

# NOVEL METHODS FOR DISTRIBUTED AND PRIVACY-PRESERVING DATA STREAM MINING

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY  
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF COMPUTER AND INFORMATION SCIENCES

Supervisors

Associate Professor Russel Pears

Dr. Muhammad Asif Naeem

March 2019

By

Benjamin James Denham

School of Engineering, Computer and Mathematical Sciences

# Abstract

The growing number of “big” datasets present many opportunities for data mining, but also raise a variety of new challenges. Datasets may take the form of continuous streams with constantly changing patterns, they may be too widely distributed to be centralised for analysis at a single location, or they may contain sensitive values that data owners are not willing to share due to privacy concerns. Much past research has considered these issues individually, but few existing methods can address combinations of these properties. Therefore, this research develops methods for distributed and privacy-preserving data stream mining: a novel Hierarchical Distributed Stream Miner (HDSM) that learns relationships between the features of separate streams with minimal data transmission to central locations, and two data perturbation methods for privacy-preserving stream mining based on the combination of random projection, random translation, and additive noise. Experimental evaluation of HDSM demonstrates significant improvements in classification accuracy over existing distributed stream mining approaches while minimising data transmission and computational costs. HDSM’s ability to dynamically trade-off accuracy with these costs is also demonstrated. Variations of the known input-output Maximum A Posteriori (MAP) attack are developed to experimentally evaluate the data perturbation methods, and the proposed composite methods are shown to achieve a better trade-off between privacy and model accuracy than random projection alone. Finally, an approach is described for combining HDSM with data perturbation to achieve distributed privacy-preserving stream mining.

# Contents

<b>Abstract</b>	<b>2</b>
<b>Attestation of Authorship</b>	<b>9</b>
<b>Publications</b>	<b>10</b>
<b>Acknowledgements</b>	<b>11</b>
<b>Glossary</b>	<b>12</b>
<b>1 Introduction</b>	<b>16</b>
1.1 Data Stream Mining . . . . .	17
1.2 Distributed Data Mining . . . . .	18
1.3 Privacy-Preserving Data Mining . . . . .	18
1.4 Research Objective and Contributions . . . . .	20
1.5 Thesis Structure . . . . .	21
<b>2 Related Work in Distributed Data Mining</b>	<b>23</b>
2.1 Existing Methods for Vertically Distributed Data Mining . . . . .	24
2.2 Foundational Distributed Data Mining Approach . . . . .	25
<b>3 Proposed HDSM Architecture for Distributed Stream Mining</b>	<b>27</b>
3.1 Trouble Site Hierarchies . . . . .	27
3.2 Protocol for Regulating Data Transmission . . . . .	32
3.3 Learning Trouble Site Hierarchies . . . . .	35
3.3.1 Monitor Thresholds . . . . .	38
3.4 Alternative Aggregation Methods . . . . .	40
<b>4 Experimental Evaluation of HDSM</b>	<b>41</b>
4.1 Performance Evaluation Metrics . . . . .	42
4.2 Demonstration of Dynamic Trouble Site Hierarchy . . . . .	43
4.3 HDSM Performance Evaluation and Comparison . . . . .	48
4.4 Suitability of HDSM for Anytime Classification . . . . .	57
4.5 HDSM Parameter Sensitivity Analysis . . . . .	60
4.6 Limitations and Applicability of HDSM . . . . .	61

<b>5</b>	<b>Variations of HDSM</b>	<b>63</b>
5.1	Batch Transmission . . . . .	63
5.2	Two-Phase Trouble Record Transmission . . . . .	64
5.3	Alternative Confidence Threshold . . . . .	66
5.4	Alternative Trouble Record Selection . . . . .	69
5.4.1	Alternative Metric for Selecting Trouble Records . . . . .	72
5.4.2	Communication of Confidence Distributions . . . . .	74
5.4.3	Selecting Trouble Sites . . . . .	77
5.5	Additional Variations . . . . .	78
5.6	Section Summary . . . . .	78
<b>6</b>	<b>Related Work in Privacy-Preserving Data Mining</b>	<b>79</b>
6.1	Existing Techniques for PPDM and PPDP . . . . .	79
6.1.1	Secure Multiparty Computation . . . . .	79
6.1.2	Anonymisation . . . . .	80
6.1.3	$\epsilon$ -Differential Privacy . . . . .	80
6.1.4	Data Perturbation . . . . .	81
6.2	Methods for Privacy-Preserving Data Perturbation . . . . .	82
6.2.1	Additive Noise . . . . .	82
6.2.2	Random Rotation . . . . .	83
6.2.3	Random Projection . . . . .	84
6.3	Attacks on Random Projection . . . . .	84
6.3.1	Known Sample Attacks . . . . .	85
6.3.2	Known Projection Matrix Attacks . . . . .	85
6.3.3	Known Input-Output Attacks . . . . .	85
<b>7</b>	<b>Proposed Perturbation Methods for Privacy-Preserving Stream Mining</b>	<b>87</b>
7.1	Data Perturbation Methods . . . . .	87
7.1.1	Foundational Random Projection Model . . . . .	87
7.1.2	Random Projection with Independent Noise . . . . .	89
7.1.3	Random Projection with Cumulative Noise . . . . .	90
7.1.4	Comparison of Independent and Cumulative Noise . . . . .	91
7.1.5	Interpolating Cumulative Noise Between Known Points . . . . .	93
7.1.6	Data Perturbation Efficiency . . . . .	94
7.1.7	Applying Noise to the Random Projection Matrix . . . . .	95
7.2	Known Input-Output Attacks . . . . .	95
7.2.1	Notation . . . . .	96
7.2.2	Known Input-Output MAP Attack on Random Projection . . . . .	96
7.2.3	Extended MAP Attack for Random Translation . . . . .	98
7.2.4	Extended MAP Attack for Independent Noise . . . . .	99
7.2.5	Extended MAP Attack for Cumulative Noise . . . . .	101
7.2.6	Numerical Optimisation . . . . .	103
7.2.7	Attacks When $p \geq m$ . . . . .	104

<b>8</b>	<b>Experimental Evaluation of Proposed Perturbation Methods</b>	<b>106</b>
8.1	Experimental Setup . . . . .	107
8.2	Datasets . . . . .	108
8.3	Attack Type Comparison . . . . .	111
8.3.1	Attack Execution Time Comparison . . . . .	113
8.4	Perturbation Method Comparison . . . . .	116
8.5	Trend Analysis for Cumulative Noise Perturbation . . . . .	122
<b>9</b>	<b>Integrating Data Perturbation Methods into Distributed Stream Mining</b>	<b>126</b>
9.1	Secure Multiparty Data Perturbation . . . . .	127
9.1.1	Decomposing the Perturbation Process for Distributed Computation . . . . .	128
9.1.2	Secure Sum Protocol for Partial Perturbed Records . . . . .	131
<b>10</b>	<b>Conclusions and Future Work</b>	<b>135</b>
10.1	Research Achievements . . . . .	135
10.2	Limitations . . . . .	137
10.3	Future Work in Distributed Data Stream Mining . . . . .	138
10.4	Future Work in Privacy-Preserving Data Stream Mining . . . . .	139
	<b>References</b>	<b>141</b>
	<b>Appendices</b>	<b>147</b>
<b>A</b>	<b>Quantile Threshold Examples and Proofs</b>	<b>147</b>
A.1	Agreement Extrema . . . . .	147
A.2	Example of Agreement in the Case of Unequal Record Transmissions	148
A.3	Reduction Coefficient Extrema . . . . .	149
A.4	Two-Phase Reduction Coefficients Extrema . . . . .	149
A.5	First-Phase Transmission Equality . . . . .	151
A.6	Two-Phase Transmission Limits . . . . .	151
<b>B</b>	<b>HDSM Data Compression</b>	<b>153</b>
B.1	Key Compression . . . . .	153
B.2	Feature Compression . . . . .	155
<b>C</b>	<b>Omitting Unconfident Classifications in HDSM</b>	<b>157</b>

# List of Tables

4.1	Effects of HDSM’s trouble site hierarchy on performance with a concept drifting data stream. . . . .	45
4.2	Properties of datasets used for distributed stream mining performance evaluation. . . . .	50
4.3	Distributed stream mining accuracy comparison. . . . .	52
4.4	Distributed stream mining resource time comparison. . . . .	55
4.5	HDSM performance as a function of trouble factor. . . . .	56
4.6	Sensitivity analysis of HDSM parameters. . . . .	61
4.7	Comparison of HDSM and centralised model accuracy. . . . .	62
5.1	Example distributions of schemas in example data stream. . . . .	70
5.2	Example classification confidences assigned to example schemas at primary sites. . . . .	70
8.1	Properties of datasets used for experimental evaluation. . . . .	108
8.2	Attack types evaluated for each perturbation method. . . . .	112
8.3	Perturbation method legend for Figures 8.3, 8.4, and 8.5. . . . .	116

# List of Figures

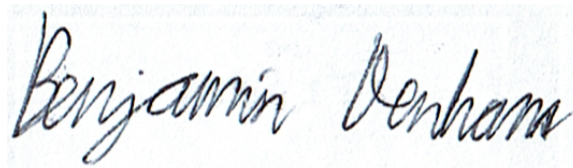
3.1	Physical view of the HDSM architecture. . . . .	28
3.2	Continuous process for creating, removing, and un-blacklisting HDSM trouble sites. . . . .	36
4.1	Timeline of accuracy and data transmission for the synthetic cross-term dataset with concept drift points. . . . .	46
4.2	Timeline of trouble site [3, 2] creation (agreement) and removal (utilisation and accuracy) monitors for the synthetic cross-term dataset with concept drift points. . . . .	47
4.3	Critical difference diagram showing the statistically significant differences in accuracy between distributed stream mining algorithms. . . . .	52
4.4	Accuracy and resource time ranks for HDSM Voting and DStack classifiers. . . . .	57
4.5	Effect of anytime classification on HDSM accuracy and resource time. . . . .	59
5.1	Plot of expected agreement as the proportion transmitted from primary sites to trouble sites is varied for the example data stream. . . . .	71
5.2	Plot of expected agreement as the proportion transmitted from primary sites to trouble sites is varied for the example data stream when trouble records are selected according to the $T$ score. . . . .	73
7.1	Comparison of independent and cumulative noise when the total noise of each approach is equal. . . . .	92
8.1	Comparison of MAP attacks on perturbation with RPIN and RPCN applied to the TAXI dataset. Legend in Table 8.2. . . . .	114
8.2	Comparison of the execution time of MAP attacks on the TAXI dataset. . . . .	115
8.3	Trade-off between accuracy and privacy for different perturbation methods applied to the RBF dataset. . . . .	117
8.4	Trade-off between accuracy and privacy for different perturbation methods applied to the TAXI dataset. . . . .	118
8.5	Trade-off between accuracy and privacy for different perturbation methods applied to the ELEC dataset. . . . .	119
8.6	Critical difference diagram for the accuracy/privacy trade-off evaluation of perturbation methods at different noise levels. . . . .	121

8.7	Trends of record recovery as the distance between known and unknown records increases in a data stream perturbed with RPCN. . . . .	122
8.8	Trends of accuracy over the course of data streams perturbed with RP, RPIN, and RPCN. . . . .	124
8.9	Trends of ARF tree depth over the course of the TAXI data stream perturbed with RP, RPIN, and RPCN. . . . .	125



# Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

A handwritten signature in black ink that reads "Benjamin Denham". The signature is written in a cursive style with a large initial 'B' and 'D'.

---

Signature of student

# Publications

Denham, B., Pears, R., & Naeem, M. A. (2018). HDSM: A Distributed Data Mining Approach to Classifying Vertically Distributed Data Streams. Manuscript submitted for publication to Knowledge-Based Systems.

Denham, B., Pears, R., & Naeem, M. A. (2019). Combining Random Projection and Additive Noise for Privacy-Preserving Data Stream Mining. Manuscript in preparation for submission to Knowledge-Based Systems.

# Acknowledgements

Firstly, my chief thanks is to my supervisors Associate Professor Russel Pears and Dr. Muhammad Asif Naeem for their contributions and support in this research. I have learned a great deal from them about research and data mining, and their insightful feedback has challenged me to strive for deeper understanding. I would also like to acknowledge all of my past lecturers and teachers who have equipped me with the skills and knowledge I used in completing this thesis.

Thanks to my past and present colleagues at Catalyst and Spectrum, who have endured my endless monologues about machine learning and Clojure. I am especially grateful to my managers Katrina Bassett and Patrick Brennan, who have been incredibly supportive of my postgraduate studies. Also, a special thanks to Dr Grant Paton-Simpson for his consistently wise advice regarding work, study, and life.

Thank you to all my friends and family for their love and encouragement. Thanks particularly to Berend and Dieuwe de Boer, my mentor and best friend, who introduced me to and have guided me in the art of software development. My utmost thanks must go to my mother Shona, who has been the greatest supporter of all my endeavours, and to my father Paul, who sparked my life-long interest in computers.

Finally, I give thanks to my God, who has provided me with opportunities and strength throughout my life, and to whose glory I have performed this research.

# Glossary

Term	Definition
$\varepsilon$ -privacy breach	An attempt to recover the original values of a perturbed record that results in a relative error less than $\varepsilon$ .
A-RP	A type of known input-output MAP attack against the RP data perturbation method.
A-RPCN	An extension of the A-RP attack type that accounts for the cumulative noise of the RPCN data perturbation method.
A-RPCN-1	A variation of the A-RPCN attack type that only uses the single known input-output pair that is closest in the stream to the unknown record that is being attacked.
A-RPIN	An extension of the A-RP attack type that accounts for the independent noise of the RPIN data perturbation method.
A-RPIN-1	A variation of the A-RPIN attack type that only uses the single known input-output pair that is closest in the stream to the unknown record that is being attacked.
Additive noise	A data perturbation method that involves adding values drawn from a random distribution to data values.
Agreement	The proportion of trouble records transmitted to a trouble site that were processed at the trouble site because fragments were received from all source sites.
Aggregator	The site in the HDSM architecture that receives classification results from all primary and trouble sites and aggregates them to produce the final classification for each record.
Concept drift	Changes in the underlying patterns of a data stream that can occur abruptly or gradually over time.
Confidence threshold	In HDSM, record fragments classified with a confidence below their site's confidence threshold will be forwarded to a trouble site as a trouble record.

*Continued over page*

<b>Term</b>	<b>Definition</b>
Cross-term	A relationship or pattern among the record features distributed across multiple sites that can only be discovered by viewing the data from the combined perspective of those sites.
Cumulative noise	A form of additive noise that perturbs a data stream by adding to each record the sum of a small random value and all of the random values that were added to previous records in the stream.
DDM	Distributed Data Mining: A class of techniques for performing data mining when data is distributed across multiple, geographically separate sites.
Data perturbation	A method for PPDM or PPDP that involves distorting data values to prevent the original values being discovered by an attacker.
Distance-preserving perturbation	A class of data perturbation methods that do not change the magnitudes of the difference vectors between pairs of record vectors; a property that preserves utility for common data mining tasks.
HDSM	Hierarchical Distributed Stream Miner: A novel architecture proposed by this research for distributed data stream mining that makes use of a hierarchy of trouble sites to minimise data transmission while maximising model accuracy.
Horizontal distribution	The distribution of data records across sites such that each site has the same set of features for a unique set of records.
Independent noise	A form of additive noise that perturbs a set or stream of data records by adding a random value to each record.
Known I/O attack	Known input-output attack: A class of privacy-breaching attack types that attempts to recover the original values of a record by making use of prior knowledge in the form of a set of known input-output pairs.
Known input-output pair	A form of prior knowledge for a privacy-breaching attack: a known record from the original dataset and its perturbed equivalent.
MAP attack	Maximum A-Posteriori attack: A class of privacy-breaching attack types that utilise a Bayesian model to estimate the original values of perturbed records based on some prior knowledge.

*Continued over page*

<b>Term</b>	<b>Definition</b>
PPDM	Privacy-Preserving Data Mining; A class of techniques for performing data mining without disclosing sensitive information from the raw data to the data miner.
PPDP	Privacy-Preserving Data Publishing; A class of techniques for removing sensitive information from datasets so that they may be published for use by data miners.
Prequential evaluation	An execution environment for evaluating online classification accuracy where the classifier is used to classify and is then immediately trained on each record.
Primary site	A site in the HDSM architecture that initially receives a vertical fragment of each record, which it uses to build a local classification model.
RP	A data perturbation method based on a combination of random projection and random translation.
RPCN	An extension of the RP data perturbation method that incorporates cumulative additive noise.
RPIN	An extension of the RP data perturbation method that incorporates independent additive noise.
Random projection	An approximately distance-preserving data perturbation method that involves performing a multiplication of each record with a randomly generated matrix to project each record into a different feature space that may be of a different dimensionality.
Random translation	A distance-preserving data perturbation method that involves adding to each record value a fixed value that is generated randomly for each feature, such that each record is translated by the same fixed vector.
Record fragment	A partial record consisting of only a subset of the features for that record. More specifically known as a vertical record fragment.
Relative error	The magnitude of the Euclidean distance between a data record vector and its estimate recovered by a privacy-breaching attack, normalised to the magnitude of the original record vector.
SMC	Secure Multiparty Computation: A class of techniques for performing distributed computations involving data from multiple parties in such a way that the correct result can be produced without any party gaining knowledge of the raw data of any other party.
Source site	A primary or trouble site that may forward trouble records to another trouble site.

*Continued over page*

---

<b>Term</b>	<b>Definition</b>
Trouble factor	A configuration parameter for each trouble site in the HDSM architecture that determines the proportion of the data stream that will be transmitted to that trouble site in the form of trouble records.
Trouble record	A record fragment transmitted from a source site to a trouble site because it was classified with low confidence.
Trouble site	A site in the HDSM architecture that receives and joins trouble records from a set of source sites, which it uses to build a classification model to capture cross-terms between the source sites' feature sets.
Trouble site blacklist monitor	A monitor in the HDSM architecture that performs concept drift detection to determine when to remove a trouble site from the blacklist so that it may be re-created.
Trouble site creation monitor	A monitor in the HDSM architecture that determines when to create a new trouble site for a potential set of source sites based on the level of agreement among the source sites.
Trouble site order	The depth of a trouble site in the HDSM architecture's site hierarchy. Each primary site can be considered to be of order 0, and each trouble site's order is one greater than the maximum order among its source sites.
Trouble site removal monitor	A monitor in the HDSM architecture that determines when to remove and blacklist a trouble site for either low utilisation as the final classification or low classification accuracy.
Vertical distribution	The distribution of data records across sites such that each site has a unique set of features for the same set of records.

