. Updating subtrees of internal nodes will help us to identify new anomalous regions by introducing new shallow leaf nodes.

The worst-case time complexity of constructing the iForest is $O(S)$, where S is the set of data points used to construct each iTree[4]. Each iTree is a binary tree. When constructing any iTree, all the selected sample points initially belong to the root node. In the worst case, each sample might belong to a different leaf node. Then, we will have to construct $|S|$ leaf nodes in the tree. For our method $I^2Forest$, the worst case time complexity is $O(\Delta S)$, where ΔS is the sample selected from new data ΔD_i. While updating the structure of each iTree, we initially put all data points of ΔS into the root node. In the worst case, these new points can introduce ΔS new leaf nodes.

Algorithm 10 $updateLeftSubTree(N)$

```
1: N_L = N.left
2: if N_L.height == T.maxHeight then
3:    return
4: end if
5: if N_L is leaf node then
6:    splitNode(N_L)
7:    BFT.enqueue(N_L.left)
8:    BFT.enqueue(N_L.right)
9: else
10:   N_L' = N_L // Creating a copy of N_L
11:   N.left = N_L'
12:   N_L'.left = N_L'' // Creating new node N_L''
13:   N_L'.right = N_L
14:   N_L'.SA = N.SA // SA is splitting attribute of the node N_L'
15:   N_L'.SV = N.SA.min // SV is splitting value of the node N_L'
16:   N_L'.SA.min = N_L'.newPoints.SA.min
17:   BFT.enqueue(N_L'')
18:   BFT.enqueue(N_L)
19: end if
```

5.5 Experiments

We evaluate our approach $I^2Forest$ through extensive experimental evaluation. First, we will describe the datasets, followed by the experimental setup and discussion about the results.

5.5.1 Datasets

We have performed experiments on a synthetic 2-dimensional dataset and five real-world datasets. Four of the five real-world datasets (*gassensor*, *shuttle*, *covtype*, and *poker*) are chosen based on a detailed study about benchmarking datasets for concept drift handling algorithms [117]. The fifth real-world dataset (*crop*) is from the UCI Machine Learning repository.

We also experimented with a *synthetic* dataset that we generated. It is a two-dimensional dataset with 30030 data points. The dataset is plotted in Figure 5.1. The dataset consists of five classes corresponding to five regions in the feature space. Three of the five classes have 10000 data points each, and the other two have 15 data points each. The *gassensor* dataset consists of 128 features from 16 chemical sensor observations

Algorithm 11 *updateRightSubTree*(N)

```
1: N_R = N.right
2: if N_R.height == T.maxHeight then
3:    return
4: end if
5: if N_R is leaf node then
6:    splitNode(N_R)
7:    BFT.enqueue(N_R.left)
8:    BFT.enqueue(N_R.right)
9: else
10:   N_R' = N_R // Creating a copy of N_R
11:   N.right = N_R'
12:   N_R'.right = N_R'' // Creating new node N_R''
13:   N_R'.left = N_R
14:   N_R'.SA = N.SA // SA is splitting attribute of the node
15:   N_R'.SV = N.SA.max // SV is splitting value the node N_R'
16:   N_R'.SA.max = N_L'.newPoints.SA.max
17:   BFT.enqueue(N_R)
18:   BFT.enqueue(N_R'')
19: end if
```

collected over a period of three years in the gas delivery platform at the University of California. There are 13910 instances, each belonging to one of the six gas types. The *shuttle* dataset is a statlog dataset where each instance has nine numeric features collected from the sensors' measurements and belongs to one of the seven different categories. The *fcovtype* dataset is a forest cover dataset from United State's Forest Service Information system. The dataset has a total of 581012 records. These records are categorized into seven different categories. Each record has 54 attributes. These attributes describe the various factors that impact vegetation on land cover. The *poker* dataset is a poker-hand prediction dataset consisting of 1025010 belonging to 9 different classes. It has ten integer attributes representing the rank and suit of each card in a hand of 5 cards. The *crop* dataset is a temporal and optical-radar-based image dataset for cropland classification. A total of 174 features (2*49 radar and 2*38 optical features) facilitate 325834 records to be categorized in one of the seven crop categories. Please refer to Table 5.2 for a brief summary of datasets.

The datasets chosen have at least five classes. Out of these five classes, at least three have a significant number of data points. We must induce all three types of concept

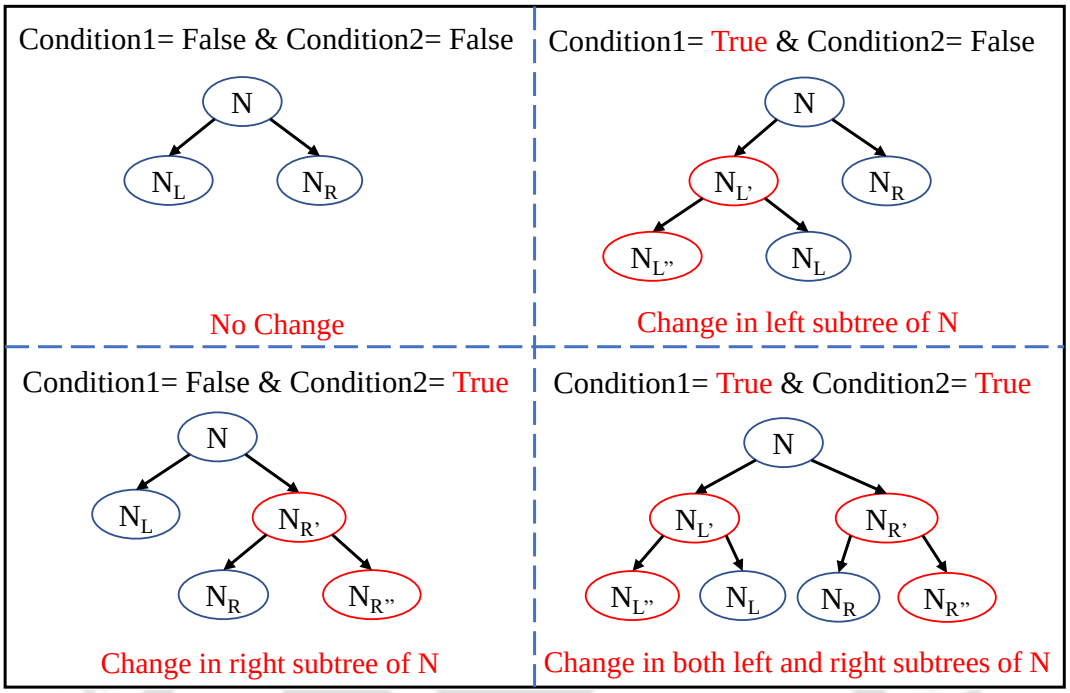


Figure 5.2: Updating an Internal Node. (Condition1=Line 7, Algorithm 4 Condition2=Line12, Algorithm 4)

drifts we are targeting. The CDI concept drift corresponds to the addition of anomalous points in the new regions of the feature space. The CDII concept drift corresponds to adding new points that convert old anomalies into normal points. The CDIII concept drift corresponds to adding new normal points in new regions of feature space.

For experiments, we carefully construct two partitions D_i as old data and ΔD_i as new data. A separate ΔD_i is constructed for each concept drift type. Consider the five classes in the dataset as A, B, C, D, and E. These classes are arranged in the descending order of number of data points for each class in the dataset. D_i consists of all the points of the largest class A and a small fraction of class B and D. For CDI drift, ΔD_i consists of a small number of data points from class E as new anomalies. Similarly, for CDII drift, ΔD_i consists of all the points of class B, the second largest class in the dataset. D_i had some anomalies corresponding to class B. After introducing CDII concept drift, they will be converted to normal data points. To introduce CDIII drift, ΔD_i consists of all points from the third largest class C. Points from class C will introduce a new normal. Please refer to table 5.2 for details about the sizes of D_i and ΔD_i.

Table 5.2: Dataset details

	synthetic	gas sensor	shuttle	covtype	poker	crop
Attributes	2	128	9	54	10	174
$\|D_i\|$	10030	3045	45620	283301	415626	85144
%anomaly in D_i	0.3	1.18	0.074	0.066	0.069	0.082
$\|\Delta D_i\|$ for CD-I (All Anomaly)	15	30	50	200	100	35
$\|\Delta D_i\|$ for CD-II (All Normal)	9085	2926	8903	211840	350526	75673
$\|\Delta D_i\|$ for CD-III (All Normal)	10000	2565	3267	35745	39432	74067

5.5.2 Experimental Setup

Our experiments were carried out on a machine with Ubuntu 20.04.5 LTS operating system and 16 GB RAM. Each forest structure that we built consisted of 100 iTrees. For the construction of each iTree, the sample size was set to 256 or 1% of the input data points, whichever is more. All results are averaged across five runs due to the inherent randomization in constructing iForest. The anomaly rate for iF_{ASD} is set to the original fraction of anomalies in old data D_i. For $SENCForest$, the parameter s is set to 30, as mentioned in the paper by the authors.

For a fair comparison, we train the respective underlying forest of each method over full datasets D_i and construct forest F_i for every baseline, then with the arrival of ΔD, each of the F_i is updated to $F_{i+1}^{'}$. $RRCF$ processes ΔD_i point by point and updates the model for each data point. $PCBIF$ and iF_{ASD} process each point to detect drift and select a sample form ΔD_i to build trees from scratch. To create the same sized trees as existing ones, the sample chosen from ΔD_i will be the same size as in the existing forest F_i. After adapting to the changes in ΔD_i, the respective existing F_i updates to its own $F_{i+1}^{'}$. Eventually, all the comparing approaches use their respective $F_{i+1}^{'}$ to find anomalies in $D_i + \Delta D_i$.

Table 5.3: Running Time and AUC.

Datsets	Methods	CDI			CDII			CDIII		
		AUC	Running Time	Speedup by $I^2Forest$	AUC	Running Time	Speedup by $I^2Forest$	AUC	Running Time	Speedup by $I^2Forest$
synthetic	Lazy	0.578			0.328			0.327		
	Aggressive	0.584	0.124	10.7	0.447	0.19	1.7	0.62	0.19	1.8
	iF_{ASD}	0.612	0.0022	0.2	0.619	0.252	2.3	0.502	0.143	1.4
	$PCBIF$	0.593	0.036	3.1	0.918	10.591	97.2	0.685	9.197	89.3
	$SENCForest$	0.607	0.041	3.5	0.513	0.28	2.6	0.63	0.279	2.7
	$RRCF$	0.685	0.419	36.1	0.932	813.661	7464.8	0.693	807.807	7842.8
	$I^2Forest$	0.621	0.0116	1	0.916	0.109	1	0.683	0.103	1
gas sensor	Lazy	0.749			0.379			0.62		
	Aggressive	0.795	0.845	9.9	0.422	0.866	1.9	0.600	0.866	2.3
	iF_{ASD}	0.740	0.0003	0.0004	0.324	0.033	0.1	0.562	0.030	0.1
	$PCBIF$	0.722	0.55	6.4	0.447	3.988	8.6	0.473	3.455	9.2
	$SENCForest$	0.756	0.34	4	0.685	0.805	1.7	0.729	0.738	2
	$RRCF$	0.620	0.836	9.8	0.251	134.819	289.3	0.452	118.425	315
	$I^2Forest$	0.789	0.085	1	0.484	0.466	1	0.67	0.376	1
shuttle	Lazy	0.951			0.940			0.921		
	Aggressive	0.957	0.410	12.4	0.973	0.517	4.7	0.931	0.441	4.6
	iF_{ASD}	0.160	0.012	0.4	0.821	0.219	2	0.787	0.099	1
	$PCBIF$	0.916	0.038	1.2	0.795	9.051	81.5	0.825	2.408	25.1
	$SENCForest$	0.859	0.011	0.3	0.968	0.385	3.5	0.966	0.150	1.6
	$RRCF$	0.98	0.772	23.4	0.75	685.027	6171.4	0.775	318.739	3320.2
	$I^2Forest$	0.988	0.033	1	0.971	0.111	1	0.973	0.096	1
covtype	Lazy	0.784			0.524			0.539		
	Aggressive	0.802	8.189	39	0.591	16.026	6	0.571	9.241	17
	iF_{ASD}	0.251	0.37	1.8	0.688	5.529	2.1	0.502	1.323	2.4
	$PCBIF$	0.500	2.923	13.9	0.494	283.028	106.4	0.502	48.087	88.4
	$SENCForest$	0.739	0.983	4.7	0.629	37.250	14	0.581	4.857	8.9
	$RRCF$	0.797	3.23	15.2	NA	NA	NA	0.53	25021.5	45995
	$I^2Forest$	0.808	0.212	1	0.589	2.66	1	0.574	0.544	1
poker	Lazy	0.775			0.426			0.560		
	Aggressive	0.778	4.870	26.6	0.443	9.141	2.2	0.564	5.343	8
	iF_{ASD}	0.35	0.315	1.7	0.597	20.417	5	0.542	3.168	4.8
	$PCBIF$	0.614	1.23	6.7	0.498	479.633	117.8	0.513	51.711	77.9
	$SENCForest$	0.803	0.40	2.2	0.545	30.706	7.5	0.599	3.139	4.7
	$RRCF$	0.80	6.427	35.1	NA	NA	NA	0.541	27602.4	41569.8
	$I^2Forest$	0.799	0.183	1	0.542	4.071	1	0.569	0.664	1
crop	Lazy	0.994			0.997			0.961		
	Aggressive	0.994	4.586	19.9	0.993	10.750	5.9	0.983	10.736	6.3
	iF_{ASD}	0.237	0.102	0.4	0.913	1.562	0.9	0.958	1.506	0.9
	$PCBIF$	0.465	0.372	1.6	0.405	106.567	58.8	0.419	105.199	61.4
	$SENCForest$	0.995	0.325	1.4	0.997	22.421	12.4	0.989	22.936	13.4
	$RRCF$	0.99	0.981	4.1	0.511	5821	3210.7	0.653	2993.3	1720.2
	$I^2Forest$	0.996	0.236	1	0.998	1.813	1	0.974	1.714	1

5.5.3 Results

We compare $I^2Forest$ against two extreme approaches (Lazy and Aggressive) and four incremental approaches (iF_{ASD}, $PCBIF$, $RRCF$, and $SENCForest$) described in Section 5.3. We want to compare these methods for all three types of concept drifts (CDI, CDII, and CDIII). We are specifically interested in evaluating the success of $I^2Forest$ for the three design goals mentioned in Section 5.3. To compare the efficiency, we measure the time required to update the model using each competing method. To compare the quality, we measure the AUC (Area Under the ROC Curve) score for the anomaly detection task using each competing method. Please refer to Table 5.3 for all experimental results.

Comparison with the Lazy approach

The running time column for the Lazy approach is blank because it does not update the model at all. Therefore, we cannot compute the speed up of $I^2Forest$ over the Lazy approach. The AUC score for $I^2Forest$ is better than the Lazy approach for all three concept drifts across all six datasets. This observation confirms the intuition that the old model needs to be updated when new data introduces concept drift.

Comparison to the Aggressive approach

The Aggressive approach completely rebuilds the whole Isolation Forest using the old and new data. Compared to the $I^2Forest$, we expect that its running time will be higher. $I^2Forest$ is more efficient than the Aggressive approach for all three concept drifts across all six datasets. The relative size of ΔD_i with respect to D_i plays a key role in deciding the speed that $I^2Forest$ can achieve. A larger size of ΔD_i means that the $I^2Forest$ needs to do more updates to the structure of each iTree. Concept drifts of type CDII and CDIII represent this scenario. In such a case, the speed-up of $I^2Forest$ over the Aggressive approach is lower. In our experiments, it is in the range of 1.6X to 6X for CDII, and the CDIII speed-up range is from 1.8X to 17X. For the CDI update, the relative size of ΔD_i is relatively small as it only introduces new anomalies. In such

a case, $I^2Forest$ needs to make minimal changes to the structure of each iTree. As a result, the speed-up of $I^2Forest$ is in the range of 10.7X to 39X for CDI update.