---

# Chapter 1

## Introduction

The dawn of the Big Data age has presented many opportunities for applying data mining to discover patterns across large and diverse collections of data. However, mining these big datasets also presents a number of challenges. Typical data mining methods often assume that a dataset:

- Is of a fixed-size and is a representative sample of a static population (i.e. there are no continuous temporal dynamics in the dataset's patterns).
- Is available for analysis at a single physical location.
- Can be disclosed in its entirety to data miners without concerns relating to the privacy of individuals represented in the dataset.

However, none of these assumptions may apply for many datasets that are promising candidates for data mining. Increasing numbers of home and personal devices can now be considered part of the Internet of Things (IoT), and are continuously producing streams of sensor readings. Together these IoT devices form distributed sensor networks that can be mined to better understand environments and user behaviour. However, producing a model from the readings of such a network requires continual adaptation



to changes in the underlying patterns over time. Furthermore, it can be impractical to transmit all readings to a single location for centralised analysis when (1) there are many sensors in the network, (2) sensors are widely distributed geographically, or (3) new readings are produced too frequently. Finally, if sensor measurements are related to the private information of one or more individuals or organisations, then sharing the data in its raw form may result in breaches of privacy. Personal geolocation readings present obvious privacy concerns, but there are also less obvious risks. For example, it has been demonstrated that electricity smart meter readings can reveal daily household routines and even the types of appliances in use (Basu, Debusschere & Bacha, 2013). Some organisations may also wish to share their private data streams for collective analysis, such as medical facilities sharing anonymised patient records for the purpose of rapidly detecting international disease outbreaks (Obenshain, 2004).

In response to each of these challenges, researchers have developed the fields of data stream mining, distributed data mining, and privacy-preserving data mining.

## 1.1 Data Stream Mining

Data stream mining algorithms have been developed to learn from continuous streams of data records (Bifet & Kirkby, 2009). In order to cater for high stream velocity, such algorithms are designed to scan and process each record only once, and therefore avoid the storage of a large sample of records for training/learning. In the case of supervised learning algorithms, this also means a prediction for each record can be produced as soon as it is received, enabling near-real-time analysis. Additionally, these algorithms are designed to adapt to changes in the underlying patterns of the data stream (so-called concept drift).

Online classifiers are an example of data stream mining algorithms. Online classifiers are supervised learners that use nominal and numeric features to assign one of a set

of class values to each record that is processed. As online classification is an example of a typical stream mining problem, it is used to describe and evaluate many of the methods developed in this research, although the methods are amenable to other stream mining problems.

## **1.2 Distributed Data Mining**

It is not always possible to analyse a dataset in one physical location. Some datasets are so large that they must be distributed for efficient processing, while others are inherently geographically distributed across autonomous sites. Even if data centralisation is feasible, it is still wasteful of network resources to unnecessarily transmit data that can be mined locally. Research into distributed data mining (DDM) has sought to address these issues by devising methods of combining distributed “local” data mining models into “global” models while minimising data transmission costs (Devi, 2014).

It is common for data streams to be mined to also be inherently distributed, such as streams originating from mobile devices (Rehman, Liew, Wah & Khan, 2017) or the logs of multiple web-servers that can be used to model customer behaviour and detect anomalies in web traffic (Samuelsson, 2016).

## **1.3 Privacy-Preserving Data Mining**

Many individuals and organisations are naturally reluctant to share their data when it is sensitive and may compromise their privacy, thus limiting the availability of such valuable data for mining. The consequences of inadequately considering privacy before sharing data are apparent in the case of the Netflix Prize dataset, where it was shown that individual users could be identified by cross-referencing their public movie ratings on the Internet Movie Database (Narayanan & Shmatikov, 2008).

In order to address such concerns, the data mining community has developed techniques for privacy-preserving data mining (PPDM) and privacy-preserving data publishing (PPDP) (T. Wang, Zheng, Rehmani, Yao & Huo, 2018). While PPDM is concerned with preserving privacy during the activity of data mining, the aim of PPDP is to produce a sanitised version of a dataset that can be safely shared. Such methods transform sensitive data into a form that preserves data privacy while ensuring that models built from such data have minimal loss of accuracy when compared to models built from the original data. This often results in a trade-off between accuracy and privacy: the more privacy is guaranteed, the less utility the data will have for analytical purposes, resulting in reduced model accuracy. However, it must be emphasised that in sensitive data environments, privacy is paramount. Model accuracy becomes important only if guarantees on the level of privacy can be offered. Optimising the trade-off between privacy and data utility is a driving force behind continuing developments. Furthermore, new attacks against proposed privacy-preservation methods are constantly being devised (Okkalioglu, Okkalioglu, Koc & Polat, 2015).

If each source site of a distributed dataset is owned by a different individual or organisation, then privacy-preservation may be a necessary pre-requisite for sharing data between sites. Furthermore, when attempting to preserve the privacy of a data stream, it must be ensured that:

1. The privacy-preserving protocol or transformation is efficient enough to cope with the velocity of the data stream
2. If privacy is only a concern for a certain duration, then attacks that attempt to breach the privacy of the stream must require such a level of computation so as to render any recovered data obsolete during the time taken to execute the attack.

## 1.4 Research Objective and Contributions

While each of the fields identified above has been extensively researched (Bifet & Kirkby, 2009; Devi, 2014; T. Wang et al., 2018), there has been relatively little research into methods that account for datasets with multiple challenging properties. Adhikari, Adhikari and Pedrycz (2016) considered distributed data mining methods for data streams an open research problem. At the same time, existing approaches for privacy-preserving data stream mining are either limited in scope to a certain class of mining algorithms or are not applicable when combining distributed streams (as reviewed in Chapter 6).

Therefore, the objective of this research was to develop new methods to enable distributed and privacy-preserving data stream mining. The following contributions have been produced by this research:

- A novel Hierarchical Distributed Stream Miner (HDSM) for distributed data stream mining, which:
  - Supports the autonomous creation of local classification models at each source site that produces its own data stream.
  - Augments local classification capability by automatically building classifiers that learn and adapt to changes in relationships between record fragments located at different source sites.
  - Utilises a data transmission protocol that minimises data transmission volume while providing superior accuracy to previously proposed distributed mining approaches, as demonstrated through experimentation.
  - Is able to dynamically trade-off between model accuracy and CPU resource utilisation time, as demonstrated through experimentation.

- Novel data perturbation methods for privacy-preserving data stream mining that leverage random projection, random translation, and additive noise, including:
  - Two novel methods for privacy-preservation based on different forms of additive noise that are designed to operate in a data stream environment.
  - Adaptations of the previously proposed known input-output Maximum A Posteriori (MAP) attack on random projection to account for the addition of random translation and additive noise.
  - An experimental evaluation of the proposed data perturbation methods and their respective attacks, including conducting the first experimental study of the performance of random projection in the context of online classification as far as could be determined from an extensive literature review.
- A description of how the distributed data stream mining and privacy-preservation methods can be combined in practice to achieve a distributed privacy-preserving data stream mining architecture.

## 1.5 Thesis Structure

The remainder of this thesis is divided into two primary sections.

The first section covers the proposed HDSM architecture for distributed data stream mining and is comprised of Chapters 2-5<sup>1</sup>. Chapter 2 discusses previous work in the area of distributed data mining. Chapter 3 then presents the architecture of HDSM for distributed data stream mining. The experimental results that demonstrate the performance and benefits of HDSM are presented next in Chapter 4. To conclude the

---

<sup>1</sup>A paper based on the research presented in Chapters 2-4 has been submitted for publication to Knowledge-Based Systems as “HDSM: A Distributed Data Mining Approach to Classifying Vertically Distributed Data Streams”.

section, Chapter 5 describes possible variations of HDSM to improve transmission efficiency and model accuracy under certain conditions.

The second section covers the proposed data perturbation methods for privacy-preserving data stream mining and is comprised of Chapters 6-8<sup>2</sup>. Chapter 6 reviews the development of privacy-preservation methods and the attacks that have been proposed to counter them. Chapter 7 describes the elements and properties of the proposed composite data perturbation methods, as well as the proposed novel attack types for breaching their privacy. Then, Chapter 8 presents an experimental evaluation of the effectiveness of attacks on different perturbation methods and the accuracy that can be achieved when performing online classification with perturbed datasets.

Finally, Chapter 9 discusses how HDSM and the data perturbation methods can be combined in practice, and Chapter 10 summarises the achievements of the research, including a discussion of its limitations and the opportunities it has created for future work.

---

<sup>2</sup>A paper based on the research presented in Chapters 6-8 is in preparation for submission to Knowledge-Based Systems as “Combining Random Projection and Additive Noise for Privacy-Preserving Data Stream Mining”.

## Chapter 2

# Related Work in Distributed Data Mining

Distributed data mining assumes that the target dataset is distributed across different physical *sites*, and that a global model of the data must be produced with as little data transmission from the individual sites as possible. According to Park, Ayyagari and Kargupta (2001), data is typically distributed in one of two ways: Either each site describes different data records according to the same set of features (known as horizontal or homogeneous distribution), or each site describes the same records, but according to a different set of features at each site (known as vertical or heterogeneous distribution). With vertical distribution, each site essentially has a different view of the same set of records, and it is generally assumed that all sites share at least one “unique key” that can be used to join vertical fragments of records from different sites. Classification of vertically distributed data has been the focus of much prior research, and is also the focus in this research.

## 2.1 Existing Methods for Vertically Distributed Data Mining

Past approaches to classifying vertically distributed streams have included merging classifiers generated at local sites into a combined classifier (R. Chen, Sivakumar & Kargupta, 2001) and parallelising the construction of a centralised classifier (Kourtellis, Morales, Bifet & Murdopo, 2016). These approaches involve constructing a “centralised classifier” at a central site by merging the outputs of local sites, which means that the classification of a record can only occur when all of its features are available at the location of the centralised classifier. This is appropriate for the use cases of producing a model of relationships within the data (R. Chen et al., 2001) or distributing already centralised data for parallel learning (Kourtellis et al., 2016), but not for performing online classification with distributed streams. Y. Liu, Xu and Li (2018) present a method for vertically distributed, online, and semi-supervised classification, but their method is specific to a support vector machine classifier. A more general approach to vertically distributed data mining is to apply ensemble learning techniques to aggregate many local classifications, which can be produced by any kind of classifier. Because ensemble learning only requires the centralisation of local classifications and not data features, it can also be used for data stream classification. The key difference between various ensemble learning techniques is in how they aggregate the local classifications.

Several types of ensemble learning approaches have been used to aggregate results from local classifiers in the distributed (though not necessarily streaming) context. Aggregation can be based on local classification confidence, such as selecting the classification with the maximum confidence (Park et al., 2001), selecting a classification based on other order statistics (Tumer & Ghosh, 2000), or treating local classification confidences as probabilities that can be used to approximate a global posterior probability (Basak & Kothari, 2004). An approach used by Skillicorn and McConnell (2008)



is to select the classification that receives the majority vote of local sites, optionally weighting votes by test accuracy (or in a stream mining context, the recently observed accuracy could potentially be used instead). Alternatively, a “stacked” classifier can be trained to predict the true classification based on local classifications. Parker, Mustafa and Khan (2012) use a hierarchy of stacked classifiers to classify vertically distributed data streams with the ability to learn new classes as they appear over time. However, not all methods of ensemble aggregation are suitable for fully online stream learning. For example, those used by Recamonde-Mendoza and Bazzan (2016) require batches of records to be ranked according to local confidence before applying social choice functions to merge sets of local rankings.

More recent approaches to classifying vertically distributed, non-streaming data either produce a “centralised classifier” (T. Li, Li, Liu, Li & Jia, 2018; Y. Li et al., 2017; Moghadam & Ravanmehr, 2018; Omer, Gao & Mustafa, 2017) or are privacy-preserving methods that require multiple rounds of communication between sites (Khodaparast, Sheikhalishahi, Haghighi & Martinelli, 2018). Therefore, these methods are not suitable for a data stream environment. The most relevant approaches for comparison are ensembles of local classifiers with aggregation based on maximum confidence (Park et al., 2001), voting (Skillicorn & McConnell, 2008), and stacking (Parker et al., 2012). The method of Park et al. (2001) that centralises a small subset of the data is also suitable for comparison, as it forms the basis for the approach of HDSM (as described in the following section).

## 2.2 Foundational Distributed Data Mining Approach

The previous work of most relevance to this research is the work of Park et al. (2001), which forms the foundation for the architecture of HDSM presented in the next chapter.

In their approach, each local or “primary” site (*p-site*) constructs a classifier

from its data and identifies a subset of “trouble records” that were classified with a confidence value below a certain “confidence threshold”. If a particular record appears in the trouble records of all primary sites, then all fragments of the record are transmitted to a central “trouble site” (*t-site*) to be used to train an additional classifier. The trouble site’s classifier captures relationships (hereafter referred to as cross-terms) that exist within the data: patterns that can only be discovered by viewing the data from the combined perspective of more than one site. To classify new records, the models at each primary site and the trouble site are applied, and the classification results are transmitted to a central “aggregator”. The classification result that achieved the highest confidence is then selected as the final classification. Experimentation showed that as the confidence threshold was increased to allow more data to be sent to the trouble site, the overall classification accuracy also increased as a result.

There are some limitations of the approach of Park et al. that are addressed by the HDSDM architecture proposed in this research. Firstly, having only a single, central, trouble site to collect records from all primary sites may not be necessary if cross-terms exist between the features of only a subset of primary sites. While the problem of data stream mining was not specifically addressed by Park et al., their approach is generally amenable for use in classifying data streams. However, it is not equipped to control data transmission volume in a dynamic stream environment, as is demonstrated in the following chapters.

## **Chapter 3**

# **Proposed HDSM Architecture for Distributed Stream Mining**

This chapter describes the proposed HDSM architecture for performing online classification on multiple vertically-distributed data streams without joining all record fragments at a central location. HDSM is based on the previously described approach of Park et al. (2001), but contains three main extensions to make it applicable to a distributed data stream mining context, as detailed below.

### **3.1 Trouble Site Hierarchies**

A key limitation of Park et al.'s approach is that there is only one trouble site. This may result in a large amount of unnecessary data transmission when cross-terms only exist between the features across some subset of primary sites. Therefore, HDSM allows for multiple trouble sites, each of which may receive trouble records from only a subset of two or more primary or trouble sites (a trouble site that considers only a small subset of features may need to escalate to a higher-level trouble site that considers more features). Figure 3.1 presents a physical view of the HDSM architecture, which is composed of

multiple primary sites and trouble sites, and a single classification aggregator. The behaviours of each component are described in Algorithms 1, 2, and 3.

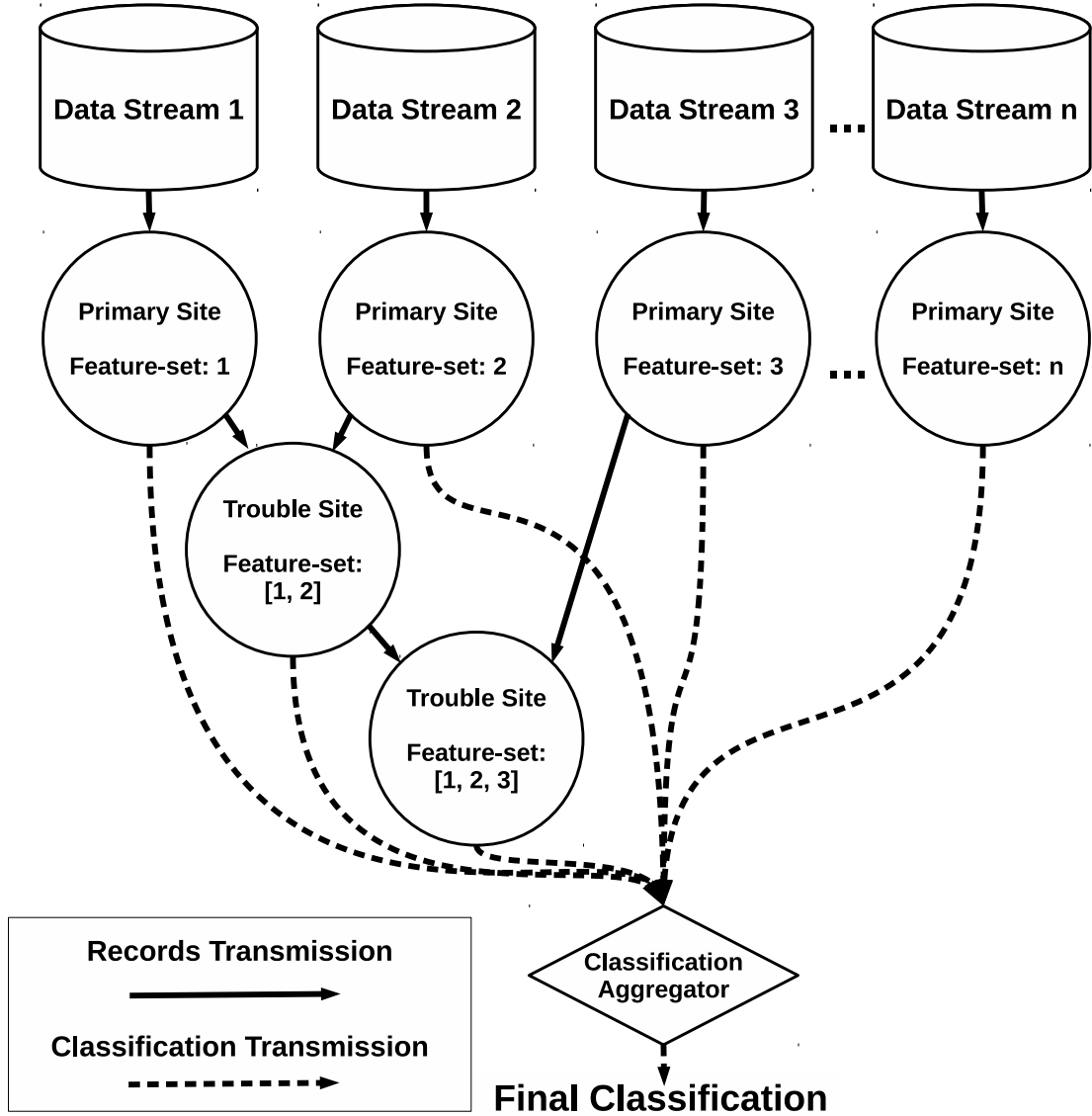


Figure 3.1: Physical view of the HDSM architecture.