The aggressive approach has access to the old and new data while rebuilding the tree. Intuitively, we expect it to beat the $I^2Forest$ for the anomaly detection quality easily. However, the results are contrary. For CDI updates, $I^2Forest$ has a better AUC score for all datasets except the *gas sensor* dataset. Moreover, even for the *gas sensor* dataset, the AUC score of $I^2Forest$ is only marginally lower than the Aggressive approach. For CDII and CDIII updates, $I^2Forest$ can achieve a better AUC score for some datasets. This surprising success of $I^2Forest$ for quality over the Aggressive approach can be explained by the selective updates that $I^2Forest$ performs. With old data D_i, we already have a good model F_i. $I^2Forest$ makes only minimal changes to it to accommodate new data ΔD_i. As a result, it is able to maintain most of the performance on old data, and it performs well over the new data. In contrast, the Aggressive approach has to establish a good model for all the data in a single attempt.

Comparison with iF_{ASD},

When new data ΔD_i arrives, iF_{ASD} behaves like the Aggressive approach if it detects the concept drift. Therefore, we can observe that the running time of iF_{ASD} is sometimes higher than $I^2Forest$. However, when iF_{ASD} fails to detect the concept drift, it behaves like the Lazy approach. In such a scenario, the running time of iF_{ASD} is lower than $I^2Forest$.

Whenever iF_{ASD} runs faster than $I^2Forest$, its AUC score is always lower than $I^2Forest$. This result is expected as iF_{ASD} is just replicating the Lazy approach. In contrast, when iF_{ASD} runs slower than $I^2Forest$, it sometimes achieves a higher AUC score than $I^2Forest$. This observation tallies with the behavior of the Aggressive approach.

Comparison with $PCBIF$

$PCBIF$ is a three-step method. First, it detects the concept drift. Second, it identifies bad-performing trees. Third, it replaces such trees. The computational overheads of this three-step approach are high. As a result, $I^2Forest$ is always faster than $PCBIF$. Updates in $I^2Forest$ are more focused on handling the concept drift as compared to the coarse grain updates of $PCBIF$. Except for the synthetic dataset, the AUC score of

$I^2Forest$ is always higher than $PCBIF$. Even for the synthetic dataset, the $I^2Forest$ AUC score is only marginally lower than $PCBIF$.

Comparison with $SENCForest$,

When the new anomalies are added during the CDI updates, $SENCForest$ does not change the structure of the tree. It just updates its anomaly score mechanism in response to new data in the CDI update. Hence, for CDI updates, $SENCForest$ sometimes runs faster than $I^2Forest$. The tree structure update process in $SENCForest$ is far more complex than the $I^2Forest$. During the CDII and CDIII updates, the update size is large. This large update size triggers the tree structure update mechanism of $SENCForest$. As a result, the running time of $SENCForest$ is always higher than $I^2Forest$ for CDII and CDIII updates.

For the AUC score, $I^2Forest$ is able to beat $SENCForest$ during CDI updates for five out of the six datasets. For the *poker* dataset, their AUC scores are comparable during the CDI update. However, during CDII and CDIII updates, there is no clear winner between $SENCForest$ and $I^2Forest$ for the AUC score. Both methods update the tree structure in their own way. Nevertheless, the primary limitation of the $SENCForest$ is that it needs labeled data for training. All other methods can work with unlabelled data for training.

Comparison with $RRCF$

$RRCF$ updates the tree structure in response to every single data point in the new data ΔD_i. Therefore, $RRCF$ is the most time-consuming baseline among all the compared methods. For the two largest datasets in our experiments (*covtype* and *poker*), $RRCF$ did not finish the update even after twenty-four hours for the CDII update. Therefore, we have not mentioned the results for $RRCF$ for these two datasets for the CDII update.

Tree structure updates for every single new data point help $RRCF$ to perform well on CDI updates. Its performance is comparable to $I^2Forest$. However, for CDII and CDIII updates, $RRCF$ performance degrades as it is not designed to handle CDII and CDIII updates.

5.6 Conclusion and Future Work

In this chapter, we have presented an incremental algorithm $I^2Forest$. It quickly updates the existing iForest model in response to the arrival of the new data. $I^2Forest$ handles three types of concept drifts. We have compared our algorithm with six baselines. The experimental results show the trade-offs that exist between various incremental methods. This work can be further improved in two ways. First, we can identify which updates to the iTree structure matter the most to improve the anomaly detection performance. The second possible direction is to check whether the new data introduces any concept drift. If there is no concept drift, then there should not be any change to the iTree structure.

6

Conclusion and Future Work

In this thesis, we proposed Isolation Forest-based approaches to prune the redundant computations. We focused on three tasks: clustering, anomaly detection, and model update. The major challenge while pruning the redundant computations was maintaining the quality of the downstream tasks. Our three research contributions successfully pruned the redundant computation without any significant loss in quality.

In Chapter 3, we prune redundant computation when the machine learning application requires computation of all-pair MBD. We proposed an algorithm ($fMBD$) to compute all-pair MBD faster by optimizing the order of distance computation using Isolation Forest. Our algorithm $fMBD$ computes exact distances without any approximation. The proposed $fMBD$ enables clustering and anomaly detection tasks to execute 2X to 5X faster with the same quality.

In Chapter 4, we prune the redundant computation when a mass-based clustering algorithm only requires a small fraction of all-pair distances. We proposed an algorithm $sMBScan$ that identifies a superset of necessary data point pairs out of all $\binom{n}{2}$ pairs. We compute MBD only for these identified pairs. The superset includes all of the required data point pairs. As a result, there is no loss in the clustering quality. Our $sMBScan$ algorithm saves up to 53X time and up to 96% of memory over the existing $MBScan$ algorithm.

In Chapter 5, we selectively update the existing Isolation Forest structure to handle the concept drift. We proposed $I^2Forest$ that quickly updates each tree in the forest to imitate the model constructed with both old and new data. The proposed approach quickly updates the model with minimal loss in the anomaly detection quality.

6.1 Future Directions

In conclusion our thesis is about making Isolation Forest based unsupervised machine learning algorithms efficient by reducing execution time or memory requirement. In line with the same objective, we discuss the future scope of improvement of the algorithms discussed in the thesis. In addition, we discuss a few other directions we can effectively explore for efficient machine learning utilizing Isolation Forest.

1. **Incorporating Parallelization:** All the algorithms discussed in this thesis are implemented in a serial manner. A further scope of efficiency improvement lies in incorporating parallelization at various stages of the algorithms. The potential stages of parallelization include i) the construction of Isolation Forests across all three algorithms, (ii) the computation of mass-based distances in sMBSCAN and fMBD, and (iii) incremental updates to isolation trees in $I^2Forest$. With this improvement, the running time of the algorithm will be equivalent to that of an Isolation forest consisting of a single isolation tree.

2. **Enhancing Density-Based Clustering and Nearest Neighbor based Tasks:** Our research in sMBSCAN demonstrates the efficacy of using Isolation Forest structures to efficiently identify mass-based distances below a given threshold in density-based clustering. Extending this approach to other clustering methods, such as Density Peak Clustering, presents an exciting avenue for future exploration. Beyond clustering, the Isolation Forest structure also holds the potential for efficiently identifying k-nearest neighbors, which is fundamental to many machine learning and data mining applications. Hence, on similar lines of $sMBSCAN$, many other algorithms can be improved using isolation forest

structure.

3. **Theoretical Exploration of Mass-Based Distance (MBD) as a Metric:** With respect to MBD, this thesis improves the efficiency of the downstream data mining tasks by targeting optimization opportunities using the isolation forest structure. However, exploring the theoretical aspects of mass-based distances has an interesting research scope. Research in this direction will help in answering questions like, Can we approximate the MBD similar to Euclidean distances or any other metric using triangle inequality and optimizations? In our primary observations, MBD is observed to follow triangle inequality; however, it is interesting to dive deeper and research for theoretic justifications.