Each primary site receives input from a different heterogeneous data stream. The feature set of each stream may contain a different number of features, and there may be overlap between feature sets. It is assumed that each feature set will contain a common “key” feature that is not used for classification, but allows record fragments and classifications to be merged at trouble sites and the classification aggregator respectively.

```

1 Procedure processRecord (site, record) :
2   Classify record at site to get the classification;
3   foreach t-site that site may forward to do
4     if classification confidence < threshold from site to t-site then
5       forwardTroubleRecord (t-site, record) ;
6     end
7   end
8   notifyAggregator (site, record-key, classification,
                      t-site-forwarding) ;

```

**Algorithm 1:** Procedure for primary site behaviour.

```

1 Procedure forwardTroubleRecord (t-site, record-fragment) :
2   if t-site has received a record-fragment from all source sites then
3     processRecord (t-site, merged-record) ;
4   else
5     add record-fragment to a size-limited buffer at t-site;
6   end

```

**Algorithm 2:** Procedure for trouble site behaviour.

Because each primary site only needs to process data from a single stream, it is logical for each primary site to be physically located at the source of its data stream. Procedure `processRecord` (Algorithm 1) shows how each record that arrives at a primary site will be classified by a local stream classifier. It is possible to use any stream classification algorithm at each site, or even different algorithms at different sites, as long as it is possible to produce a confidence value on a scale consistent with all other sites for the purpose of determining the classification with the highest confidence at the aggregator.

Each trouble site in the network is connected to two or more “source sites” (either

```

1 Procedure notifyAggregator (site, record-key, classification,
                                t-site-forwarding) :
2   log classification as received from site for record-key;
3   foreach t-site in t-site-forwarding do
4     if site forwarded record-key to t-site AND record-key has not been
       logged as unexpected from t-site then
5       log that a result for record-key may be expected from t-site;
6     else
7       log that a result for record-key is unexpected from t-site;
8     end
9   end
10  if all p-sites and expected t-sites have provided a classification then
11    aggregate all classifications for record-key into a final-classification;
12  end

```

**Algorithm 3:** Procedure for aggregator site behaviour.

primary sites or other trouble sites) that forward trouble records to the trouble site, as shown in Figure 3.1. A trouble site can be said to be of a certain “order”, which is one higher than the order of any of its source sites (where primary sites are of order 0). Procedure `forwardTroubleRecord` (Algorithm 2) shows that if every source site for a given trouble site “agrees” a particular record is “trouble”, then that trouble site will apply its own stream classifier to a merged record containing all of the features provided by its source sites. If any source site does not forward its fragment of a record as trouble, then the trouble site does not need to process that record, as the classification confidence was high enough at that source site to serve as a classification for the record without additional processing. To prevent a trouble site from waiting indefinitely for a record fragment that will never be sent, a fixed-size buffer may be used to store

incoming record fragments. If all of the fragments arrive for a particular record, then it can be removed from the buffer and processed. However, if some fragments of a record never arrive, then it will eventually be removed from the end of the buffer to make space for new incoming record fragments. Furthermore, if the key values for each record are known to arrive at primary sites in a sequential order (such as a timestamp identifier), then a trouble site may choose to stop waiting for the fragments of records from a particular source site if it receives a record from it with a later sequential key.

Every site (primary or trouble) within the network must forward its classification and confidence for each record it processes to the “classification aggregator”, which provides the final classification for all records. The simplest aggregation method, used by Park et al., is to select the classification with the highest confidence. Procedure `notifyAggregator` (Algorithm 3) shows how the aggregator keeps track of which trouble sites a classification may be “expected” from (because at least one of its source sites declared that it was forwarding the record to that trouble site) and which trouble sites a classification is definitely “unexpected” from (because at least one of its source sites declared that it was not forwarding the record to that trouble site). In this way, the aggregator is able to produce a final classification as soon as all primary and trouble sites that will process the record have provided their classifications.

It is assumed that each site will eventually receive the true classification label of each record it processes, so that it may train its stream classifier with them. A trouble site will only be trained on a particular record if there is agreement between its source sites; it cannot be trained if it does not have the combination of features from all of its source sites.

### 3.2 Protocol for Regulating Data Transmission

Park et al. used a static confidence threshold to determine which records to forward as trouble records, which may result in vastly different volumes of data being forwarded depending on how confident a site's classifier is. This is acceptable when dealing with non-streaming data, as the static threshold can be selected after evaluating the distribution of confidence values achieved at each primary site. However, in a data stream mining context, the confidence of site classifiers may change over time, which could lead to overloading trouble sites with more data than they are capable of receiving. HDSM addresses this problem: each source site forwards a fixed proportion of its records below a "quantile threshold" that is determined by the volume of data the trouble site can process. By finding the rank of a confidence value within a sliding window of recent local classification confidences, a site can determine whether that confidence value falls below the quantile threshold.

A number of different factors must be taken into account when determining the quantile threshold. Firstly, the combined volume of data forwarded from all source sites should be approximately equal to the volume of data that can physically be processed by the trouble site, irrespective of the number of source sites or the size of the feature set each provides. Secondly, the proportion of records forwarded from a particular trouble site should not be affected by the level of "agreement" at that site (i.e. the proportion of received record fragments that were actually processed). Finally, the number of trouble records sent by each source site should be approximately equal, even though record fragments from different source sites may be of different sizes. Because a trouble record is only processed at a trouble site if fragments are received from all source sites, it is not worthwhile sending more trouble records from one source site than from another.

Before an expression for the quantile threshold can be defined, a method for quantifying the agreement at a particular trouble site must be established, which should



represent the probability that all source sites agree any received record is “trouble”. To measure the agreement, a trouble site must maintain a sliding “agreement window” with entries for each unique record received by the site. Each entry records the number of source sites that forwarded their fragment of that unique record. Consider a scenario involving three records (A, B and C) and four source sites sending record fragments to a single trouble site. The first three source sites send fragments of records A and B in sequence, while the fourth sends fragments of records A and C in sequence. This transmission will result in three window entries:  $[4, 3, 1]$  (record A was forwarded by all 4 source sites, B by 3 source sites, and C by only a single source site). Out of the record fragments sent from each source site, there was agreement on only one record (A) which accounts for 4 matches out of a total transmission of 8 fragments, and so the level of agreement is 0.5, as illustrated by Equation (3.1).

$$a_t = s \left( \frac{\sum_{i=1}^{|w|} I(w_i = s)}{\sum_{i=1}^{|w|} w_i} \right) \quad (3.1)$$

Equation (3.1) defines the level of agreement  $a_t$  at a trouble site  $t$ ;  $s$  is the number of source sites that feed data to the trouble site;  $I(k)$  is the identity function that returns 1 if  $k$  is true and 0 otherwise, and  $w_i$  is the  $i^{th}$  entry in the “agreement window”. Therefore,  $I(w_i = s)$  equals 1 if there is agreement for the  $i^{th}$  entry, and 0 otherwise. Applying Equation (3.1) to the example above results in:  $a_t = 4 \times \frac{1}{8} = 0.5$ .

To quantify the diminishing number of records transmitted between successive trouble sites (as more features are processed, fewer records can be processed), the concept of a reduction coefficient needs to be defined. This reduction coefficient represents the probability that a site will both receive and process any given record. Equation (3.2) defines an expression for computing the reduction coefficient ( $r_s$ ) at site  $s$ . For primary sites, which process all records entering the system, the  $r_s$  coefficient is

always equal to 1. For a trouble site,  $r_s$  is the product of the trouble site's agreement ( $a_s$ ) with a scaling factor ( $k_s$ ) that regulates the maximum number of records that can be processed at a given trouble site. The scaling factor takes into account the processing capacity of a trouble site; a scaling factor of  $k_s$  indicates that the trouble site is capable of processing  $k_s$  times the combined volume of data available at its source sites. The scaling factor is expressed in terms of a user-configured trouble factor ( $t_s$ ), which allows the user to control the volume of data that will be processed at a trouble site independently of the sizes of its source sites' feature sets. A typical trouble factor would be the maximum feature set size of any primary site, which indicates that the trouble site is capable of receiving the same volume of data as any primary site.

$$l_s = \sum_{i=1}^{m_s} l_s^i \quad k_s = \frac{t_s}{l_s}$$

$$r_s = \begin{cases} 1 & \text{if } s \text{ is a primary site} \\ a_s \times \min(1, k_s) & \text{otherwise} \end{cases} \quad (3.2)$$

where  $l_s$  is the size of the feature set at site  $s$ ,  $m_s$  is the number of features processed at site  $s$ , and  $l_s^i$  denotes the size of feature  $i$  at site  $s$ .

Having defined agreement and the reduction coefficient, Equation (3.3) can now be used to compute the quantile threshold  $(q_{s \Rightarrow d})^1$  that determines the fraction of records that will be forwarded from a given source site  $s$  to a destination trouble site  $d$ .

$$q_{s \Rightarrow d} = \min\left(1, \frac{k_d}{r_s}\right) \quad (3.3)$$

Equation (3.3) shows that the quantile threshold is essentially the quotient of the scaling factor that can be tolerated at the destination site  $d$  divided by the reduction factor achieved at the source site  $s$ .

Note that when the source site  $s$  happens to be a trouble site, the quantile threshold

---

<sup>1</sup>Proofs and examples related to this quantile threshold and its components are given in Appendix A.

becomes inversely proportional to the agreement at  $s$  (through the definition of  $r_s$ ). This means that a trouble site with low agreement may send nearly all of the records it processes to a subsequent trouble site. Because of this and the fact that a trouble site with low agreement will make use of far less data than it receives; trouble sites with low agreement should be avoided in practice.

This quantile-based approach to controlling data transmission ensures that a trouble site receives a proportion of the dataset that is within its processing capacity, and no more. This control is exercised by the trouble factor parameter, and solves the issue of flooding a trouble site with too many records when the confidences of its source sites' classifiers drop.

### 3.3 Learning Trouble Site Hierarchies

By supporting multiple trouble sites, HDSM allows many possible trouble site hierarchies for a given set of primary sites. Some trouble site hierarchies may be partially or entirely determined by physical limitations, such as introducing trouble sites for groups of physically neighbouring primary sites. In other cases, where there is no practical basis for a particular configuration, a hierarchy must be selected that achieves the best classification accuracy gains with the least data transmission. The optimal hierarchy will contain trouble sites that capture cross-terms between the features of different primary sites. Furthermore, as the underlying patterns and cross-terms in the data streams change over time, trouble sites may need to be added and removed from the hierarchy to optimise performance. The following section presents an online approach to learning and adapting the trouble site hierarchy.

To begin learning the trouble site hierarchy, the system can be initialised with a minimal structure consisting solely of the primary sites and the required classification aggregator, with no trouble sites. Figure 3.2 demonstrates three sets of “monitors” that

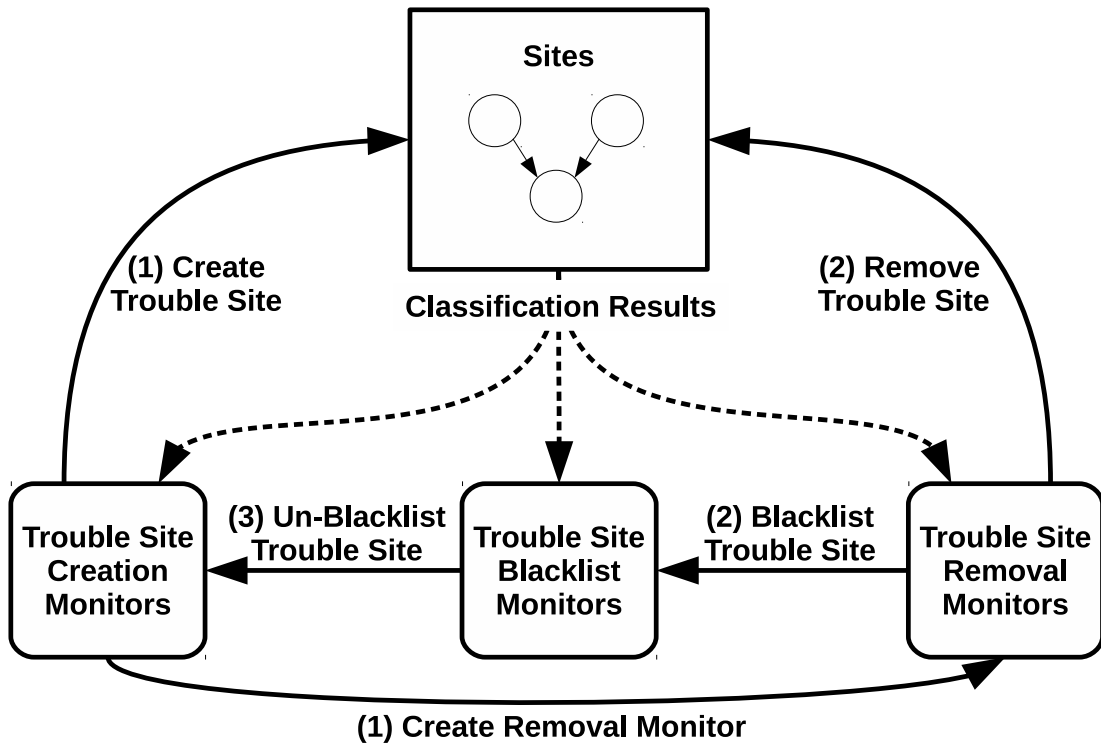


Figure 3.2: Continuous process for creating, removing, and un-blacklisting HDSM trouble sites.

will then add or remove trouble sites based on the classification results of the system. Firstly, the “trouble site creation monitors” watch for possible sets of source sites that achieve agreement above a threshold for a sufficient period of time, which indicates a new trouble site should be created. Step 1 shows the creation of a trouble site along with a “trouble site removal monitor” to evaluate both how often it is used for the final classification and how accurate it is when used. The trouble site will be removed (step 2) if its utilisation or accuracy drops below a threshold for a sufficient period of time, and it will also be added to a “blacklist” of trouble sites that should not be recreated until they are “un-blacklisted”. The “trouble site blacklist monitors” then watch for concept drifts in the data streams that indicate a trouble site should be un-blacklisted (step 3), which will allow a creation monitor to recreate them if and when appropriate. To make these evaluations, each type of monitor must receive for each record: the

final classification result of the system; the identifier of the site that provided the final classification; the individual classifications and confidences of each site; and finally, a representation of the current structure of the hierarchy. The remainder of this section describes the behaviour of each type of monitor in more detail.

A creation monitor will always exist for each combination of existing sites that could be used as the source sites for a new trouble site, provided that trouble site is not currently blacklisted. In the current implementation of HDSM, the set of potential source site combinations is limited to prevent an explosion of monitors when there are many potential trouble sites. Specifically, potential source sites must not already be acting as a source site to a different trouble site, which reduces redundant transmission of the same features to multiple trouble sites. Additionally, only pairs of source sites are allowed, as agreement will generally drop with a greater number of source sites, resulting in more fruitless data transmission. Each creation monitor uses the same window-based agreement monitoring method described in the previous section, but only considers records that were classified incorrectly by the entire HDSM classifier, as there is no need to improve the classifications of records that are already classified correctly. If the level of agreement in the window rises above a threshold ( $T_a$ ) for a configured “threshold-time-period” (a number of records), then the trouble site is created.

A trouble site must be removed when it ceases to provide a meaningful benefit to the accuracy of the overall classifier, typically when there is some concept drift in the data streams. It is also possible that a trouble site never provides a meaningful benefit, as creation monitors must decide whether to create trouble sites based on limited information that may not reflect the actual benefit of the trouble site to the accuracy of the system. It is also important that a trouble site being used as a source site for another trouble site is not monitored for removal, as the subsequent trouble site may be providing a benefit even if the intermediary trouble site is not. A removal monitor will use sliding windows to monitor both how often the trouble site is used as the

final classification of the overall classifier, and the accuracy it achieves when it is. If either the utilisation or accuracy drop below their respective thresholds ( $T_u$  and  $T_{acc}$ ) for a configured “threshold-time-period” (a number of records), then the trouble site is removed and blacklisted.

For each blacklisted trouble site, a blacklist monitor will be created to watch for drift in the accuracy of the classifier at each source site. If drift is detected at any source site, then a cross-term may have appeared in the dataset that the trouble site could capture. As a result of this, the trouble site may now meet the criteria for creation once again. The trouble site is therefore removed from the blacklist, which allows a new creation monitor to be created for it. For experimental evaluation, ADWIN (Bifet & Gavalda, 2007) was used for drift detection in blacklist monitors.

### 3.3.1 Monitor Thresholds

The creation and removal monitors use four thresholds to determine when to trigger their respective actions. Two of these must be configured through user-defined parameters: both monitor types’ threshold-time-periods (set to a sufficient period of time to judge the permanence of a pattern) and the removal monitor’s utilisation threshold (set based on how frequently a site must be used for it to justify its real-world resource cost). Conversely, the removal monitor’s accuracy threshold and creation monitor’s agreement threshold can be set dynamically based on stream behaviour, as shown in Equations (3.5) and (3.6) and described below.

A trouble site’s accuracy threshold ( $T_{acc}$ ) must require a significant improvement over the accuracy that would be achieved without that site. Specifically, accuracy should be compared with the accuracy that could be achieved by aggregating the classifications of only lower-order sites ( $acc_{o-1}$ ), which prevents simultaneously monitored sites of the same order being removed for mutually redundant accuracy improvement. Therefore,

the accuracy threshold is set to this lower-order accuracy plus an additional term that demands a significant improvement ( $\varepsilon$ ).

Similarly, the agreement threshold ( $T_a$ ) must require a significant improvement over the level of agreement that would be expected by chance ( $\tilde{a}$ ).  $\tilde{a}$  can be calculated as the expected proportion of overlap between the trouble record sets transmitted by source sites ( $s$ ) if trouble records were selected randomly, which is the product of the proportion of records ( $p$ ) each source site transmits. As with the accuracy threshold, the required improvement is represented by the  $\varepsilon$  term, but it is smoothed by an additional term ( $\Theta$ ). This smoothing term diminishes the required improvement as expected agreement approaches 100%, where exceeding the expected agreement is less likely. The degree of smoothing is determined by a manually set smoothing-factor ( $\theta$ ).

$$\tilde{a} = \prod_{i=1}^s p_i$$

$$\Theta = 1 - \tilde{a}^\theta \quad (3.4)$$

$$\varepsilon = \sqrt{\frac{\ln(1/\delta)}{2n}} \cdot R$$

$$T_{acc} = \min(1, acc_{o-1} + \varepsilon) \quad (3.5)$$

$$T_a = \min(1, \tilde{a} + \Theta\varepsilon) \quad (3.6)$$

For both dynamically set thresholds, the term to require significant improvement ( $\varepsilon$ ) is provided by the Hoeffding bound (Witten, Frank, Hall & Pal, 2016). When computing the Hoeffding bound, the range ( $R$ ) is 1 for both accuracy and agreement (which vary from 0-100%). The number of observations  $n$  is the current size of the window monitoring accuracy or agreement plus the threshold-time-period, as this represents the size of the entire sample the improvement will be evaluated over. The  $\delta$  parameter of the Hoeffding bound determines the confidence level  $(1 - \delta)$  for  $\varepsilon$ , and

hence  $\delta$  is set to a small value, such as  $10^{-3}$ . The influence of the Hoeffding bound is also restricted to never increase either threshold above the maximum possible accuracy and agreement value of 100%, as shown in Equations (3.5) and (3.6).

### 3.4 Alternative Aggregation Methods

Apart from selecting the classification with the highest confidence, it is also possible to use other ensemble learning methods at the aggregator. However, because other methods do not necessarily select a single site's classification as the "winner", they may not be compatible with the utilisation monitoring of trouble sites by removal monitors. For this reason, aggregation with other methods should be performed in two stages: first using whatever alternative method was selected to aggregate primary site classifications, and then aggregating the produced classification with those from trouble sites using the "maximum-confidence" method. This also means that the first phase of aggregation can be performed while trouble sites are still processing the record.

Two alternative aggregation methods considered in the following chapter's experimental evaluation are voting and stacking. Voting is based on the approach of Skillicorn and McConnell (2008), where the class with the majority of votes from primary sites is selected, and the confidence is the mean confidence of all primary sites. Stacking is based on the approach of Parker et al. (2012). Primary site classifications are used as the input records to another stream classifier (using the same or a different algorithm to that used at primary sites and trouble sites), which produces a new classification and confidence.



## Chapter 4

# Experimental Evaluation of HDSM

HDSM was implemented for experimental evaluation using Clojure and machine-learning algorithms from the MOA data stream mining framework (Bifet, Holmes, Kirkby & Pfahringer, 2010). The source-code (including data preparation and experimentation) has been made available in a GitHub repository<sup>1</sup>. The experiments presented below illustrate HDSM’s ability to improve classification accuracy with limited transmission of data features from primary sites to trouble sites, including: a demonstration of the behaviour of HDSM’s dynamic trouble-site hierarchy, a comparison of HDSM’s performance with other distributed stream mining approaches on a variety of real-world and synthetic stream datasets, and a demonstration of HDSM’s potential to dynamically trade off between accuracy and response time.

All experiments were performed with OpenJDK 1.8.0 on a 64-bit Ubuntu 16.04 installation running on a 4x2.60GHz Intel Core i5 CPU with 8GB of memory. As all of the data features within each experiment were of the same data-type, they were treated as being the same size ( $l_s^i = 1$ ), meaning that a trouble factor equal to the number of features at a primary site would represent a trouble site capable of receiving as much data as a primary site. The list below summarises configuration parameters that were

---

<sup>1</sup><https://github.com/ben-denham/hdsm>

kept constant throughout experimentation (except where noted otherwise):

- Size-limit for all windows at sites and monitors: 1000 latest records
- Threshold-time-period for creation and removal monitors: 500 records
- $\delta$  of Hoeffding bound in agreement and accuracy thresholds:  $10^{-3}$
- Smoothing factor for agreement threshold ( $\theta$ ): 5
- Site utilisation threshold for removal monitors ( $T_u$ ): 5%

## 4.1 Performance Evaluation Metrics

The following section describes the metrics that were used to compare the accuracy, volume of data transmission, and resource time for different stream mining methods. In order to evaluate the typical running state of each method, the first 1000 records of each experiment were considered a “warm-up” time and ignored in all metric calculations.

Accuracy was evaluated prequentially: each record was classified and then immediately used for training. Each site that processed a record was only trained with the features that were transmitted to that site.

Record feature transmission is reported in terms of TDTV (Total Data Transmission Volume between all sites) and MDTV (Maximum Data Transmission Volume to any single trouble site, averaged across windows of 100 records). Both transmission metrics are reported as proportions of the total data volume contained in the dataset. TDTV may exceed 100% in cases when record fragments are transmitted through multiple layers of trouble sites. Transmission of site classifications to the aggregator is not reported, as transmission of class labels and confidence values from primary sites is the same for all evaluated methods while being a much less significant cost in comparison to feature

transmission. Because of this, all experiments that did not result in transmission of data features to trouble sites have 0% TDTV and MDTV.

Resource time is reported in terms of MTCP (Mean Time on Critical Path) and MTBC (Mean Time Between Completions). MTCP is the cumulative CPU time across the longest-running sequence of classifiers used to process a record, averaged over all records. MTBC is the mean difference between the completion times of successive records, given each classifier can begin processing a record after its source sites have processed it and after the site has finished processing the previous record. While MTCP measures the classification latency (response time) for a single record on average, MTBC measures the throughput rate for stream-processing. Resource time includes the CPU processing time of classifiers at primary and trouble sites and stacked aggregators; maximum confidence aggregation and voting overheads were excluded because these costs are less significant and the underlying operations can be parallelised in practice. To account for JVM warm-up, each set of time-evaluated experiments was performed twice in sequence, and timing was recorded from the second run. Both MTCP and MTBC are reported in nanoseconds.

## 4.2 Demonstration of Dynamic Trouble Site Hierarchy

To demonstrate the behaviour of HDSM's dynamic trouble site hierarchy, it was tested with a synthetic dataset that abruptly drifted between different underlying cross-terms. A dataset of 30,000 records was generated with 6 binary features and a binary class. Equation (4.1) defines the rule used to generate a cross-term dependent class value from random binary features.

$$\begin{aligned}
 class = & (f_1^1 \wedge f_1^2 \wedge \dots \wedge f_1^k) \vee \dots \\
 & \vee (f_m^1 \wedge f_m^2 \wedge \dots \wedge f_m^k)
 \end{aligned}
 \tag{4.1}$$

where  $k$  is the number of features in each cross-term,  $m$  is the number of cross terms, and  $f_j^i$  is the  $i$ -th feature involved in the  $j$ -th cross-term.

The dataset always contained two cross-terms ( $m = 2$ ) between two features ( $k = 2$ ): the first and last 10,000 records had cross-terms between feature pairs  $[0, 1]$  and  $[2, 3]$ , while the middle batch of 10,000 records had cross-terms between pairs  $[0, 1]$  and  $[4, 5]$ . The features were distributed across 6 primary sites, with each site receiving a single feature, thus representing a high degree of data distribution. HDSM was configured with a trouble factor of 1 so that each trouble site would receive approximately the same volume of data as any primary site (as there is one feature per primary site).

HDSM was compared with a distributed ensemble without trouble sites where final classifications were based solely on the maximum-confidence aggregation of primary site classifications. To allow fast adaptation to abrupt concept drift, Hoeffding trees (Hulten, Spencer & Domingos, 2001) were used as classifiers at primary and trouble sites.

Table 4.1 reports the utilisation of each site in the classification process, the contribution made by each site to overall accuracy, and the data volume transmitted to each site. Overall, Table 4.1 shows clearly that HDSM significantly outperformed the distributed ensemble with an increase in accuracy of 27.27% with data transmission equivalent to only 37.29% of the dataset's total data volume. Trouble sites are represented as pairs of source sites, so it can be seen that trouble sites were created to capture the cross-term pairs ( $[0, 1]$ ,  $[3, 2]$ , and  $[4, 5]$ ), and the accuracy contributions of those trouble sites were proportional to the number of records that each cross-term was present for. A higher-order trouble site for  $[[3, 2], 5]$  was also created, but was removed due to low utilisation shortly after its creation. It can also be seen that the primary sites are used for classification less often in the configuration with trouble sites, but their accuracy contribution drops less significantly when compared to their drop in utilisation. This indicates that the trouble sites are successfully processing records that the primary sites

Table 4.1: Effects of HDSM’s trouble site hierarchy on performance with a concept drifting data stream.

Site	Without Trouble Sites			With Trouble Sites		
	Utilisation	Accuracy Contrib.	TDTV	Utilisation	Accuracy Contrib.	TDTV
0	20.69%	13.10%	N/A	12.41%	10.80%	N/A
1	37.75%	26.76%	N/A	24.96%	21.78%	N/A
2	16.80%	8.92%	N/A	8.62%	8.04%	N/A
3	13.21%	6.70%	N/A	5.83%	5.03%	N/A
4	7.74%	4.91%	N/A	4.60%	4.30%	N/A
5	3.80%	1.68%	N/A	1.87%	1.64%	N/A
[0, 1]				21.43%	21.35%	16.17%
[3, 2]				14.17%	12.11%	12.10%
[4, 5]				6.11%	4.29%	8.69%
[[3, 2], 5]				0.00%	0.00%	0.33%
Totals	100.00%	62.08%	0.00%	100.00%	89.35%	37.29%

found difficult to classify.

Figure 4.1 shows how trouble sites were added and removed over the course of the HDSM experiment, and how this affected the volume of data transmitted and the accuracy improvement over the distributed ensemble. Because the system begins with no trouble sites, there is initially no data transmission and no difference in accuracy between HDSM and the distributed ensemble. However, after processing approximately 1,300 records, three candidate trouble sites exceed the required agreement threshold and are therefore added to the system. Trouble sites [0, 1] and [3, 2] capture the cross-terms in the dataset and boost system accuracy at the expense of increased data transmission. The trouble site [4, 5] is quickly removed (and therefore blacklisted) because that cross-term is not yet present, and the site’s classifications are not confident enough to be used for the final classification. After the concept drift point at 10,000 records, the HDSM accuracy drops closer to the distributed ensemble’s accuracy, and there is renewed activity in altering the trouble site hierarchy. Eventually, the trouble site

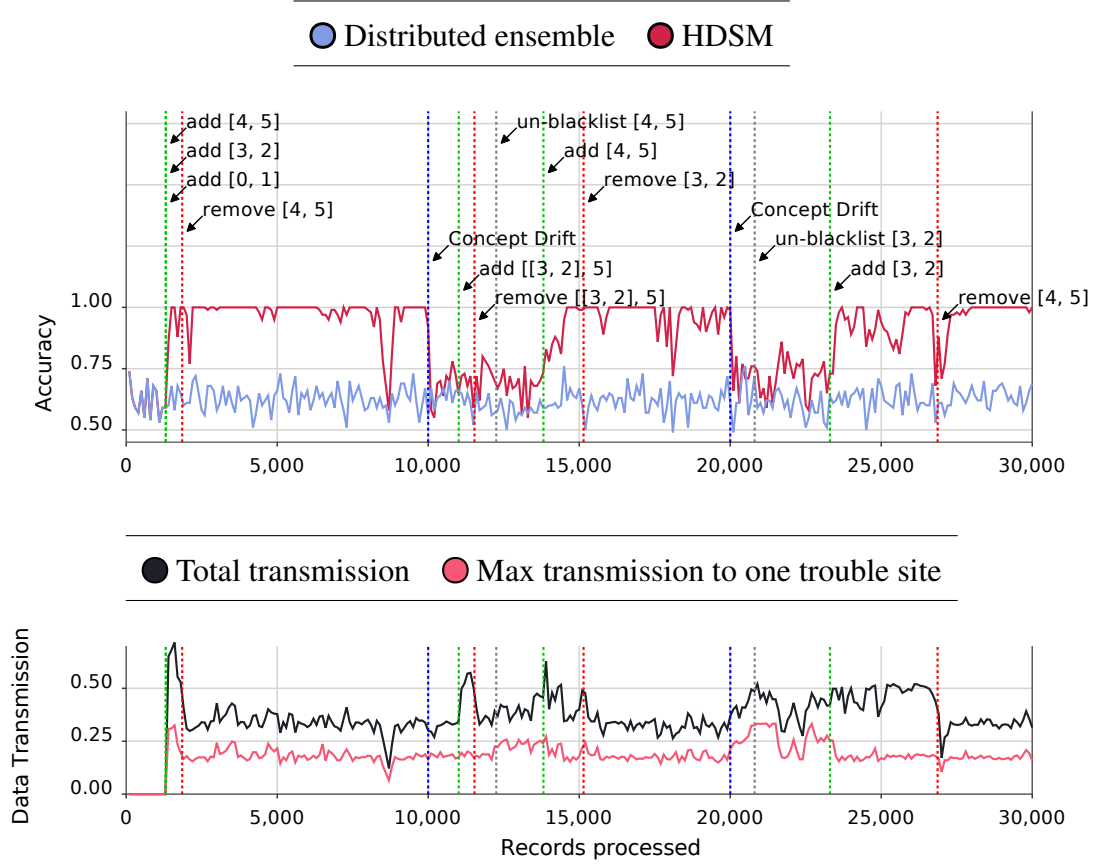


Figure 4.1: Timeline of accuracy and data transmission for the synthetic cross-term dataset with concept drift points.

[4, 5] is un-blacklisted (triggered by the concept drift) and the trouble site is recreated, improving accuracy by capturing that introduced cross-term. The trouble site [3, 2] is removed because its utilisation has dropped due to the fact that cross-term is no longer present in the data stream. Finally, after the second drift point at 20,000 records, trouble site [3, 2] is re-created and trouble site [4, 5] is removed, reflecting the shift back to the original cross-terms. Trouble site [0, 1] is never removed because that cross-term is always present and the site is consistently used to improve classification accuracy. While the level of total transmission fluctuates depending on the number of trouble sites, the maximum transmission to any trouble site remains relatively stable near the proportion determined by the trouble factor (in this case,  $\sim 16.67\%$  of the dataset).

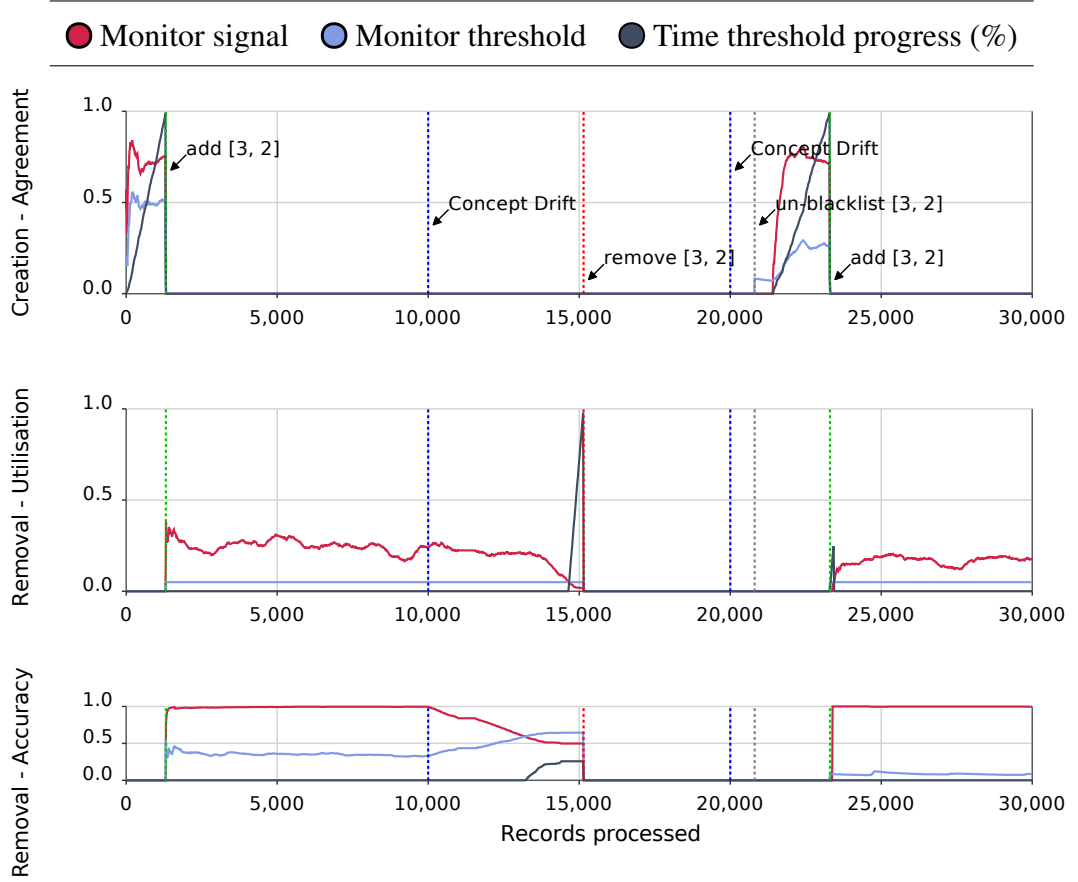


Figure 4.2: Timeline of trouble site  $[3, 2]$  creation (agreement) and removal (utilisation and accuracy) monitors for the synthetic cross-term dataset with concept drift points.

Figure 4.2 demonstrates the behaviour of the monitors for trouble site  $[3, 2]$  over the course of the experiment. When the measured agreement exceeds its threshold, progress against the creation-time threshold increases until it reaches 100%, triggering the creation of the trouble site. Because agreement monitors are only updated based on incorrectly classified records, progress against the time threshold is not perfectly linear. The creation monitor is inactive while the trouble site exists and while it is still blacklisted after removal. The removal monitors become active after the trouble site is created, and they observe a high degree of accuracy and utilisation while the cross-term between the two source sites is present in the dataset. After the concept drift at 10,000 records, the accuracy and utilisation decrease until they drop below their respective

thresholds. The time threshold for utilisation is exceeded first, triggering the removal of the site. It can also be seen that while the utilisation threshold for removal remains constant over time, the agreement threshold varies as site communication affects the expected agreement, and the accuracy threshold varies with the combined accuracy of lower-order sites for records where the monitored site is used for the final classification. Finally, a short period can be observed around the 11,000 record point where the monitor values for utilisation and accuracy do not change. This is because another trouble site ( $[[3, 2], 5]$ ) is created that depends on  $[3, 2]$ , suspending its removal monitors until site  $[[3, 2], 5]$  is removed.