4. **Representation Learning and Self-Supervised Learning (SSL):** Isolation Forests and related techniques, such as isolation kernels, could contribute significantly to representation learning in deep learning models. A particularly promising direction is their application in contrastive learning-based self-supervised learning (SSL) approaches. In the context of tabular data, where, unlike images, text, or speech, there is no inherent structural consistency across datasets, Isolation Forest-based methods could play a crucial role in defining positive and negative sample pairs for contrastive learning. SSL for tabular data remains an underexplored research area, and integrating Isolation Forest techniques into this domain could lead to novel advancements.

5. **Incremental Learning in Deep Learning Models:** Incremental updates to machine learning models are becoming increasingly critical, especially in the context of large-scale deep learning models. The cost, time, and energy consumption associated with retraining such models pose significant challenges, including environmental concerns related to carbon emissions. Our $I^2Forest$ approach, which focuses on updating only the necessary parts of the model, presents a potential solution for efficient incremental learning in deep learning architectures. Exploring this approach further could lead to substantial improvements in resource-efficient model updates.

Bibliography

[1] Kai Ming Ting, Ye Zhu, Mark Carman, Yue Zhu, and Zhi-Hua Zhou. "Overcoming key weaknesses of distance-based neighbourhood methods using a data dependent dissimilarity measure". In *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1205–1214, New York, NY, USA, 2016. ACM. ISBN 9781450342322. doi: 10.1145/2939672.2939779. URL https://doi.org/10.1145/2939672.2939779.

[2] Stephen D. Bay and Mark Schwabacher. "Mining distance-based outliers in near linear time with randomization and a simple pruning rule". In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 29–38, New York, NY, USA, 2003. ACM. ISBN 1581137370. doi: 10.1145/956750.956758. URL https://doi.org/10.1145/956750.956758.

[3] João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. "A Survey on Concept Drift Adaptation". *ACM Comput. Surv.*, 46(4):1 – 37, 2014. ISSN 0360-0300. doi: 10.1145/2523813. URL https://doi.org/10.1145/2523813.

[4] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest". In *Proceedings of Eighth International Conference on Data Mining*, pages 413–422, Pisa, Italy, 2008. IEEE. ISBN 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.17. URL https://doi.org/10.1109/ICDM.2008.17.

[5] Meng Wang, Weijie Fu, Xiangnan He, Shijie Hao, and Xindong Wu. "A survey on large-scale machine learning". *IEEE Transactions on Knowledge and Data*

Engineering, 34(6):2574–2594, 2022. doi: 10.1109/TKDE.2020.3015777. URL https://doi.org/10.1109/TKDE.2020.3015777.

[6] Alexandra L'Heureux, Katarina Grolinger, Hany F. Elyamany, and Miriam A. M. Capretz. "Machine Learning With Big Data: Challenges and Approaches". *IEEE Access*, 5:7776–7797, 2017. doi: 10.1109/ACCESS.2017.2696365. URL https://doi.org/10.1109/ACCESS.2017.2696365.

[7] Omar Y Al-Jarrah, Paul D Yoo, Sami Muhaidat, George K Karagiannidis, and Kamal Taha. "Efficient machine learning for big data: A review". *Elsevier Big Data Research*, 2(3):87–93, 2015. doi: 10.1016/j.bdr.2015.04.001. URL https://doi.org/10.1016/j.bdr.2015.04.001.

[8] Nicola Segata and Enrico Blanzieri. "Fast Local Support Vector Machines for Large Datasets". In *Proceedings of 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 295–310, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03070-3. doi: 10.1007/978-3-642-03070-3_22. URL https://doi.org/10.1007/978-3-642-03070-3_22.

[9] Fabrizio Angiulli. " Fast Nearest Neighbor Condensation for Large Data Sets Classification". *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1450–1464, 2007. doi: 10.1109/TKDE.2007.190645. URL https://doi.org/10.1109/TKDE.2007.190645.

[10] Xiaoou Li, Jair Cervantes, and Wen Yu. " Fast classification for large data sets via random selection clustering and support vector machines ". *IOS Press Intelligent Data Analysis*, 16(6):897–914, 2012. doi: 10.5555/2595532.2595537. URL https://dl.acm.org/doi/10.5555/2595532.2595537.

[11] Georgios Exarchakis, Omar Oubari, and Gregor Lenz. "A sampling-based approach for efficient clustering in large datasets". In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, page 12403–12412, New Orleans,

Louisiana, 2022. IEEE/CVF. ISBN 978-1-6654-6946-3. doi: 10.1109/CVPR52688.2022.01208. URL https://doi.org/10.1109/CVPR52688.2022.01208.

[12] Hichem Frigui. "SyMP: an efficient clustering approach to identify clusters of arbitrary shapes in large data sets". In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 507–512, New York, NY, USA, 2002. ACM. ISBN 158113567X. doi: 10.1145/775047.775121. URL https://doi.org/10.1145/775047.775121.

[13] Xin Geng and Kate Smith-Miles. *"Incremental Learning"*, pages 731–735. Springer US, Boston, MA, 2009. ISBN 978-0-387-73003-5. doi: 10.1007/978-0-387-73003-5_304. URL https://doi.org/10.1007/978-0-387-73003-5_304.

[14] Alex Rodriguez and Alessandro Laio. "Clustering by fast search and find of density peaks". *AAAS Science*, 344(6191):1492–1496, 2014. doi: 10.1126/science.1242072. URL https://doi.org/10.1126/science.1242072.

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231, Portland, Oregon, 1996. AAAI. URL https://cdn.aaai.org/KDD/1996/KDD96-037.pdf.

[16] Zhiguo Ding and Minrui Fei. "An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window". *IFAC Proceedings Volumes*, 46(20):12–17, 2013. doi: https://doi.org/10.3182/20130902-3-CN-3020.00044. URL https://www.sciencedirect.com/science/article/pii/S1474667016314999.

[17] Michael Heigl, Kumar Ashutosh Anand, Andreas Urmann, Dalibor Fiala, Martin Schramm, and Robert Hable. "On the improvement of the isolation forest algorithm for outlier detection with streaming data". *MDPI Electronics*, 10(13):

1534, 2021. doi: 10.3390/electronics10131534. URL https://doi.org/10.3390/electronics10131534.

[18] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. "Robust random cut forest based anomaly detection on streams". In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, ICML'16, page 2712–2721, New York, NY, USA, 2016. JMLR.org. doi: 10.5555/3045390.3045676. URL https://proceedings.mlr.press/v48/guha16.pdf.

[19] Xin Mu, Kai Ming Ting, and Zhi-Hua Zhou. "Classification under streaming emerging new classes: A solution using completely-random trees". *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1605–1618, 2017. doi: 10.1109/TKDE.2017.2691702. URL https://doi.org/10.1109/TKDE.2017.2691702.

[20] James MacQueen. "Some methods for classification and analysis of multivariate observations". In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–298, 1967.

[21] "Jain, Anil K and Dubes, Richard C". *Algorithms for clustering data*. Prentice-Hall, Inc., 1998. ISBN 013022278X. URL https://dl.acm.org/doi/abs/10.5555/42779.

[22] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977. doi: 10.1111/j.2517-6161.1977. URL https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

[23] Andrew Ng, Michael Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm". *Advances in neural information processing systems*, 14, 2001. doi: 10.1111/j.2517-6161.1977. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf.

[24] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Fast algorithms for

projected clustering". *ACM SIGMoD record*, 28(2):61–72, 1999. doi: 10.1145/304181.304188. URL https://doi.org/10.1145/304181.304188.

[25] Junyuan Xie, Ross Girshick, and Ali Farhadi. "Unsupervised deep embedding for clustering analysis". In *International conference on machine learning*, pages 478–487. PMLR, 2016. URL https://proceedings.mlr.press/v48/xieb16.html.

[26] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". *ACM computing surveys (CSUR)*, 41(3):1–58, 2009. doi: 10.1145/1541880.1541882. URL https://doi.org/10.1145/1541880.1541882.

[27] Edwin M Knox and Raymond T Ng. "Algorithms for mining distance based outliers in large datasets". In *Proceedings of the international conference on very large data bases*, pages 392–403. Citeseer, 1998. URL https://www.vldb.org/conf/1998/p392.pdf.