### 4.3 HDSM Performance Evaluation and Comparison

The following experiments compare the performance of HDSM with four previously proposed distributed data stream mining algorithms. Three of these algorithms can be referred to as distributed ensemble methods, as they use local classifiers at primary sites and transmit local classifications to a central site for aggregation. The three methods differ in the rule that is used to aggregate classifications. DMaxConf selects the local classification with the highest confidence, which is the same aggregation rule used by Park et al. (2001). DVote selects the classification with the majority vote, as used by Skillicorn and McConnell (2008). DStack trains a stacked classifier on the local classifications, which is the basis of the approach proposed by Parker et al. (2012). HDSM was also tested using all three of these aggregation rules to aggregate primary site classifications, as described in Section 3.4. The fourth algorithm compared to HDSM is the original approach of Park et al. with a minor modification to make it suitable for streaming data: the single, fixed trouble site that uses all primary sites as source sites was configured with the same dynamic confidence quantile threshold as HDSM. Without this modification, a static threshold would need to have been set, which



would not have adequately controlled the trouble record transmission rate.

For these experiments, Naive Bayes classifiers were used at primary sites because the algorithm is lightweight enough to run at sensor network edge nodes, and because it almost always produces varying confidence values which distinguish trouble records (unlike Hoeffding trees, which return constant confidence values when there is not enough information to perform splits). Adaptive Random Forest (ARF) classifiers (Gomes et al., 2017) were used at trouble sites and for stacking, as they were found to effectively learn from cross-terms.

A combination of real-world and synthetic stream datasets were used in these experiments. The real-world datasets were selected because they represent data streams that could be vertically-distributed in practice and because they exhibit cross-terms between distributed feature-sets. The seven real-world datasets were EEG Eye State (EEG) (T. Wang, Guan, Man & Ting, 2014), Gas Sensor Array Drift (GAS) (Vergara et al., 2012), HIGGS (HIG) (Baldi, Sadowski & Whiteson, 2014), NASA FLTz (FLT) (Oza, 2011), Occupancy Detection (OCC) (Candanedo & Feldheim, 2016), Sensorless Drive Diagnosis (SDD) (Paschke et al., 2013), and Wall-Following Robot Navigation (WFR) (Freire, Barreto, Veloso & Varela, 2009). Additionally, four synthetic stream datasets that had previously been used in evaluating the Adaptive Random Forest classifier (Gomes et al., 2017) were selected to evaluate performance under known concept drift conditions, which were: incremental, abrupt, and gradual drift. These four datasets were based on the Radial Basis Function generator (RBF; Bifet and Kirkby (2009)) and SEA generator (Street & Kim, 2001). Table 4.2 lists the properties of these datasets, which represent a range of feature counts, degrees of distribution, and class counts.

Some datasets required shuffling for meaningful evaluation, as they consisted of long periods where only a single class value was present. Because the SDD dataset is composed of several streams (one for each class), the streams were interleaved to

Table 4.2: Properties of datasets used for distributed stream mining performance evaluation.

Dataset	Feature Count	Features per Site	P-Site Count	Class Count	Record Count	Stream?
EEG	14	1 (site/sensor)	14	2	14,980	Shuffled
GAS	128	8 (site/sensor)	16	6	13,910	Stream
HIG	19	3-4 (see text)	5	2	10,000	Stream
FLT	20	1-3 (see text)	9	2	25,034	Shuffled
OCC	3	1 (site/sensor)	3	2	20,560	Shuffled
SDD	48	4 (site/logical set)	12	11	58,509	Interleaved
WFR	24	4 (sensors groups)	6	4	10,913	Stream
RBF-F	10	1	10	5	50,000	Fast drift
RBF-M	10	1	10	5	50,000	Moderate drift
SEA-A	3	1	3	2	100,000	Abrupt drift
SEA-G	3	1	3	2	100,000	Gradual drift

preserve time-series progression within each stream. Interleaving was performed by continuously making a random selection of which stream to draw the next record from (drawing records in order from each stream), until all records had been drawn from all streams. Only the temperature, humidity, and CO<sub>2</sub> features were used from the OCC dataset as the light level was highly predictive on its own and the humidity-ratio was derivable from humidity and temperature. Only the raw features were used for the HIG dataset, of which each jet's features were assigned to a different primary site (3 jets, with 4 features each) and the 3 lepton features were assigned to their own primary site. Furthermore, the original HIG dataset was restricted to the first 10,000 records. Due to the small number of records in the WFR dataset, its record set was repeated once to have an adequate number of records for stream learning. The FLT dataset was constructed from a series of test flights, where each class value represented whether the moving average of the forward velocity variable over a window size of 10 records had been rising or falling. Primary sites were created for the velocity and acceleration features (along each of the lateral and vertical axes), for the reading, rate, and acceleration features (for each of pitch, yaw, and roll), and also for the readings

from each group of aileron, flap, and rudder features. Except for the FLT dataset, all real-world datasets were retrieved from the UCI Machine Learning Repository (Dheeru & Karra Taniskidou, 2017). As far as could be determined from an extensive literature review, this is the first assembled set of real-world datasets for evaluating vertically-distributed data stream mining, and could therefore be used as a standard set for future research. The synthetic RBF and SEA datasets were generated with MOA (Bifet et al., 2010). The RBF generator was configured to generate 5 possible classes with 50 incrementally drifting centroids, and two datasets were generated to simulate fast incremental drift (RBF-F, with a speed-of-change of 0.001) and moderate incremental drift (RBF-M, with a speed-of-change of 0.0001). The SEA generator was configured with 3 concept drift points (at the 25,000, 50,000, and 75,000 record points), and two datasets were generated to simulate abrupt drift (SEA-A, with a drift-width of 1) and gradual drift (SEA-G, with a drift-width of 10,000).

For experiments with each dataset, HDSM was configured with a trouble factor equal to the maximum number of features taken across the set of primary sites. Because low accuracy was observed for Park's approach (due to low agreement, as discussed later), the reported results are for a trouble factor twice that used with HDSM to give it the best chance of achieving a high accuracy (though there is little difference to results achieved with lower trouble factors). Tables 4.3 and 4.4 report the performance of each algorithm on each dataset.

Table 4.3 gives the classification accuracy and the standard deviation for each classifier configuration. The standard deviation was computed by dividing the stream into segments of 100 records each and then sampling over these segments<sup>2</sup>. Table 4.3 shows that for 10 out of 11 datasets, variations of HDSM outperform all other classifiers. For reference, the most accurate result (breaking ties by lowest standard deviation) for each dataset is bolded in Tables 4.3 and 4.4.

<sup>2</sup>The last segment was not taken into consideration if it contained fewer than 50 records.

Table 4.3: Distributed stream mining accuracy comparison.

Dataset	Distributed Ensembles						Park (TF 2×)		HDSM (TF 1×)					
	DMaxConf		DVote		DStack		MaxConf		MaxConf		Voting		Stacking	
	Acc%	(SD)	Acc%	(SD)	Acc%	(SD)	Acc%	(SD)	Acc%	(SD)	Acc%	(SD)	Acc%	(SD)
EEG	48.20	6.82	54.36	4.95	57.72	6.12	48.20	6.82	49.36	6.82	61.04	4.57	<b>61.65</b>	5.20
GAS	47.54	29.39	47.28	27.84	91.80	9.32	47.54	29.39	69.47	23.19	78.16	17.41	<b>91.94</b>	9.11
HIG	53.18	4.96	53.04	5.33	51.34	5.68	53.40	4.92	53.57	5.08	<b>53.57</b>	5.51	51.28	5.23
FLT	67.85	4.49	55.66	4.99	70.82	4.92	67.85	4.49	79.65	5.34	<b>81.78</b>	7.02	80.94	5.54
OCC	78.98	4.08	82.04	3.98	81.86	3.96	81.86	3.89	87.24	4.28	<b>90.18</b>	4.21	89.36	4.12
SDD	49.72	16.32	55.72	14.02	91.47	5.83	49.72	16.32	86.66	8.53	91.04	6.86	<b>97.17</b>	4.29
WFR	56.69	16.15	57.12	14.01	73.88	10.31	56.77	16.10	69.36	12.98	75.75	11.55	<b>80.77</b>	8.57
RBF-F	30.06	4.75	29.96	4.64	35.34	7.60	30.06	4.75	34.79	6.08	34.59	5.82	<b>37.47</b>	6.44
RBF-M	32.61	5.58	30.36	4.75	<b>58.32</b>	6.64	32.61	5.58	44.08	6.24	43.74	5.13	56.02	5.92
SEA-A	75.37	7.66	70.65	7.82	74.65	5.74	79.30	6.69	<b>82.96</b>	5.48	79.97	5.76	78.02	6.32
SEA-G	75.42	6.86	70.63	7.03	73.80	5.43	79.29	6.06	79.25	7.28	<b>79.36</b>	5.44	76.30	6.35

In order to evaluate the statistical significance of the accuracy differences between the different algorithms across the datasets, a Friedman test and an accompanying Nemenyi post-hoc analysis were performed (as recommended by Demšar (2006) for such cases when comparing multiple algorithms over multiple datasets). The null-hypothesis that “all seven algorithms produce equivalent accuracies” was rejected by the Friedman test at the 95% confidence level. The proceeding Nemenyi test produced a critical difference of 2.72, revealing which algorithms were statistically significantly different to each other, as plotted in Figure 4.3. Figure 4.3 shows that while all HDSM

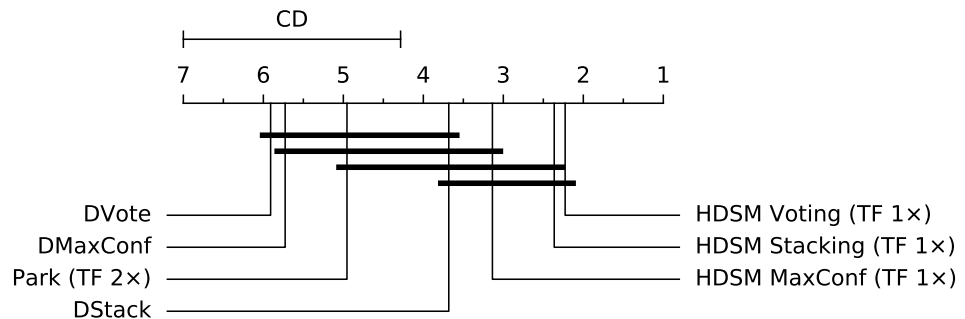


Figure 4.3: Critical difference diagram showing the statistically significant differences in accuracy between distributed stream mining algorithms.

variations were ranked higher than other algorithms, each was significantly different to a different subset. Of particular note is that HDSM with voting was significantly more accurate than all other non-HDSM algorithms other than DStack.

The accuracy study revealed the following trends:

- Park's approach outperformed DMaxConf by more than 1% in only three of the datasets, despite using trouble records to learn cross-term relationships. Its performance was hampered by the use of a single trouble site. A major limitation of using a single trouble site is that there is agreement on only a small fraction of the trouble records transmitted by all primary sites. The implication of a low level of agreement is that the trouble site is used to classify very few records, and the training set size is also low, leading to model underfitting and loss of precision. The three datasets where Park's approach did show improvement (OCC, SEA-A, and SEA-G) all have very few primary sites, and therefore suffer less from this problem.
- The distributed ensembles are clearly inadequate and only emerged the winner in one dataset (RBF-M). This under-performance was due to the absence of any mechanism to learn cross-term relationships between distributed features.
- The use of meta-learning significantly improved the performance of both HDSM and distributed ensemble approaches. Meta-learning methods such as voting and stacking provide opportunities for learning cross-term relationships. The accuracy of the distributed ensemble in particular was boosted significantly by stacking. However, it is interesting to note that a substantial gap in accuracy still exists between HDSM with stacking and DStack for most datasets. This can be explained by the fact that HDSM learns cross-term relationships by training on actual data features while DStack learns them from class labels which do not embody as much information as the original data itself.

Table 4.4 presents the resource time cost of each classifier, with the most accurate result from Table 4.3 bolded for ease of reference. DStack is orders of magnitude slower than the other distributed ensembles because the additional ARF stacked classifier is much more computationally expensive than the Naive Bayes classifiers at primary sites. The single trouble site in Park's approach typically has little effect on resource time because the low agreement means few records are actually processed at the trouble site. In terms of MTCP, HDSM is the slowest classifier because of its multiple layers of trouble site ARF classifiers that often process more features than even stacked classifiers (though this point of difference is diminished in cases where there are many primary sites with few features). On the other hand, HDSM is competitive to DStack in terms of the MTBC metric. In general, there is a much greater gap between MTCP and MTBC for HDSM when compared to the other approaches. This is due to two reasons. Firstly, lower-order trouble sites begin processing new records before higher-order trouble sites have finished processing older records. Secondly, trouble sites of the same order can process different records in parallel, as only a subset of trouble sites will be involved in processing any given record.

Table 4.5 shows the impact of increasing trouble factors with the two most promising variations of HDSM: voting and stacking (results with maximum confidence HDSM demonstrated similar trends, and are excluded for brevity). Trouble factors equal to 1, 1.5, and 2 times the maximum number of features taken across the set of primary sites were used. It can be seen that higher trouble factors generally result in improved accuracy as greater proportions of the less confident set of records are sent to trouble sites for learning and classification. Such records are more likely to be associated with cross-terms and models trained on data record fragments from multiple source sites will thus be more precise than those learned at the primary sites. However, this improved accuracy comes at the expense of increased resource time and data transmission. On the

Table 4.4: Distributed stream mining resource time comparison.

	Distributed Ensembles						Park (TF 2×)		HDSM (TF 1×)					
	DMaxConf		DVote		DStack		MaxConf		MaxConf		Voting		Stacking	
Dataset	MTCP	MTBC	MTCP	MTBC	MTCP	MTBC	MTCP	MTBC	MTCP	MTBC	MTCP	MTBC	MTCP	MTBC
EEG	6.59E3	5.78E3	6.73E3	5.95E3	1.17E5	1.10E5	8.88E3	7.80E3	2.46E5	3.52E4	3.47E5	1.30E5	<b>3.54E5</b>	<b>1.43E5</b>
GAS	1.04E4	9.08E3	1.03E4	9.13E3	1.36E5	1.25E5	1.68E4	1.60E4	2.08E5	3.28E4	2.55E5	8.28E4	<b>2.17E5</b>	<b>1.50E5</b>
HIG	6.24E3	5.88E3	6.42E3	6.10E3	8.08E4	7.45E4	7.99E3	7.03E3	1.39E5	6.87E4	<b>1.50E5</b>	<b>8.21E4</b>	1.91E5	9.90E4
FLT	6.31E3	5.78E3	6.31E3	5.83E3	9.76E4	9.13E4	8.26E3	7.52E3	2.29E5	9.34E4	<b>2.70E5</b>	<b>1.16E5</b>	2.90E5	1.25E5
OCC	5.96E3	5.77E3	5.90E3	5.74E3	1.28E5	1.22E5	1.00E4	6.81E3	7.77E4	6.77E4	<b>7.46E4</b>	<b>6.56E4</b>	1.81E5	1.40E5
SDD	9.61E3	8.57E3	9.70E3	8.68E3	1.69E5	1.59E5	1.35E4	1.17E4	1.75E5	8.42E4	1.71E5	8.53E4	<b>2.54E5</b>	<b>1.85E5</b>
WFR	7.45E3	7.04E3	7.51E3	7.09E3	1.32E5	1.25E5	9.57E3	8.65E3	1.61E5	5.68E4	1.49E5	5.66E4	<b>2.67E5</b>	<b>1.74E5</b>
RBF-F	8.34E3	7.31E3	7.99E3	7.14E3	1.10E5	1.02E5	1.28E4	1.05E4	2.95E5	6.91E4	2.97E5	6.73E4	<b>3.50E5</b>	<b>1.46E5</b>
RBF-M	8.45E3	7.44E3	8.76E3	7.91E3	<b>1.47E5</b>	<b>1.39E5</b>	1.25E4	1.05E4	3.16E5	9.51E4	3.18E5	9.04E4	3.31E5	1.65E5
SEA-A	7.17E3	6.83E3	7.33E3	7.05E3	1.68E5	1.61E5	1.21E4	8.69E3	<b>8.32E4</b>	<b>7.21E4</b>	8.15E4	7.04E4	3.06E5	2.69E5
SEA-G	6.77E3	6.37E3	7.26E3	6.92E3	1.83E5	1.75E5	1.35E4	9.30E3	7.07E4	4.51E4	<b>9.21E4</b>	<b>8.03E4</b>	3.51E5	3.08E5

Table 4.5: HDSM performance as a function of trouble factor.

Dataset	TF	HDSM Voting						HDSM Stacking					
		Acc%	(SD)	MTCP	MTBC	TDTV	MDTV	Acc%	(SD)	MTCP	MTBC	TDTV	MDTV
EEG	1×	61.04	4.57	3.47E5	1.30E5	58.72	10.55	61.65	5.20	3.54E5	1.43E5	56.22	10.28
	1.5×	64.78	5.23	5.13E5	2.66E5	91.27	13.88	63.65	5.09	4.87E5	2.63E5	84.36	13.94
	2×	66.14	4.58	5.73E5	3.25E5	104.65	14.34	66.99	5.23	5.25E5	2.99E5	95.92	15.03
GAS	1×	78.16	17.41	2.55E5	8.28E4	38.81	9.18	91.94	9.11	2.17E5	1.50E5	12.41	3.26
	1.5×	85.41	13.15	3.00E5	1.24E5	48.94	11.50	93.63	7.96	2.37E5	1.51E5	18.23	4.48
	2×	89.05	10.98	2.83E5	1.60E5	49.09	12.52	94.00	7.43	2.55E5	1.49E5	25.39	5.04
HIG	1×	53.57	5.51	1.50E5	8.21E4	54.94	23.41	51.28	5.23	1.91E5	9.90E4	43.52	21.29
	1.5×	52.76	5.12	2.44E5	1.21E5	77.95	31.23	52.91	5.67	2.95E5	1.62E5	85.10	33.83
	2×	53.11	4.81	3.95E5	2.25E5	125.21	43.24	52.92	5.23	3.62E5	1.25E5	94.03	38.45
FLT	1×	81.78	7.02	2.70E5	1.16E5	64.31	16.66	80.94	5.54	2.90E5	1.25E5	55.80	16.46
	1.5×	89.02	6.04	3.69E5	2.03E5	90.56	22.55	88.80	5.87	3.78E5	2.01E5	91.13	22.26
	2×	89.37	5.91	4.36E5	1.93E5	114.72	27.90	88.78	5.79	4.13E5	1.94E5	93.18	27.35
OCC	1×	90.18	4.21	7.46E4	6.56E4	31.17	30.12	89.36	4.12	1.81E5	1.40E5	31.34	30.29
	1.5×	89.47	4.32	1.25E5	9.56E4	69.67	45.83	90.18	3.97	2.02E5	1.44E5	48.48	45.74
	2×	89.24	6.04	1.70E5	1.24E5	77.34	51.64	87.24	5.73	1.99E5	1.36E5	56.36	42.37
SDD	1×	91.04	6.86	1.71E5	8.53E4	23.26	8.78	97.17	4.29	2.54E5	1.85E5	24.05	8.21
	1.5×	97.94	5.12	2.04E5	1.28E5	37.13	12.29	97.90	3.85	2.51E5	1.86E5	17.12	11.44
	2×	98.61	4.87	2.12E5	1.60E5	29.61	14.85	97.85	4.07	2.71E5	1.85E5	23.51	12.22
WFR	1×	75.75	11.55	1.49E5	5.66E4	51.10	20.30	80.77	8.57	2.67E5	1.74E5	47.69	18.82
	1.5×	86.23	10.82	2.90E5	1.42E5	79.86	28.57	85.32	8.46	3.45E5	1.69E5	72.46	26.92
	2×	90.32	7.69	4.00E5	2.42E5	102.13	32.93	89.07	8.60	4.16E5	2.15E5	93.64	30.04
RBF-F	1×	34.59	5.82	2.97E5	6.73E4	51.78	11.24	37.47	6.44	3.50E5	1.46E5	51.24	11.36
	1.5×	37.05	6.27	4.56E5	1.05E5	87.06	16.53	39.36	5.98	4.77E5	1.44E5	82.59	16.48
	2×	37.56	6.50	5.33E5	1.97E5	116.98	20.66	40.99	6.98	5.53E5	2.23E5	111.52	20.79
RBF-M	1×	43.74	5.13	3.18E5	9.04E4	51.72	11.11	56.02	5.92	3.31E5	1.65E5	42.22	10.92
	1.5×	45.72	5.65	4.80E5	2.17E5	81.71	16.34	54.15	6.43	4.31E5	1.63E5	62.36	16.08
	2×	52.25	5.20	5.63E5	3.63E5	107.61	20.18	55.04	6.93	4.47E5	1.61E5	71.58	18.05
SEA-A	1×	79.97	5.76	8.15E4	7.04E4	33.42	33.07	78.02	6.32	3.06E5	2.69E5	33.15	32.56
	1.5×	86.97	4.21	1.85E5	1.67E5	57.17	49.72	77.59	6.34	3.38E5	2.88E5	27.40	26.25
	2×	79.27	10.96	1.74E5	1.53E5	39.61	34.57	82.58	6.96	3.98E5	3.27E5	40.99	35.14
SEA-G	1×	79.36	5.44	9.21E4	8.03E4	33.66	33.07	76.30	6.35	3.51E5	3.08E5	33.44	30.68
	1.5×	85.78	4.35	2.02E5	1.85E5	54.56	49.69	79.52	6.97	3.78E5	3.18E5	40.18	38.61
	2×	78.73	10.38	1.74E5	1.56E5	36.62	35.00	86.71	4.27	5.15E5	3.89E5	74.24	66.37

other hand, the effective data transmission cost (captured by the MDTV metric) rarely exceeds 50%. This means that the data transmission cost (as measured by the data volume transmitted across the critical path on the trouble site hierarchy) rarely exceeded more than half of the dataset. It should also be noted that the total data transmission cost as measured by the TDTV metric rarely exceeded 100%. However, this metric does not take into account concurrent streaming of data across different pipelines of the trouble site hierarchy and is thus not a true indicator of transmission cost. Nevertheless,



it does reflect the bandwidth requirements of the data transmission network.

For a final comparison, Figure 4.4 presents the relative ranks in accuracy and resource time of DStack (being the most accurate distributed ensemble) and HDSM voting (having the best trade-off between accuracy and performance for HDSM) averaged across the eleven datasets. While the more accurate HDSM configurations are slower, HDSM voting (TF=1 $\times$ ) is often able to achieve better accuracy than DStack with better MTBC. Additionally, DStack is further away from the origin point than HDSM voting (TF=1 $\times$ ) in both charts, showing that it does not represent as good a trade-off between accuracy and resource time. Another crucial point is that HDSM's trouble factor can be varied to achieve an ideal trade-off between accuracy and resource time, whereas the performance of DStack is fixed.

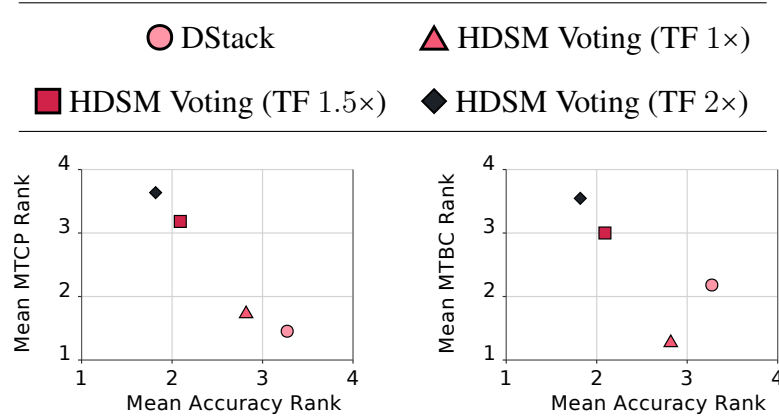


Figure 4.4: Accuracy and resource time ranks for HDSM Voting and DStack classifiers.

## 4.4 Suitability of HDSM for Anytime Classification

Given that HDSM outperformed other approaches, an important question that arises is whether it could be tuned to obtain better performance in terms of the key performance metrics of MTCP and MTBC. One distinguishing characteristic of HDSM is the distributed and layered nature of its classification model. In this context, the distribution of

HDSM accuracy over its layers is investigated with a view to determining the effectiveness of an early exit in the classification process. Experiments were performed with the FLT, WFR and GAS datasets, as well as a synthetic dataset (Synthetic Cross-Term, or SCT) that was generated with known cross-term relationships to assess the effect of layering on accuracy. The GAS dataset was configured with only four features per primary site and a trouble factor equal to the total number of features (the maximum reasonable value), which provided an extreme case with a higher degree of data distribution and transmission to trouble sites. Equation (4.2) defines the rule used for generating the SCT dataset with 32 binary features (distributed sequentially across 8 primary sites) and a binary class. This rule was devised to create complex cross-terms across many primary sites while ensuring a reasonable class balance. The results for other datasets followed the same broad trends, so were omitted in the interests of brevity. Voting was used as the aggregation method for all datasets.

$$\begin{aligned}
 class = & ((f_1 \wedge f_5) \vee (f_9 \wedge f_{13}) \vee \\
 & (f_{17} \wedge f_{21}) \vee (f_{25} \wedge f_{29})) \wedge \\
 & ((f_2 \wedge f_{18}) \vee (f_6 \wedge f_{26}) \vee \\
 & (f_{10} \wedge f_{22}) \vee (f_{14} \wedge f_{30}))
 \end{aligned} \tag{4.2}$$

Figure 4.5 shows how the accuracy and resource time vary with the maximum trouble site order used for classification. MTCP and MTBC are normalised by dividing their values with the corresponding values produced when using the primary sites only (order 0). Order 3 results are not shown for WFR because it never produced sites of this order.

The first chart in Figure 4.5 shows that higher-order sites have little or no impact on accuracy; only SCT's accuracy increases because it has relationships across many sites. This suggests that cross-terms generally involved small groups of sites. There

---

○ FLT (TF 2×)    ▲ WFR (TF 2×)    ■ GAS (Max TF)    ◆ SCT (Max TF)

---

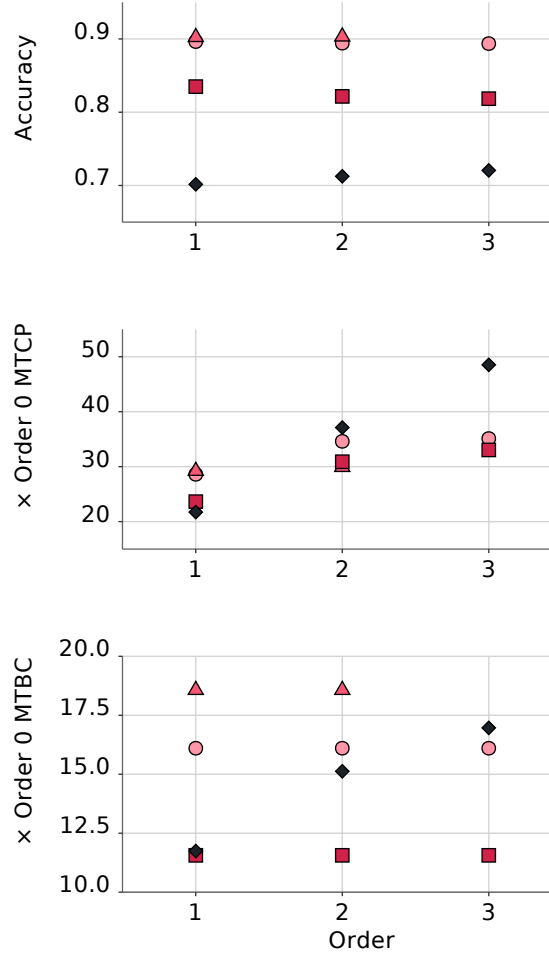


Figure 4.5: Effect of anytime classification on HDSM accuracy and resource time.

is however a significant resource time cost associated with higher-order sites. MTCP increases for all datasets when using higher-order sites, though it tails off for GAS and WFR because few records are transmitted to order 3 sites. On the other hand, MTBC generally does not increase with higher-order sites because few records are transmitted to these sites or because there is parallelism between sites at the same layer. The exception to this is SCT, where many records are transmitted to higher-order sites with little parallelism: all features (for some records) are combined at a single order 3 site. These results show clearly that an early exit generally has a minimal effect on

accuracy whilst substantially boosting MTCP and leaving MTBC either unchanged or improved in value.

The implication is that HDSM can be considered to be an “anytime” classifier in the sense that it can provide a fast and accurate classification from its lower-order trouble sites (typically order 1) without the need for using its entire hierarchy for classification. This short-circuiting process is a direct contributory factor to lowering overheads and reducing the MTCP metric value. This anytime classification property of HDSM makes it even more attractive when compared to approaches such as DStack that require input from every classifier before a final classification outcome can be determined. In effect, HDSM is now able to compete even more favourably to DStack as it maintains its accuracy advantage whilst lowering its computational overheads.

## 4.5 HDSM Parameter Sensitivity Analysis

In addition to the experiments with different trouble factors, the sensitivity of HDSM to its other parameters was also evaluated. Table 4.6 presents the results achieved with different parameter values for HDSM with voting on the WFR dataset. While varying each parameter, all other parameters were fixed to their default values (the same values used in the previous experiments). The trouble factor was fixed to the number of features per primary site.

The results in Table 4.6 show that the effects of varying these parameters are generally minor. Varying the window size limit has a small impact on accuracy, as this affects exactly which records are selected as trouble records when comparing confidence values to those in the recent window. Decreasing the threshold-time-period slightly improves accuracy at the cost of extra transmission, as this causes the initial trouble sites to be created slightly sooner. If the site utilisation threshold is increased, trouble

Table 4.6: Sensitivity analysis of HDSM parameters.

Parameter	Value	Acc%	(SD)	MTCP	MTBC	TDTV	MDTV
Window size limit	500	75.96	11.43	1.79E5	7.39E4	53.39	20.80
	1000	75.75	11.55	1.81E5	6.85E4	51.10	20.30
	2000	77.19	10.51	1.98E5	7.70E4	51.31	20.82
Threshold-time-period	100	77.89	10.38	1.94E5	7.46E4	52.69	21.29
	500	75.75	11.55	1.80E5	6.91E4	51.10	20.30
	1000	73.00	14.13	1.54E5	5.64E4	43.64	17.32
Site utilisation threshold ( $T_u$ )	0.01	75.79	11.51	1.88E5	6.96E4	57.67	20.44
	0.05	75.75	11.55	1.79E5	6.77E4	51.10	20.30
	0.15	69.57	12.58	1.38E5	3.66E4	35.22	17.82
Hoeffding bound $\delta$	1.00E-09	75.76	11.60	1.77E5	6.72E4	50.12	20.15
	1.00E-03	75.75	11.55	1.80E5	6.72E4	51.10	20.30
	1.00E-01	75.76	11.57	1.82E5	6.92E4	51.05	20.34
Smoothing Factor ( $\theta$ )	1	75.75	11.55	1.80E5	6.78E4	51.10	20.30
	5	75.75	11.55	1.77E5	6.61E4	51.10	20.30
	10	75.75	11.55	1.78E5	6.75E4	51.10	20.30

sites are removed more aggressively when they are infrequently used in classification, resulting in a notable drop in both accuracy and transmission. Finally, the changes in the smoothing factor ( $\theta$ ) and the  $\delta$  of the Hoeffding bounds demonstrate virtually no impact on the behaviour of HDSM, as the  $\delta$  has a small impact on the thresholds for trouble site creation and removal, and the smoothing factor will only impact the creation of trouble sites with very high proportions of records.

## 4.6 Limitations and Applicability of HDSM

It is worth highlighting the current limitations of HDSM, and the situations where it may not be applicable. It can be seen in Table 4.7 that for some datasets (such as EEG and RBF), even the most accurate variation of HDSM (of all aggregation method and trouble factor combinations) was not able to achieve accuracy comparable to that of an ARF classifier trained on a copy of the dataset containing all features (as if the data had been centralised, though this is not plausible in real circumstances). The complex relationships in these datasets require a greater degree of data centralisation for effective

learning. Additionally, if a primary site does not have enough information to reasonably classify the proportion of records that will not be forwarded as trouble, then those records will be classified poorly. The extreme case of this is when a primary site cannot learn any kind of model, and all records are classified with the same low confidence. In this case, the selection of trouble records based on confidence will be arbitrary. One way to combat these extreme cases is to perform complete centralisation of all records at low order trouble sites before selecting subsets of trouble records for higher-order trouble sites. Full centralisation at low order trouble sites can be achieved by using a high trouble factor (e.g. at least twice the number of features at each primary site).

Table 4.7: Comparison of HDSM and centralised model accuracy.

Dataset	Best HDSM	Centralised ARF
EEG	66.99	82.07
GAS	94.00	95.95
HIG	53.57	53.57
FLT	89.37	89.74
OCC	90.18	95.65
SDD	98.61	99.52
WFR	90.32	92.82
RBF-F	40.99	68.54
RBF-M	56.02	80.71
SEA-A	86.97	88.61
SEA-G	86.71	87.26

# Chapter 5

## Variations of HDSM

This chapter describes a collection of variations to the HDSM architecture that could be used to improve accuracy or reduce data transmission under particular circumstances.

### 5.1 Batch Transmission

Thus far, the transmission of trouble records and classification results has only been discussed in terms of individual records. However, it is possible to buffer the data for multiple records, and then transmit the buffer as a single batch from one site to another. Doing so has no impact on the classification behaviour of HDSM, as exactly the same data will be transmitted in the batches. On the other hand, batching the transmissions between sites reduces the constant overhead inherent in each separate transmission with the utilised communication protocol (such as TCP/IP).

However, there are several trade-offs to consider when transmitting record and classification data in batches. A larger batch size will increase the average latency from records arriving at a primary site to being finally classified by the aggregator. Records that are placed at the beginning of a batch will need to wait for the batch's buffer to fill with other records before they are transmitted, resulting in a delay in processing those

records. Transmitting the posterior true classes in batches also creates a delay before trouble site classifiers are able to learn from those classes. Therefore, a larger batch size will make the overall classifier less responsive to concept changes in the data streams. Finally, if batches are too large, aggregators or trouble sites may be idle while waiting for data from the next batch to arrive. Furthermore, they may then be overwhelmed with too many records at once when the batch does arrive, leading to processing delays.

Given these delays to classification and learning that can occur with larger batch sizes, a batch size must be selected that is large enough to significantly reduce transmission overhead while not unacceptably (1) increasing system latency, (2) decreasing concept drift responsiveness, or (3) overloading sites with periodic surges of records.

## 5.2 Two-Phase Trouble Record Transmission

The previously proposed method for transmitting trouble records to a trouble site can be described as “single-phase”: the key (used to join record fragments from different source sites) and features for each record identified as “trouble” at a source site are transmitted to the trouble site, and the trouble site discards any records that all source sites do not “agree” are trouble. This results in much of the transmitted data being discarded by the trouble site in cases where the agreement between source sites is low.

With the introduction of batch transmission, a two-phase transmission protocol could be used instead to reduce this wasteful transmission. In this two-phase protocol, each source site first transmits to the trouble site the key for each record it identified as trouble (the first transmission phase). The trouble site then finds the intersection of the sets of keys provided by its source sites, which represents the subset of records that the source sites “agree” on. The trouble site then sends this subset of keys back to the source sites, which then respond with the record fragments containing the features of those records (the second transmission phase). Note that the second-phase transmission



does not need to include the record keys, as record features can be transmitted in the same order as the received subset of record keys.

While it may be more efficient for a pair of source sites to find the intersection without involving the trouble site (i.e. the first source site sends its trouble record keys to the second source site, which computes and responds with the intersection of trouble record keys), using the trouble site to compute the intersection reduces computational load on source sites and is scalable to greater numbers of source sites.

The relative costs of single-phase and two-phase transmission can be compared, given:

$f$  = size of a record's feature set in one source site's fragment

$g$  = size of a record's key

$n$  = number of record fragments considered trouble at source sites (5.1)

$p$  = number of record fragments processed at trouble site

$$a(\text{agreement}) = \frac{p}{n}$$

Then the total cost of single-phase transmission is the cost of transmitting the key and features for all trouble record fragments from the source sites to the trouble site:  $n(g + f)$ . The cost of two-phase transmission is the sum of the cost of transmitting all trouble record keys from source sites to the trouble site in the first phase ( $ng$ ) with the cost of the transmitting the subset of keys back to all source sites ( $pg$ ) and the cost of transmitting the fragments of features for that same subset from the source sites to the trouble site in the second phase ( $pf$ ). Therefore, the cost of two-phase transmission is less than single-phase transmission when:

$$\begin{aligned}
ng + pg + pf &< n(g + f) \\
ng + ang + anf &< n(g + f) \\
g + ag + af &< g + f \\
ag &< f - af \\
g &< f\left(\frac{1}{a} - 1\right)
\end{aligned} \tag{5.2}$$

In general, two-phase transmission is a better choice when the size of record keys is substantially less than the size of record fragments containing features, and when the level of agreement is sufficiently low. It would even be possible to use both single-phase and two-phase transmission simultaneously for different trouble sites, with each trouble site dictating to their source sites which protocol to use based on the observed level of agreement in relation to record fragment sizes.