[28] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. "LOF: identifying density-based local outliers". In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000. doi: 10.1145/342009.335388. URL https://doi.org/10.1145/342009.335388.

[29] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. "Data clustering: a review". *ACM computing surveys (CSUR)*, 31(3):264–323, 1999. doi: 10.1145/331499.331504. URL https://doi.org/10.1145/331499.331504.

[30] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. "Estimating the support of a high-dimensional distribution". *Neural computation*, 13(7):1443–1471, 2001. doi: 10.1162/089976601750264965. URL https://doi.org/10.1162/089976601750264965.

[31] Charu C Aggarwal and Charu C Aggarwal. "Outlier analysis: advanced concepts". *Data Mining: The Textbook*, pages 265–283, 2015. doi: 10.1007/978-3-319-14142-8_9. URL https://doi.org/10.1007/978-3-319-14142-8_9.

[32] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. "A novel anomaly detection scheme based on principal component classifier". In *Proceedings of the IEEE foundations and new directions of data mining workshop*, pages 172–179. IEEE Press Piscataway, NJ, USA, 2003. URL https://neuro.bstu.by/ai/To-dom/My_research/Paper-0-again/For-research/D-mining/Anomaly-D/KDD-cup-99/NN/ICDM03_WS.pdf.

[33] Leman Akoglu, Hanghang Tong, and Danai Koutra. "Graph based anomaly detection and description: a survey". *Data mining and knowledge discovery*, 29: 626–688, 2015. doi: 10.1007/s10618-014-0365-y. URL https://link.springer.com/article/10.1007/s10618-014-0365-y.

[34] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. "Long short term memory networks for anomaly detection in time series". In *Proceedings*, volume 89, page 94, 2015. URL https://web.archive.org/web/20210506232144id_/https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf.

[35] Aitor Sánchez-Ferrera, Borja Calvo, and Jose A Lozano. A review on self-supervised learning for time series anomaly detection: Recent advances and open challenges. *arXiv preprint arXiv:2501.15196*, 2025. URL https://arxiv.org/abs/2501.15196.

[36] Amos Tversky. "Features of similarity". *American Psychological Association Psychological Review*, 84(4):327–352, 1977. URL https://psycnet.apa.org/doi/10.1037/0033-295X.84.4.327.

[37] Dimitar Karev, Christopher McCubbin, and Ruslan Vaulin. "Cyber Threat Hunting Through the Use of an Isolation Forest". In *Proceedings of the 18th International Conference on Computer Systems and Technologies*, CompSysTech '17, page 163–170, New York, NY, USA, 2017. ACM. ISBN 9781450352345. doi: 10.1145/3134302.3134319. URL https://doi.org/10.1145/3134302.3134319.

[38] Asaf Nadler, Avi Aminov, and Asaf Shabtai. "Detection of malicious and low throughput data exfiltration over the DNS protocol". *Elsevier Computers Security*, 80:36–53, 2019. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2018.09.006. URL https://www.sciencedirect.com/science/article/pii/S0167404818304000.

[39] Jawad Ahmed, Hassan Habibi Gharakheili, Qasim Raza, Craig Russell, and Vijay Sivaraman. Real-time detection of dns exfiltration and tunneling from enterprise networks. In *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 649–653. IEEE, 2019. ISBN 978-3-903176-15-7. URL https://ieeexplore.ieee.org/abstract/document/8717806.

[40] Md Amran Siddiqui, Jack W Stokes, Christian Seifert, Evan Argyle, Robert McCann, Joshua Neil, and Justin Carroll. "Detecting cyber attacks using anomaly detection with explanations and expert feedback". In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2872–2876, Brighton, UK, 2019. IEEE. ISBN 978-1-4799-8131-1. doi: 10.1109/ICASSP.2019.8683212. URL https://doi.org/10.1109/ICASSP.2019.8683212.

[41] Charlie Soh, Sicheng Yu, Annamalai Narayanan, Santhiya Duraisamy, and Lihui Chen. "Employee profiling via aspect-based sentiment and network for insider threats detection". *Elsevier Expert Systems with Applications*, 135:351–361, 2019. doi: 10.1016/j.eswa.2019.05.043. URL https://doi.org/10.1016/j.eswa.2019.05.043.

[42] Gaurang Gavai, Kumar Sricharan, Dave Gunning, Rob Rolleston, John Hanley, and Mudita Singhal. "Detecting Insider Threat from Enterprise Social and Online Activity Data". In *Proceedings of the 7th CCS International Workshop on Managing Insider Security Threats*, MIST '15, page 13–20, Denver Colorado USA, 2015. ACM. ISBN 9781450338240. doi: 10.1145/2808783.2808784. URL https://doi.org/10.1145/2808783.2808784.

[43] Duc C Le, Nur Zincir-Heywood, and Malcolm I Heywood. "Analyzing data gran-

ularity levels for insider threat detection using machine learning". *IEEE Transactions on Network and Service Management*, 17(1):30–44, 2020. doi: 10.1109/TNSM.2020.2967721. URL https://doi.org/10.1109/TNSM.2020.2967721.

[44] Anagi Gamachchi, Li Sun, and Serdar Boztas. A graph based framework for malicious insider threat detection. *arXiv preprint arXiv:1809.00141*, 2018. doi: arXiv:1809.00141. URL https://doi.org/10.48550/arXiv.1809.00141.

[45] Rodrigo N Calheiros, Kotagiri Ramamohanarao, Rajkumar Buyya, Christopher Leckie, and Steve Versteeg. "On the effectiveness of isolation-based anomaly detection in cloud data centers". *Wiley Concurrency and Computation: Practice and Experience*, 29(18):4169, 2017. doi: 10.1002/cpe.4169. URL https://doi.org/10.1002/cpe.4169.

[46] Ali Moradi Vartouni, Saeed Sedighian Kashi, and Mohammad Teshnehlab. "An anomaly detection method to detect web attacks using stacked auto-encoder". In *Proceedings of 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*, pages 131–134, Kerman, Iran, 2018. IEEE. ISBN 978-1-5386-2836-2. doi: 10.1109/CFIS.2018.8336654. URL https://doi.org/10.1109/CFIS.2018.8336654.

[47] J Ren, J Guo, W Qian, H Yuan, X Hao, and H Jingjing. "Building an Effective Intrusion Detection System by Using Hybrid Data Optimization Based on Machine Learning Algorithms". *Security and Communication Networks*, 2019(1): 7130868, 2019. doi: 10.1155/2019/7130868. URL https://doi.org/10.1155/2019/7130868.

[48] Xiaomu Luo, Huoyuan Tan, Qiuju Guan, Tong Liu, Hankz Hankui Zhuo, and Baihua Shen. "Abnormal Activity Detection Using Pyroelectric Infrared Sensors". *MDPI Sensors*, 16(6):822, 2016. doi: 10.3390/s16060822. URL https://doi.org/10.3390/s16060822.

[49] Salisu Wada Yahaya, Ahmad Lotfi, and Mufti Mahmud. "A consensus novelty detection ensemble approach for anomaly detection in activities of daily living".

Elsevier Applied Soft Computing, 83:105613, 2019. ISSN 1568-4946. doi: 10.1016/j.asoc.2019.105613. URL https://doi.org/10.1016/j.asoc.2019.105613.

[50] Jessamyn Dahmen, Brian L Thomas, Diane J Cook, and Xiaobo Wang. Activity learning as a foundation for security monitoring in smart homes. *MDPI Sensors*, 17(4):737, 2017. doi: 10.3390/s17040737. URL https://doi.org/10.3390/s17040737.

[51] Mariam Elnour, Nader Meskin, Khaled Khan, and Raj Jain. "A Dual-Isolation-Forests-Based Attack Detection Framework for Industrial Control Systems". *IEEE Access*, 8:36639–36651, 2020. doi: 10.1109/ACCESS.2020.2975066. URL https://doi.org/10.1109/ACCESS.2020.2975066.

[52] Patrick Strauß, Markus Schmitz, René Wöstmann, and Jochen Deuse. "Enabling of Predictive Maintenance in the Brownfield through Low-Cost Sensors, an IIoT-Architecture and Machine Learning". In *Proceedings of International Conference on Big Data (Big Data)*, pages 1474–1483, Seattle, WA, USA, 2018. IEEE. ISBN :978-1-5386-5035-6. doi: 10.1109/BigData.2018.8622076. URL https://doi.org/10.1109/BigData.2018.8622076.