Another drawback that must be considered before utilising two-phase transmission is the latency introduced by the two additional transmission steps. In addition to the source sites transmitting the trouble record keys, the trouble site must respond with the intersection of the key sets, and the source sites must then transmit the feature values for that subset of records. These two additional steps approximately triple the time required to transmit trouble records to each trouble site.

### 5.3 Alternative Confidence Threshold

In the previous definition of the confidence quantile threshold (see Equations (3.2) and (3.3)), the level of agreement at a trouble site is only taken into consideration to increase the proportion of data transmitted to a subsequent trouble site, which offsets the reduction in records available to transmit due to the disagreement. Even though high disagreement at a trouble site leads to fewer received record fragments being processed

at that trouble site, the level of agreement is not used to counteract this by increasing the proportion of data transmitted from source sites. This correction is not performed because, with single-phase transmission, it would result in many more record fragments being transmitted to the trouble site, potentially overloading the transmission channel.

However, with two-phase transmission, the trouble site will receive all trouble record keys from its source sites (in the first phase), but only receive fragments containing features for records where agreement between source sites has already been established (in the second phase). Therefore, it is reasonable to control the level of transmission in each phase by means of separate trouble factors:  $t^{(1)}$  and  $t^{(2)}$ . These trouble factors can be used to reformulate the quantile threshold  $q_{s \Rightarrow d}$ <sup>1</sup>:

$$\begin{aligned}
 r_x^{(1)} &= \begin{cases} 1 & \text{if primary site} \\ \frac{1}{|S_x|} \sum_{s \in S_x} (r_s^{(2)} \times q_{s \Rightarrow x}) & \text{else if trouble site} \end{cases} \\
 r_x^{(2)} &= \begin{cases} 1 & \text{if primary site} \\ a_x \times r_x^{(1)} & \text{else if trouble site} \end{cases} \\
 q_{s \Rightarrow d} &= \min \left( 1, \frac{t_d^{(1)}}{l_d \times r_s^{(2)}}, \frac{t_d^{(2)}}{l_d \times r_s^{(2)} \times a_d} \right)
 \end{aligned} \tag{5.3}$$

where  $S_x$  is the set of source sites for a trouble site  $x$ , and reduction coefficients  $r_x^{(1)}$  and  $r_x^{(2)}$  represent the proportions of the total number of records ( $n$ ) entering the system that are transmitted to site  $x$  during the first and second phases respectively. As they are proportions, they will always lie within the interval  $[0, 1]$  (Proofs: Section A.4). It can be seen that  $r_x^{(1)} = r_x^{(2)} = 1$  for all primary sites  $x$ , as they receive and classify all records they receive. The value of  $r_d^{(1)}$  for a trouble site  $d$  is the mean<sup>2</sup> over all source

<sup>1</sup>Various proofs related to this alternative quantile threshold are provided in Appendix A.

<sup>2</sup>Even though source sites may transmit unequal numbers of record fragments in certain cases, representing  $r_d^{(1)}$  as a mean is reasonable given Equation (3.1) computes agreement in terms of the mean number of record fragments transmitted by each source site, as demonstrated in Section A.2.

sites ( $s \in S_d$ ) of the proportion of all records ( $n$ ) that are available for transmission from the source site ( $r_s^{(2)}$ ) multiplied by the proportion of those records that will be transmitted ( $q_{s \Rightarrow d}$ )<sup>3</sup>. The value of  $r_d^{(2)}$  for a trouble site  $d$  is the proportion of records received in phase 1 ( $r_d^{(1)}$ ) reduced according to the level of agreement at the trouble site ( $a_d$ ). The quantile threshold itself ( $q_{s \Rightarrow d}$ ) then determines the proportion of all records to be transmitted from source site  $s$  to trouble site  $d$  according to the specified trouble factors. As in the previous quantile threshold definition, trouble factors are set based on the processing capacity of the trouble site relative to the feature set size  $l$ , as it is guaranteed that  $r_x^{(1)} \leq \frac{t_x^{(1)}}{l_x}$  and  $r_x^{(2)} \leq \frac{t_x^{(2)}}{l_x}$  (Proofs: Section A.6). Therefore,  $t_x^{(1)}$  should always be configured with a larger value than  $t_x^{(2)}$ , representing the fact that a trouble site can receive more trouble records in the first phase (when only record keys are transmitted), than in the second phase (when feature values are transmitted, and records must be processed for classification and training).

Using the new formulation of the threshold will incur a small amount of additional communication between sites. All source sites ( $S_d$ ) for a trouble site  $d$  must regularly transmit their value of  $r_s^{(2)}$  to the trouble site for it to be able to compute  $q_{s \Rightarrow d}$  as part of the definition of  $r_d^{(1)}$ . Additionally, in order for source sites to compute  $q_{s \Rightarrow d}$ , the trouble site must regularly provide the latest value of  $a_d$ . Each agreement monitor must also receive  $r^{(2)}$  from the potential source sites it is monitoring in order to compute the quantile threshold that is used when measuring the agreement of the potential trouble site. The cost of this additional communication would be negligible in practice, as all of these values could be provided periodically along with other batch transmissions.

One potential issue of this new threshold definition is that it introduces a feedback loop between the agreement and quantile threshold: a change in the threshold changes

<sup>3</sup>Also note that  $r_a^{(2)} \times q_{a \Rightarrow d} = r_b^{(2)} \times q_{b \Rightarrow d} \forall a, b \in S_d$  when there are enough records available at the source sites (Proof: Section A.5), still ensuring that an equal number of records will be transmitted from each source site (as long as there are enough records available at the source site). This maximises potential agreement.

the proportion of records transmitted, which may impact the level of agreement, which in turn is used to determine the threshold. Instability in this relationship could cause problems for the agreement monitors, which decide to create trouble sites based on stable levels of acceptable agreement. However, any instability may be manageable through sufficiently large agreement windows that will change slowly, or by averaging the measured agreement over a period of time. Future experimentation is required to better understand the dynamics of this relationship in practice for different datasets.

It must be noted that if  $a_d = 0$  (when there is no agreement between the source sites of trouble site  $d$ ), then  $r_d^{(2)} = 0$ , and computing  $q_{s \Rightarrow d}$  will require a division by zero. In this case, the results of the divisions can be taken as  $+\infty$ , and a value of 1 will be taken as the quantile threshold, as determined by Equation (5.3). This will not flood the trouble site, as this situation is equivalent to transmitting 100% of 0 records. Note that this is an exceptional case that will be short-lived even if it occurs, as an agreement level of zero will result in no records being classified at the site, eventually triggering its removal.

## 5.4 Alternative Trouble Record Selection

Currently, trouble records are selected at each source site based on local classification confidence. However, this may not produce the optimal set of trouble records in all circumstances. Consider an example with two primary sites ( $a$  and  $b$ ) acting as source sites for a trouble site  $t$ . Each primary site has a single binary feature, meaning each full record is made up of two binary features. Table 5.1 provides one possible distribution of different feature value combinations (hereafter referred to as schemas) in the data stream. Note that some combinations of schemas representing full records never appear (probability of 0%), but there is still an equal probability of receiving either possible fragment schema at the primary sites. Table 5.2 provides a possible set of confidences

	Schema	Proportion
Full Records	(0, 0)	0%
	(0, 1)	50%
	(1, 0)	50%
	(1, 1)	0%
Site $a$ fragments	(0, $\_$ )	50%
	(1, $\_$ )	50%
Site $b$ fragments	( $\_$ , 0)	50%
	( $\_$ , 1)	50%

Table 5.1: Example distributions of schemas in example data stream.

	Schema	Confidence
Primary site $a$	(0, $\_$ )	70%
	(1, $\_$ )	50%
Primary site $b$	( $\_$ , 0)	90%
	( $\_$ , 1)	70%

Table 5.2: Example classification confidences assigned to example schemas at primary sites.

with which each schema could be classified.

If the trouble site required each primary site to forward 50% of its records as “trouble” ( $q_{a \Rightarrow t} = q_{b \Rightarrow t} = 0.5$ ), then site  $a$  would forward all of its  $(1, \_)$  fragments and site  $b$  would forward all of its  $(\_, 1)$  fragments, as these schema are classified with the lowest confidences locally (see Table 5.2). However, Table 5.1 states that  $(1, 1)$  records never appear in the stream, so agreement at the trouble site will be 0%. Figure 5.1 further illustrates the problem by plotting the expected level of agreement at the trouble site in this example as the proportion of records transmitted ( $q$ ) varies. Note that agreement will always approach 100% as the proportion transmitted approaches 100% (if all records are transmitted, then there must be full agreement), but the ideal situation is to have the line track as close to 100% agreement as possible for any value of  $q$ .

While the separate trouble thresholds for two-phase transmission described in

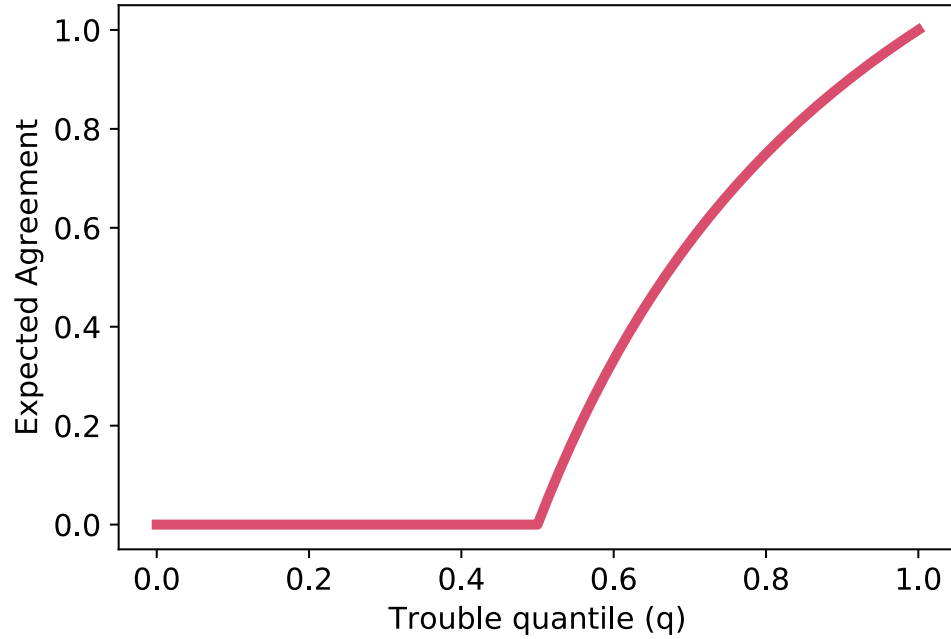


Figure 5.1: Plot of expected agreement as the proportion transmitted from primary sites to trouble sites is varied for the example data stream.

Section 5.3 can partially address this problem by increasing transmission in the presence of low agreement, it would be preferable to prioritise choosing trouble records that are more likely to achieve agreement. In the scenario described above, it would have been better for site  $a$  to transmit  $(0, \_)$  fragments. This would have resulted in  $(0, 1)$  records arriving at the trouble site, which are classified with the lowest mean confidence across the two primary sites. Unfortunately, it is not possible for each source site to know the confidence of another source site without additional transmission between these sites for every record. However, it may be possible to periodically provide a smaller amount of data to each source site that will allow it to estimate the confidence of the other source sites for each record. Therefore, this section proposes an alternative method for source sites to select trouble records based on local confidence and a probabilistic view of confidences at other source sites.

### 5.4.1 Alternative Metric for Selecting Trouble Records

This subsection will define a function  $T(x)$  of record  $x$  that can be used to prioritise records for transmission to a trouble site based on a probabilistic view of source site confidences. Firstly, let  $C_i^x$  represent the confidence for the classification of record  $x$  at source site  $i$ . While each source site knows its own confidence, it does not know the confidence at the other source sites. However, a source site  $j$  could estimate the expected confidence at another site  $i$  based on the schema of its local record fragment ( $x_j$ ), represented as:  $E(C_i^x|x_j)$ . One way of estimating this confidence value is to take the weighted mean of confidence values for each possible record schema at site  $i$  weighted by the observed probability of each schema appearing alongside  $x_j$  in the full schema ( $P(v_i \wedge x_j)$ ):

$$\begin{aligned} E(C_i^x|x_j) &= \frac{\sum_{v_i \in V_i} (C_i^v \times P(v_i \wedge x_j))}{\sum_{v_i \in V_i} P(v_i \wedge x_j)} \\ &= \frac{\sum_{v_i \in V_i} (C_i^v \times P(v_i \wedge x_j))}{P(x_j)} \\ &= \sum_{v_i \in V_i} (C_i^v \times P(v_i|x_j)) \end{aligned} \tag{5.4}$$

where  $V_i$  represents the set of possible local record schemas at site  $i$ , and  $C_i^v$  represents the classification confidence at site  $i$  for  $v_i$  (a particular local schema at site  $i$ ). The equation is simplified by recognising that  $\sum_{v_i \in V_i} P(v_i \wedge x_j) = P(x_j)$  and  $\frac{P(v_i \wedge x_j)}{P(x_j)} = P(v_i|x_j)$ .

Each source site  $s$  of a trouble site  $t$  ( $s \in S_t$ ) can then use its own local confidence and estimates of confidences at other source sites to compute  $T(x)$ :

$$T(x) = \sum_{s \in S_t} (C_s^x)^2 \tag{5.5}$$

Records with a lower  $T$  score should be prioritised for selection as trouble records according to the existing quantile threshold mechanism. A sum-of-squares approach is



used to penalise higher confidence values because a confident classification from even one site indicates that the record does not need to be considered “trouble”.

Applying this method of determining trouble records to the record distributions and confidences of the example data stream described previously results in Figure 5.2. In this plot, expected agreement linearly approaches full agreement at 50% data transmission because the source sites send  $(0, \_)$  and  $(\_, 1)$  fragments respectively. These fragments achieve agreement, except for random disagreement when the trouble proportion is less than 50%, which is caused by the selection of random subsets of records with these schema for transmission. Agreement dips after 50% as the remaining  $(1, \_)$  and  $(\_, 0)$  records are transmitted, which also achieve agreement, though once again with a degree of mismatch from random selection.

Variations of the definition of  $T$  are also possible. The contribution of each source site’s confidence could be weighted so that confidence estimations with higher variance

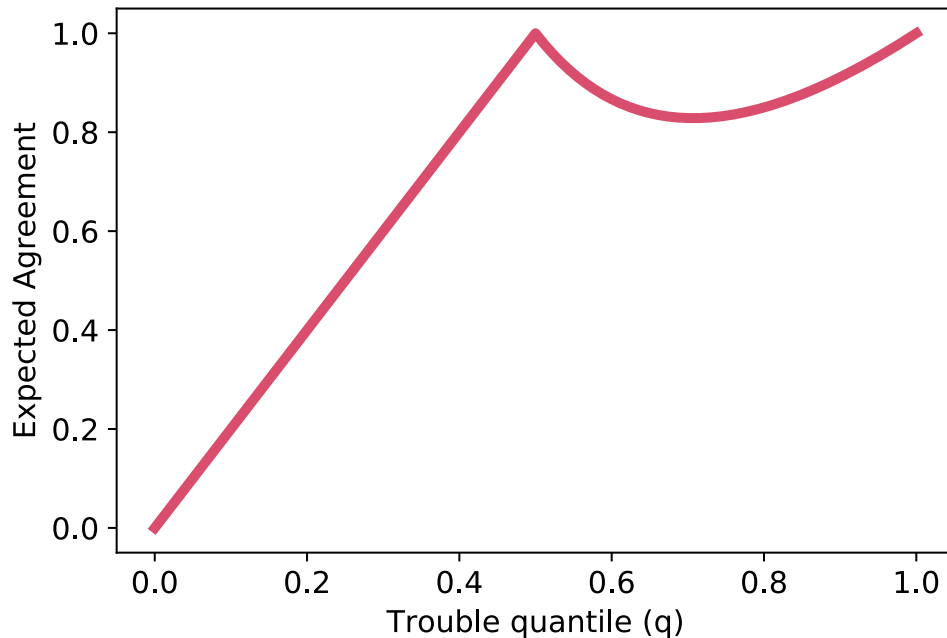


Figure 5.2: Plot of expected agreement as the proportion transmitted from primary sites to trouble sites is varied for the example data stream when trouble records are selected according to the  $T$  score.

have less weight. Another possible approach is to define  $T$  as the maximum confidence at any source site ( $\max(C_s^x) \forall s \in S_t$ ), as agreement will not be achieved when even one of the confidences is high.

If it is not possible for different sites to produce confidence values on a consistent scale, then the quantiles of confidence values could be used as a measure of confidence instead.

One disadvantage of basing trouble record selection on  $T$  is that the computational cost of computing it grows with the number of possible record schemas. Because of this curse of dimensionality, using  $T$  is likely infeasible for data streams with many features, high feature cardinalities, or numeric features. An exception to this is if a classification model produces fewer unique confidence values than there are unique schemas, such as in decision trees that have a unique confidence value for each leaf node. In this case, the cost of computing  $T$  would scale with the number of leaves.

### 5.4.2 Communication of Confidence Distributions

One challenge of implementing the  $T$  metric defined above lies in how to estimate the source site confidences for each schema ( $E(C_s^x|x_v) \forall s, v \in S_d, V$ ) and transmit them to all other source sites. Two possible solutions are presented below.

#### Centralised Random Sample

The first approach involves centralising a random sample of record fragments and their associated confidences from all source sites to a new distribution monitor site associated with each trouble site. With a sample of merged records, the confidence values can be estimated by:

$$E(C_i^x|x_j) = \frac{1}{|N(x_j)|} \sum_{n \in N(x_j)} C_i^n \quad (5.6)$$

where  $N(x_j)$  is the subset of sampled records that contain schema fragment  $x_j$ . This can be computed for all source sites ( $S_d$ ) and all possible schemas at other sites ( $x_j$ ) with a single pass over the sample of records ( $N$ ) with complexity  $O(|N||S_d|^2)$ , as demonstrated in Algorithm 4.

The estimates should be continuously updated as more samples are transmitted to the distribution monitor, and a method like exponential down-weighting could be used in `updateMean` to give greater weight to recent confidence values. The proportion of records sampled should be large enough to provide a significant sample of each unique schema, though this means the sample size will need to increase with the dimensionality and cardinality of the data stream.