[53] Yuehua Liu, Tharam Dillon, Wenjin Yu, Wenny Rahayu, and Fahed Mostafa. "Noise Removal in the Presence of Significant Anomalies for Industrial IoT Sensor Data in Manufacturing". *IEEE Internet of Things Journal*, 7(8):7084–7096, 2020. doi: 10.1109/JIOT.2020.2981476. URL https://doi.org/10.1109/JIOT.2020.2981476.

[54] Vitor Hugo Bezerra, Victor Guilherme Turrisi da Costa, Sylvio Barbon Junior, Rodrigo Sanches Miani, and Bruno Bogaz Zarpelão. "IoTDS: A One-Class Classification Approach to Detect Botnets in Internet of Things Devices". *Sensors*, 19(14): –, 2019. doi: 10.3390/s19143188. URL https://doi.org/10.3390/s19143188.

[55] Amanda Minnich, Nikan Chavoshi, Danai Koutra, and Abdullah Mueen. "BotWalk: Efficient Adaptive Exploration of Twitter Bot Networks". In *Proceedings of International Conference on Advances in Social Networks Analysis and*

Mining, ASONAM '17, page 467–474, Sydney, Australia, 2017. ACM. ISBN 9781450349932. doi: 10.1145/3110025.3110163. URL https://doi.org/10.1145/3110025.3110163.

[56] Mattia Antonini, Massimo Vecchio, Fabio Antonelli, Pietro Ducange, and Charith Perera. "Smart Audio Sensors in the Internet of Things Edge for Anomaly Detection". *IEEE Access*, 6:67594–67610, 2018. doi: 10.1109/ACCESS.2018.2877523. URL https://doi.org/10.1109/ACCESS.2018.2877523.

[57] Saeed Ahmed, Youngdoo Lee, Seung-Ho Hyun, and Insoo Koo. "Unsupervised Machine Learning-Based Detection of Covert Data Integrity Assault in Smart Grid Networks Utilizing Isolation Forest". *IEEE Transactions on Information Forensics and Security*, 14(10):2765–2777, 2019. doi: 10.1109/TIFS.2019.2902822. URL https://doi.org/10.1109/TIFS.2019.2902822.

[58] Blaž Podgorelec, Muhamed Turkanović, and Sašo Karakatič. A machine learning-based method for automated blockchain transaction signing including personalized anomaly detection. *MDPI Sensors*, 20(1):147, 2019.

[59] Shirshak Raja Maskey, Shahriar Badsha, Shamik Sengupta, and Ibrahim Khalil. "BITS: Blockchain based Intelligent Transportation System with Outlier Detection for Smart City". In *Proceedings of International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–6, Austin, TX, USA, 2020. IEEE. ISBN 978-1-7281-4716-1. doi: 10.1109/PerComWorkshops48775.2020.9156237. URL https://doi.org/10.1109/PerComWorkshops48775.2020.9156237.

[60] Xiguo Yuan, Jiaao Yu, Jianing Xi, Liying Yang, Junliang Shang, Zhe Li, and Junbo Duan. "CNV$_I$$FTV$: $AnIsolationForestandTotalVariation - BasedDetectionofCNVsfromShort - ReadSequencingData$"■. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(2):539–549, 2021. doi: 10.1109/TCBB.2019.2920889. URL https://doi.org/10.1109/TCBB.2019.2920889.

[61] Lin Guo, Lihui Lin, Xiaoshan Wang, Mingwei Gao, Shangtao Cao, Yuanbang Mai, Fang Wu, Junqi Kuang, He Liu, Jiaqi Yang, et al. "Resolving cell fate decisions during somatic cell reprogramming by single-cell RNA-Seq". *Elsevier Molecular Cell*, 73(4):815–829, 2019. doi: 10.1016/j.molcel.2019.01.042. URL https://doi.org/10.1016/j.molcel.2019.01.042.

[62] Eduardo Perez-Careta, Delia Irazú Hernández-Farías, José Rafael Guzman-Sepulveda, Miguel Torres Cisneros, Teodoro Cordoba-Fraga, Juan Carlos Martinez Espinoza, and Rafael Guzman-Cabrera. One-class classification for identifying covid-19 in x-ray images. *Springer Programming and Computer Software*, 48(4):235–242, 2022. doi: 10.1134/S0361768822040041. URL https://doi.org/10.1134/S0361768822040041.

[63] Ruinan Jin and Xiaoxiao Li. "Backdoor attack is a devil in federated gan-based medical image synthesis". In *Proceedings of International Workshop on Simulation and Synthesis in Medical Imaging*, pages 154–165. Springer, 2022. ISBN 978-3-031-16980-9. doi: 10.1007/978-3-031-16980-9_15. URL https://doi.org/10.1007/978-3-031-16980-9_15.

[64] Richard Bauder, Raquel da Rosa, and Taghi Khoshgoftaar. "Identifying Medicare Provider Fraud with Unsupervised Machine Learning". In *Proceedings of International Conference on Information Reuse and Integration (IRI)*, pages 285–292, Salt Lake City, UT, USA, 2018. IEEE. ISBN 978-1-5386-2659-7. doi: 10.1109/IRI.2018.00051. URL https://doi.org/10.1109/IRI.2018.00051.

[65] Ramiro Daniel Camino, Radu State, Leandro Montero, and Petko Valtchev. "Finding Suspicious Activities in Financial Transactions and Distributed Ledgers". In *Proceedings of International Conference on Data Mining Workshops (ICDMW)*, pages 787–796, New Orleans, LA, USA, 2017. IEEE. ISBN 978-1-5386-3800-2. doi: 10.1109/ICDMW.2017.109. URL https://doi.org/10.1109/ICDMW.2017.109.

[66] Shaosheng Cao, XinXing Yang, Cen Chen, Jun Zhou, Xiaolong Li, and Yuan

Qi. "TitAnt: online real-time transaction fraud detection in Ant Financial". *VLDB Endowment Proc. VLDB Endow.*, 12(12):2150–8097, 2019. doi: 10.14778/3352063.3352126. URL https://doi.org/10.14778/3352063.3352126.

[67] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Yacine Kessaci, Frédéric Oblé, and Gianluca Bontempi. "Combining unsupervised and supervised learning in credit card fraud detection". *Elsevier Information Sciences*, 557:317–331, 2021. doi: https://doi.org/10.1016/j.ins.2019.05.042. URL https://www.sciencedirect.com/science/article/pii/S0020025519304451.

[68] Xiaoling Deng, Zihao Zhu, Jiacheng Yang, Zheng Zheng, Zixiao Huang, Xianbo Yin, Shujin Wei, and Yubin Lan. "Detection of citrus huanglongbing based on multi-input neural network model of UAV hyperspectral remote sensing". *MDPI Remote Sensing*, 12(17):2678, 2020. doi: 10.3390/rs12172678. URL https://doi.org/10.3390/rs12172678.

[69] Tymoteusz Cejrowski and Julian Szymański. "Detection of anomalies in bee colony using transitioning state and contrastive autoencoders". *Elsevier Computers and Electronics in Agriculture*, 200:107207, 2022. doi: 10.1016/j.compag.2022.107207. URL https://doi.org/10.1016/j.compag.2022.107207.

[70] Meera Kansara and Ajay Parikh. "Unravel the Outlier Detection for Indian Ayurvedic Plant Organ Image Dataset". In *Proceedings of Third International Conference on Computing, Communications, and Cyber-Security*, pages 417–426, Singapore, 2023. Springer. ISBN 978-981-19-1142-2. doi: 10.1007/978-981-19-1142-2_33. URL https://doi.org/10.1007/978-981-19-1142-2_33.

[71] Olmo Cerri, Thong Q Nguyen, Maurizio Pierini, Maria Spiropulu, and Jean-Roch Vlimant. "Variational autoencoders for new physics mining at the large hadron collider". *Springer Journal of High Energy Physics*, 2019(5):1–29, 2019. doi: 10.1007/JHEP05(2019)036. URL https://doi.org/10.1007/JHEP05(2019)036.

[72] Thea Aarrestad, Melissa van Beekveld, Marcella Bona, Antonio Boveia, Sascha Caron, Joe Davies, Andrea De Simone, Caterina Doglioni, Javier Duarte, Amir Farbin, et al. "The dark machines anomaly score challenge: benchmark data and model independent event classification for the large hadron collider". *SciPost Physics*, 12(1):043, 2022. doi: 10.21468/SciPostPhys.12.1.043. URL https://scipost.org/10.21468/SciPostPhys.12.1.043.

[73] M Crispim Romão, NF Castro, and R Pedro. "Finding new physics without learning about it: anomaly detection as a tool for searches at colliders". *Springer European Physical Journal C*, 81(1):27, 2021. doi: 10.1140/epjc/s10052-020-08807-w. URL https://doi.org/10.1140/epjc/s10052-020-08807-w.

[74] Yan-Bo Wang, Ding-Ge Chang, Shao-Rui Qin, Yu-Hang Fan, Hai-Bao Mu, and Guan-Jun Zhang. "Separating Multi-Source Partial Discharge Signals Using Linear Prediction Analysis and Isolation Forest Algorithm". *IEEE Transactions on Instrumentation and Measurement*, 69(6):2734–2742, 2020. doi: 10.1109/TIM.2019.2926688. URL https://doi.org/10.1109/TIM.2019.2926688.

[75] Prabhas Hundi and Rouzbeh Shahsavari. "Comparative studies among machine learning models for performance estimation and health monitoring of thermal power plants". *Elsevier Applied Energy*, 265:114775, 2020. ISSN 0306-2619. doi: 10.1016/j.apenergy.2020.114775. URL https://doi.org/10.1016/j.apenergy.2020.114775.

[76] Zi Lin, Xiaolei Liu, and Maurizio Collu. Wind power prediction based on high-frequency scada data along with isolation forest and deep learning neural networks. *Elsevier International Journal of Electrical Power Energy Systems*, 118:105835, 2020. ISSN 0142-0615. doi: 10.1016/j.ijepes.2020.105835. URL https://doi.org/10.1016/j.ijepes.2020.105835.

[77] Yanghui Tan, Hui Tian, Ruizheng Jiang, Yejin Lin, and Jundong Zhang. "A comparative investigation of data-driven approaches based on one-class classifiers for condition monitoring of marine machinery system". *Elsevier Ocean Engineering*,

201:107174, 2020. ISSN 0029-8018. doi: 10.1016/j.oceaneng.2020.107174. URL https://doi.org/10.1016/j.oceaneng.2020.107174.

[78] Wei Wang, Shiyou Yang, and Yankun Yang. "An Improved Data-Efficiency Algorithm Based on Combining Isolation Forest and Mean Shift for Anomaly Data Filtering in Wind Power Curve". *MDPI Energies*, 15(13), 2022. ISSN 1996-1073. doi: 10.3390/en15134918. URL https://doi.org/10.3390/en15134918.

[79] Wenliao Du, Zhen Guo, Chuan Li, Xiaoyun Gong, and Ziqiang Pu. "From Anomaly Detection to Novel Fault Discrimination for Wind Turbine Gearboxes With a Sparse Isolation Encoding Forest". *IEEE Transactions on Instrumentation and Measurement*, 71:1–10, 2022. doi: 10.1109/TIM.2022.3187737. URL https://doi.org/10.1109/TIM.2022.3187737.

[80] Gian Antonio Susto, Alessandro Beghi, and Seán McLoone. "Anomaly detection through on-line isolation Forest: An application to plasma etching". In *Proceedings of 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 89–94. IEEE, 2017. doi: 10.1109/ASMC.2017.7969205. URL https://doi.org/10.1109/ASMC.2017.7969205.

[81] Luca Puggini and Seán McLoone. "An enhanced variable selection and Isolation Forest based methodology for anomaly detection with OES data". *Elsevier Engineering Applications of Artificial Intelligence*, 67:126–135, 2018. ISSN 0952-1976. doi: 10.1016/j.engappai.2017.09.021. URL https://doi.org/10.1016/j.engappai.2017.09.021.

[82] Jianzhou Wang, Wendong Yang, Pei Du, and Tong Niu. "Outlier-robust hybrid electricity price forecasting model for electricity market management". *Elsevier Journal of Cleaner Production*, 249:119318, 2020. ISSN 0959-6526. doi: 10.1016/j.jclepro.2019.119318. URL https://doi.org/10.1016/j.jclepro.2019.119318.

[83] Yassine Himeur, Khalida Ghanem, Abdullah Alsalemi, Faycal Bensaali, and Abbes Amira. "Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives". *Elsevier Applied*

Energy, 287:116601, 2021. ISSN 0306-2619. doi: 10.1016/j.apenergy.2021.116601. URL https://doi.org/10.1016/j.apenergy.2021.116601.

[84] Wenqian Liu, Jingsha He, Song Han, Fangbo Cai, Zhenning Yang, and Nafei Zhu. "A Method for the Detection of Fake Reviews Based on Temporal Features of Reviews and Comments". *IEEE Engineering Management Review*, 47(4):67–79, 2019. doi: 10.1109/EMR.2019.2928964. URL https://doi.org/10.1109/EMR.2019.2928964.

[85] Qiang Lin, Daqing Zhang, Kay Connelly, Hongbo Ni, Zhiwen Yu, and Xingshe Zhou. "Disorientation detection by mining GPS trajectories for cognitively-impaired elders". *Elsevier Pervasive and Mobile Computing*, 19:71–85, 2015. ISSN 1574-1192. doi: 10.1016/j.pmcj.2014.01.003. URL https://doi.org/10.1016/j.pmcj.2014.01.003.

[86] Yuan Yuan, Dong Wang, and Qi Wang. "Anomaly Detection in Traffic Scenes via Spatial-Aware Motion Reconstruction". *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1198–1209, 2017. doi: 10.1109/TITS.2016.2601655. URL https://doi.org/10.1109/TITS.2016.2601655.

[87] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. "A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems". *Elsevier Engineering Applications of Artificial Intelligence*, 85:634–644, 2019. ISSN 0952-1976. doi: 10.1016/j.engappai.2019.07.008. URL https://doi.org/10.1016/j.engappai.2019.07.008.

[88] Francisco Alonso-Sarria, Carmen Valdivieso-Ros, and Francisco Gomariz-Castillo. "Isolation Forests to Evaluate Class Separability and the Representativeness of Training and Validation Areas in Land Cover Classification". *MDPI Remote Sensing*, 11(24), 2019. ISSN 2072-4292. doi: 10.3390/rs11243000. URL https://doi.org/10.3390/rs11243000.

[89] Sahand Hariri, Matias Carrasco Kind, and Robert J Brunner. "Extended isolation forest". *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1479–

1489, 2019. doi: 10.1109/TKDE.2019.2947676. URL https://doi.org/10.1109/TKDE.2019.2947676.

[90] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "On detecting clustered anomalies using sciforest". In *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 274–290. Springer, 2010. ISBN 978-3-642-15883-4. doi: 10.1007/978-3-642-15883-4_18. URL https://doi.org/10.1007/978-3-642-15883-4_18.

[91] Mikhail Tokovarov and Paweł Karczmarek. "A probabilistic generalization of isolation forest". *Elsevier Information Sciences*, 584:433–449, 2022. ISSN 0020-0255. doi: 10.1016/j.ins.2021.10.075. URL https://doi.org/10.1016/j.ins.2021.10.075.

[92] Paweł Karczmarek, Adam Kiersztyn, Witold Pedrycz, and Ebru Al. "K-Means-based isolation forest". *Elsevier Knowledge-Based Systems*, 195:105659, 2020. ISSN 0950-7051. doi: 10.1016/j.knosys.2020.105659. URL https://doi.org/10.1016/j.knosys.2020.105659.

[93] Łukasz Gałka, Paweł Karczmarek, and Mikhail Tokovarov. "Isolation Forest Based on Minimal Spanning Tree". *IEEE Access*, 10:74175–74186, 2022. doi: 10.1109/ACCESS.2022.3190505. URL https://doi.org/10.1109/ACCESS.2022.3190505.

[94] Xuyun Zhang, Wanchun Dou, Qiang He, Rui Zhou, Christopher Leckie, Ramamohanarao Kotagiri, and Zoran Salcic. "LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis". In *Proceedings of 33rd international conference on data engineering (ICDE)*, pages 983–994, San Diego, CA, USA, 2017. IEEE. ISBN 978-1-5090-6543-1. doi: 10.1109/ICDE.2017.145. URL https://doi.org/10.1109/ICDE.2017.145.

[95] Haolong Xiang, Zoran Salcic, Wanchun Dou, Xiaolong Xu, Lianyong Qi, and Xuyun Zhang. "OPHiForest: order preserving hashing based isolation forest for

robust and scalable anomaly detection". In *Proceedings of the 29th International Conference on Information & Knowledge Management*, CIKM '20, pages 1655–1664, Virtual Event, Ireland, 2020. ACM. ISBN 9781450368599. doi: 10.1145/3340531.3411988. URL https://doi.org/10.1145/3340531.3411988.

[96] Sunil Aryal, Kai Ming Ting, Jonathan R. Wells, and Takashi Washio. "Improving iForest with Relative Mass". In *Proceedings of Advances in Knowledge Discovery and Data Mining*, pages 510–521, Cham, 2014. Springer. doi: 10.1007/978-3-319-06605-9_42. URL https://doi.org/10.1007/978-3-319-06605-9_42.

[97] Antonella Mensi and Manuele Bicego. "Enhanced anomaly scores for isolation forests". *Elsevier Pattern Recognition*, 120:108115, 2021. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108115. URL https://doi.org/10.1016/j.patcog.2021.108115.

[98] Ye Zhu, Kai Ming Ting, Yuan Jin, and Maia Angelova. "Hierarchical clustering that takes advantage of both density-peak and density-connectivity". *Elsevier Information Systems*, 103:101871, 2022. doi: 10.1016/j.is.2021.101871. URL https://doi.org/10.1016/j.is.2021.101871.

[99] UCI ML. Uci machine learning repository, -. URL http://archive.ics.uci.edu/ml.

[100] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI, 1996. doi: 10.5555/3001460.3001507. URL https://cdn.aaai.org/KDD/1996/KDD96-037.pdf.

[101] Levent Ertoz, Michael Steinbach, and Vipin Kumar. "Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data". In *Proceedings of the 2003 International Conference on Data Mining (SDM)*, pages 47–58.

SIAM, 2003. doi: 10.1137/1.9781611972733.5. URL https://doi.org/10.1137/1.9781611972733.5.

[102] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. "OPTICS: ordering points to identify the clustering structure". In *Proceedings of the International Conference on Management of Data*, SIGMOD '99, page 49–60, Philadelphia, Pennsylvania, USA, 1999. ACM. ISBN 1581130848. doi: 10.1145/304182.304187. URL https://doi.org/10.1145/304182.304187.

[103] Carol L Krumhansl. "Concerning the applicability of geometric models to similarity data: The interrelationship between similarity and spatial density." *APA Psychological Review*, 85(5):445–463, 1978. doi: 10.1037/0033-295X.85.5.445. URL https://psycnet.apa.org/doi/10.1037/0033-295X.85.5.445.

[104] Mohammad Rezaei and Pasi Fränti. "Set matching measures for external cluster validity". *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2173–2186, 2016. doi: 10.1109/TKDE.2016.2551240. URL https://doi.org/10.1109/TKDE.2016.2551240.

[105] Sami Sieranoja and Pasi Fränti. "Fast and general density peaks clustering". *Elsevier Pattern recognition letters*, 128:551–558, 2019. doi: 10.1016/j.patrec.2019.10.019. URL https://doi.org/10.1016/j.patrec.2019.10.019.

[106] Pasi Fränti and Olli Virmajoki. "Iterative shrinking method for clustering problems". *Elsevier Pattern Recognition*, 39(5):761–775, 2006. doi: 10.1016/j.patcog.2005.09.012. URL https://doi.org/10.1016/j.patcog.2005.09.012.

[107] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://doi.org/10.1109/5.726791.

[108] F. Alimoglu and E. Alpaydin. "Combining multiple representations and classifiers for pen-based handwritten digit recognition". In *Proceedings of the*

Fourth International Conference on Document Analysis and Recognition, volume 2, pages 637–640. IEEE, 1997. doi: 10.1109/ICDAR.1997.620583. URL https://doi.org/10.1109/ICDAR.1997.620583.

[109] Peter W Frey and David J Slate. Letter recognition using Holland-style adaptive classifiers. *Springer Machine learning*, 6(2):161–182, 1991. doi: 10.1007/BF00114162. URL https://doi.org/10.1007/BF00114162.

[110] Cor J. Veenman, Marcel J. T. Reinders, and Eric Backer. "A maximum variance cluster algorithm". *IEEE Transactions on pattern analysis and machine intelligence*, 24(9):1273–1280, 2002. doi: 10.1109/TPAMI.2002.1033218. URL https://doi.org/10.1109/TPAMI.2002.1033218.

[111] "Manning, Christopher D. and Raghavan, Prabhakar and Schütze, Hinrich". *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN 978-0-521-86571-5. URL http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[112] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". *ACM Comput. Surv.*, 41(3), jul 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL https://doi.org/10.1145/1541880.1541882.

[113] Charu C. Aggarwal. *"Outlier Analysis"*. Springer, 2017. doi: 10.1007/978-3-319-47578-3.

[114] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4393–4402. PMLR, 10–15 Jul 2018.

[115] Maurras Ulbricht Togbe, Mariam Barry, Aliou Boly, Yousra Chabchoub, Raja Chiky, Jacob Montiel, and Vinh-Thuy Tran. "Anomaly detection for data streams

based on isolation forest using scikit-multiflow". In *Proceedings of 20th International Conference Computational Science and Its Applications*, pages 15–30, Cagliari, Italy, 2020. Springer. ISBN 978-3-030-58811-3. doi: 10.1007/978-3-030-58811-3_2. URL https://doi.org/10.1007/978-3-030-58811-3_2.

[116] Maurras Ulbricht Togbe, Yousra Chabchoub, Aliou Boly, Mariam Barry, Raja Chiky, and Maroua Bahri. "Anomalies detection using isolation in concept-drifting data streams". *MDPI Computers*, 10(1):13, 2021. doi: 10.3390/computers10010013. URL https://doi.org/10.3390/computers10010013.

[117] Vinicius MA Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. "Challenges in benchmarking stream learning algorithms with real-world data". *Springer Data Mining and Knowledge Discovery*, 34(6):1805–1858, 2020. doi: 10.1007/s10618-020-00698-5. URL https://doi.org/10.1007/s10618-020-00698-5.

Publications

Manuscript Published

1. **Nidhi Ahlawat** Nidhi Ahlawat and Amit Awekar. "Scaling Up Mass-Based Clustering". In Proceedings of the 31st ACM International Conference on Information Knowledge Management, CIKM '22, page 3781–3785, Atlanta, GA, USA, 2022. ACM. ISBN 9781450392365. doi: 10.1145/3511808.3557691. URL https://doi.org/10.1145/3511808.3557691

2. **Nidhi Ahlawat** Nidhi Ahlawat and Amit Awekar. "Incremental Isolation Forest to Handle Concept Drift in Anomaly Detection". In Proceedings of the 7th ACM International Conference on Data Science and Management of Data, CODS-COMAD'24, page 582–583, Bangalore, India, 2024. ACM. ISBN 9798400716348. doi:10.1145/3632410.3632486. URL.https://doi.org/10.1145/3632410.3632486

Manuscript Under Review

1. **Nidhi Ahlawat** and Amit Awekar. *Fast Computation of All Pairwise Mass-based Distances.*

2. **Nidhi Ahlawat** and Amit Awekar. *Incremental Isolation Forest to Handle Concept Drift in Anomaly Detection.* [Extended Version for Journal]