A method is also required to ensure that the fragments sampled from each source site are for the same records. One possibility is to use a deterministic method of selecting records based on their keys. For example, if keys are monotonically increasing, records where  $key \bmod p = 0$  could be sampled to select a proportion of  $\frac{1}{p}$  records (provided the periodicity does not bias the selection of certain schemas over others). Alternatively, records could be sampled randomly at each source site, and a two-phase transmission scheme (similar to that in Section 5.2) could be used to only transmit features and confidence values from the intersection of records. The statistical predictability of agreement in the case of random selection (as described in Section 3.3.1) could be used to choose the size of the random sample that will result in a second phase transmission with the desired quantity of records.

### **Aggregating Confidences at the Aggregator Site**

The second approach to sharing confidences takes advantage of the confidence values already collected at the aggregator site. Specifically, the aggregator can use the definition in Equation (5.5) to compute  $T$  scores for each trouble site for every record based on the classification confidences from the source sites of each trouble site. The aggregator

**Result:** Confidence estimates for each source site given each schema fragment at other source sites

**Input:**  $\text{sourceSites} \leftarrow S_d, N \leftarrow \text{sample of full records},$   
 $\text{siteConfidences} \leftarrow \text{confidences from each source}$   
 $\text{site for each record}$

*/\* Populate default estimates for missing schemas \*/*

1 **foreach**  $j$  *in*  $\text{sourceSites}$  **do**

2     **foreach**  $x_j$  *in*  $\text{getSiteSchemas}(j)$  **do**

3         **foreach**  $i$  *in*  $\text{sourceSites}$  **do**

4              $\text{estimates}[i, x_j] \leftarrow 0.5$

5         **end**

6     **end**

7 **end**

*/\* Update estimates based on each sampled record \*/*

8 **foreach**  $x$  *in*  $N$  **do**

9     **foreach**  $j$  *in*  $\text{sourceSites}$  **do**

10          $x_j \leftarrow \text{getSiteSchema}(x, j)$

11         **foreach**  $i$  *in*  $\text{sourceSites}$  **do**

12             **if**  $i \neq j$  **then**

13                  $\text{updateMean}(\text{estimates}[i, x_j], \text{siteConfidences}[i, x])$

14             **end**

15         **end**

16     **end**

17 **end**

**Algorithm 4:** Compute confidence estimates.

can then transmit the trouble site's  $T$  scores for all records back to each of the source sites, where they can be grouped by the local schema associated with each record and averaged to produce an estimate of the  $T$  score for each local schema:

$$E(T(x)|x_j) = \frac{1}{|N(x_j)|} \sum_{n \in N(x_j)} T(n) \quad (5.7)$$

where  $N(x_j)$  is the set of records with the local schema  $x_j$ , and  $T(n)$  represents the  $T$  score provided by the aggregator for record  $n$ .

This approach may even improve agreement further than the approach above, as the same historical  $T$  scores will be used by each source site when estimating the  $T$  score for a new record.

However, this approach would introduce transmission from the aggregator back to each group of source sites, and would also require the source sites to continuously update their estimates of  $T$  for each local schema (ideally using exponential down-weighting to bias the effect of recent records).

### 5.4.3 Selecting Trouble Sites

Given trouble site selection is based on agreement (as measured by the agreement monitors), changing the way trouble records are selected requires a new way of choosing which pairs (or larger groups) of sites are good candidates to be the source sites for a new trouble site.

One approach could be to calculate the  $T$  score for all records and all possible groups of source sites. Groups that achieved a lower mean<sup>4</sup>  $T$  score would be more likely to classify the same records with low confidence, and therefore be more promising candidates. In practice, the mean  $T$  score should only include records with a  $T$  score below the estimated quantile threshold for the prospective trouble site, as these are the

---

<sup>4</sup>This could be averaged over a sliding windowing or via exponential down-weighting.

records that are the targets for transmission to the trouble site.

## 5.5 Additional Variations

Additional variations to the HDSM architecture are provided in appendices. Appendix B describes methods that can be used to compress the size of trouble record batches, improving transmission efficiency. Appendix C explores the possibility of omitting the transmission of unconfident local classifications to the aggregator, including its impacts on classification accuracy and transmission protocols.

## 5.6 Section Summary

The previous chapters have described the novel HDSM architecture for distributed data stream mining. However, HDSM as described does nothing to protect the privacy of individuals represented by the trouble records that are transmitted to trouble sites. The following chapters propose novel data perturbation methods for privacy-preserving data stream mining, and discuss how these methods can be combined with HDSM to perform distributed privacy-preserving data stream mining.

## **Chapter 6**

# **Related Work in Privacy-Preserving Data Mining**

### **6.1 Existing Techniques for PPDM and PPDP**

Many different approaches to privacy-preserving data mining (PPDM) and privacy-preserving data publishing (PPDP) have been proposed previously, often based on different models of how privacy is interpreted in practice. The most prominent models are reviewed below, with a focus on their relevance to data stream mining.

#### **6.1.1 Secure Multiparty Computation**

One approach to privacy-preservation is to develop methods that allow models to be produced without sharing any record-level data. For example, secure multiparty computation (SMC) methods provide strict cryptographic limits to the information gained by parties involved in the mining process. However, SMC often requires many rounds of communication between parties, which makes it too computationally expensive for use in online learning (Mendes & Vilela, 2017). The usefulness of

other, similar methods that can operate on streams is restricted by the fact they were specifically devised for particular types of data mining algorithms (J. Wang, Liu, Fu, Luo & Li, 2019).

### 6.1.2 Anonymisation

A different privacy-preserving approach is to perform anonymisation that ensures any individual's record in a dataset is indistinguishable from a group of similar individuals. This form of guarantee is often modelled with the k-anonymity framework and its derivatives: l-diversity, t-closeness, etc. (Aggarwal & Philip, 2008). While this approach has been applied in the context of data streams (Abdelhameed, Moussa & Khalifa, 2019; Otgonbayar, Pervez, Dahal & Eager, 2018; Sakpere & Kayem, 2014; J. Wang, Deng & Li, 2018), achieving k-anonymity when joining partial records from multiple datasets (e.g. merging an individual's financial history with their medical history, which may be owned by different parties) still requires many rounds of communication between parties (Jiang & Clifton, 2006; K. Wang, Fung & Dong, 2005).

### 6.1.3 $\epsilon$ -Differential Privacy

Another popular privacy model is that of  $\epsilon$ -differential privacy, which guarantees that adding or removing one individual's data from a dataset results in a maximal change relative to  $\epsilon$  in any released information (Dwork, 2008). This ensures that the presence or absence of a particular individual in the dataset has a limited impact on the released information, thereby protecting each individual's privacy. While this model has been applied in the context of data streams (Cao & Yoshikawa, 2016; Dwork, Naor, Pitassi & Rothblum, 2010; Kellaris, Papadopoulos, Xiao & Papadias, 2014; Q. Wang et al., 2018), these methods control the privacy of particular analytical results (such as summary statistics), and therefore cannot be used to facilitate the sharing of record-level data for



general data mining.

#### **6.1.4 Data Perturbation**

Finally, data perturbation methods alter the values of dataset records to prevent the recovery of their original values while still retaining desirable properties of the dataset. Methods such as rank-swapping (Nin, Herranz & Torra, 2008), condensation (Aggarwal & Philip, 2004), randomised response (Huang & Du, 2008), and additive noise (Agrawal & Srikant, 2000) retain dataset-wide properties (such as summary statistics), but at the expense of the properties of individual records. Other methods such as geometric perturbation (K. Chen, Sun & Liu, 2007) and random projection (K. Liu, Kargupta & Ryan, 2006) preserve (or approximately preserve) the pairwise distances between records in the dataset. This makes them more useful for performing data mining tasks that make predictions about particular records, such as classification and regression. While these transformations are typically targeted at numeric data, many real-world data streams involve numeric data (such as IoT sensors). Furthermore, these techniques typically involve straightforward transformations of records, making them efficient to continuously apply to data streams.

Relatively little research has been performed on data perturbation for preserving privacy in the context of data streams. F. Li, Sun, Papadimitriou, Mihaila and Stanoi (2007) proposed an additive noise approach that accounts for the correlation and auto-correlation of streams. They demonstrated the method's resistance to additive noise attacks that take advantage of correlation, but their study did not include an evaluation of the impact on the accuracy of data mining algorithms. Rodríguez, Nin and Nuñez-del Prado (2017) evaluated the impact of additive noise perturbation and other non-perturbative methods on stream mining algorithms, but their privacy evaluation model was based on anonymisation rather than value recovery. Finally, a method combining

both random rotation and condensation (generating synthetic records with similar statistical properties to the original dataset) was proposed by Chamikara, Bertok, Liu, Camtepe and Khalil (2018), but it was not evaluated with stream mining algorithms. Furthermore, as the synthetic records produced by condensation do not represent particular records in the original stream, it is not a viable approach for producing predictions about the individual records in the original data stream.

## **6.2 Methods for Privacy-Preserving Data Perturbation**

Given its potential to be applied to data streams, the data perturbation model is used as the basis for the methods proposed in the next Chapter. In this section, the development of data perturbation methods used for PPDM and PPDP is reviewed.

### **6.2.1 Additive Noise**

Additive noise is one of the earliest examples of privacy-preserving data perturbation (Agrawal & Srikant, 2000). This method distorts numeric values by adding random values which are drawn from a standard distribution (such as a uniform or normal distribution) with mean zero. The greater the variance of the noise distribution, the more the values are distorted, resulting in greater privacy and lower data utility for analysis. However, it has been shown that additive noise is susceptible to a variety of privacy-breaking attacks, including spectral filtering (Kargupta, Datta, Wang & Sivakumar, 2003), Eigen-analysis, Maximum A Posteriori (MAP) estimation, and distribution analysis (K. Liu, Giannella & Kargupta, 2008). Furthermore, because the noise is generated separately for each record, the relationships between records can be unacceptably distorted (K. Liu, Kargupta & Ryan, 2006).

### 6.2.2 Random Rotation

Random rotation was later proposed as a means of preserving privacy while also retaining the relationships between records in a dataset (K. Liu, Kargupta & Ryan, 2006). By performing a matrix multiplication between a dataset with  $m$  features and a random  $m \times m$  orthogonal matrix, a perturbed dataset can be produced with the same number of records and features. Privacy is preserved through the generation of the new set of features, each of which is a linear combination of the full set of original features. Because the transformation is orthogonal, the distance between each pair of records is equal before and after the perturbation, which means many data mining algorithms will perform identically on the original and perturbed datasets (Giannella, Liu & Kargupta, 2013). However, several attacks have been proposed to approximately reverse the rotation, including ICA-based attacks that require no prior knowledge (K. Liu, Kargupta & Ryan, 2006), PCA-based attacks that require a sample of data drawn from the same distribution as the original dataset (K. Liu, Giannella & Kargupta, 2006), and known input-output attacks that require pairs of original records matched with their perturbed versions. It has been shown that as few as  $m$  known input-outputs can perfectly reverse the rotation (K. Liu, Giannella & Kargupta, 2006). Carefully choosing the orthogonal matrix and augmenting the transformation with a random translation (adding an additional constant value to each value relative to the range of the feature) and additive noise can help resist some of these attacks (K. Chen et al., 2007), but attacks have been developed for combinations of random rotation with translation (Giannella et al., 2013) and random rotation with additive noise (L. Liu, Wang & Zhang, 2008).

### 6.2.3 Random Projection

Random projection has been proposed as a way to address the vulnerabilities of random rotation while still preserving distances between records as much as possible (K. Liu, Kargupta & Ryan, 2006). Random projection replaces the distance-preserving orthogonal matrix with one that is only approximately distance-preserving (such as a matrix of values drawn from a normal distribution with mean zero), and also reduces the dimensionality of the matrix to produce a projected dataset. The dataset can be projected column-wise (to preserve the number of records and the distances between them) or row-wise (to preserve the number of features and the distances between them). However, only the former is applicable in the context of a data stream, as the latter requires all records to be known before the transformation is applied. Random projection resists the ICA-based attacks used against random rotation because the reduced dimensionality results in an under-determined system of linear equations (K. Liu et al., 2008). Like the other perturbation methods discussed above, random projection is designed for numeric data. Application to discrete data is possible, though it has been noted this is much more susceptible to privacy breaches in cases where the attacker has prior knowledge of the original data schema (K. Liu, Kargupta & Ryan, 2006).

## 6.3 Attacks on Random Projection

As random projection forms the foundation for the data perturbation methods proposed in the next chapter, the primary attacks against it are reviewed in more detail. These methods have also been covered in previous surveys of attacks on data perturbation (K. Liu et al., 2008; Okkalioglu et al., 2015). All of these attacks assume some level of prior knowledge on the part of the attacker.

### 6.3.1 Known Sample Attacks

While the PCA-based known sample attack on random rotation is not effective against random projection (K. Liu, 2007), other known sample attacks have been devised based on under-determined ICA (Sang, Shen & Tian, 2012) and MAP estimation with approximated covariances and means (Sang et al., 2012). However, the number of known sample records required to produce an accurate estimate of the dataset distribution makes this an unlikely avenue of attack

### 6.3.2 Known Projection Matrix Attacks

It has been shown that if an attacker knows at least as many matching pairs of original and perturbed records (input-output pairs) as there are features in the original dataset ( $m$ ), then they will be able to perfectly recover the matrix that was used to perturb the dataset (K. Liu, 2007). The  $l_1$ -recovery attack makes use of a known projection matrix as prior knowledge, but it is specifically targeted at sparse datasets (Zhao, Yang & Zhang, 2014).

### 6.3.3 Known Input-Output Attacks

If fewer than  $m$  input-output pairs are known<sup>1</sup>, then a known input-output MAP attack may be made against random projection (K. Liu, 2007). Sang et al. (2012) note that knowledge of input-output pairs can be combated by shuffling records before publication, but this may not be possible when publishing a continuous data stream. Furthermore, Giannella et al. (2013) have proposed a way to match input records to perturbed records without other prior knowledge in the case of a random rotation, and it may be possible to adapt this technique for use against random projection in an approximate fashion.

---

<sup>1</sup>The case when at least  $m$  input-output pairs are known is discussed in Section 7.2.7.

A combination of random projection with a non-linear *repeated Gompertz* function has recently been proposed to resist MAP attacks while retaining enough data utility for clustering tasks (Lyu, Bezdek, Law, He & Palaniswami, 2018). However, the extreme nature of the non-linear transformation may make this approach infeasible for other tasks, such as classification, where decision boundaries can become unacceptably distorted.

It is highly plausible for an attacker to have the few known input records required for a known input-output attack. For example, if each record is associated with a particular individual, a small group of malicious individuals could collude and make use of their own private data to breach the privacy of other individuals in the dataset. For this reason, the known input-output MAP attack type can be considered the most plausible attack on random projection, and therefore it is used as the basis for the attacks used to evaluate the privacy-preserving capabilities of the proposed data perturbation methods.

## Chapter 7

# Proposed Perturbation Methods for Privacy-Preserving Stream Mining

### 7.1 Data Perturbation Methods

This section defines the foundational data perturbation method based on random projection and translation, and the two proposed methods that make use of independent and cumulative additive noise respectively.

In all models, the convention is adopted of representing the original dataset as an  $m \times n$  matrix  $X$  where columns represent data records and rows represent data features. Each perturbation method will transform  $X$  to produce a perturbed dataset represented by a  $k \times n$  matrix  $Y$ , where  $k \leq m$ . Note that the number of records does not differ between  $X$  and  $Y$ , though the number of features may be reduced.

#### 7.1.1 Foundational Random Projection Model

The random projection perturbation method can be represented by the matrix multiplication  $Y = \frac{1}{\sqrt{k}\sigma_r}RX$ , where  $R$  is a  $k \times m$  random matrix, each element of which is

i.i.d. and drawn from a Gaussian distribution with mean zero and variance  $\sigma_r^2$  (K. Liu, Kargupta & Ryan, 2006, Lemma 5.3). Multiplying the projection by  $\frac{1}{\sqrt{k}\sigma_r}$  ensures that the column-wise inner-product is preserved when merging horizontally-distributed datasets (datasets that represent different sets of records with the same set of features) perturbed with the same  $R$ . This is not strictly necessary when working with only one dataset, as changing the scale of the dataset as a whole will have little or no impact on many data mining tasks.

The basis of the random projection method is the Johnson-Lindenstrauss lemma (Johnson & Lindenstrauss, 1984), which proves that a dataset of  $s$  records can be reduced to  $O(\frac{\log s}{\varepsilon^2})$  dimensions while still retaining the pairwise distances within a small margin of error  $\varepsilon$  (K. Liu, Kargupta & Ryan, 2006, Lemma 5.1). Therefore, random projection can be considered an approximately distance-preserving perturbation. K. Liu, Kargupta and Ryan (2006, Lemma 5.6) show that as  $k$  is reduced, the pairwise distance error increases exponentially, but there is also a corresponding increase in resistance to privacy-breaching attacks (K. Liu, 2007).

### **Resisting Rotation Centre Attacks with Random Translation**

K. Chen et al. (2007) show that one deficiency of distance-preserving perturbations is that records near the rotation centre (i.e. origin) are perturbed less than records further away (i.e. record vectors having greater magnitude). This means that perturbed records with low magnitude may be adequate estimates of their original counterparts, without the need for any sophisticated attack methods.

While this vulnerability may be somewhat mitigated by the dimensionality reduction of random projection, the vulnerability can be completely countered by performing a random translation as part of the perturbation (K. Chen et al., 2007). This extends the perturbation method to  $Y = \frac{1}{\sqrt{k}\sigma_r}RX + \Psi$ , where each column in  $\Psi$  is identical (so the same translation is applied to each record) and the element in each row is a randomly



positive or negative value drawn from a uniform distribution. The uniform distribution ranges from a minimum translation equal to the range of the corresponding feature ( $F$ ) and a maximum translation of twice the range:

$$\begin{aligned}\Psi_{*,i} &= \Psi_{*,j}, 1 \leq i < j \leq n \\ \psi_{i,j} &= \mathcal{B}(-1, 1) \times \mathcal{U}(\mathcal{R}(F_i), 2\mathcal{R}(F_i))\end{aligned}\tag{7.1}$$

where  $\mathcal{R}(F_i)$  is the range of the feature in row  $i$ ,  $\mathcal{B}(a, b)$  is a binary random distribution with equal probabilities of producing either  $a$  or  $b$ , and  $\mathcal{U}(a, b)$  is a uniform distribution ranging from  $a$  to  $b$ .

Applying such a constant translation to all records has no effect on many common data mining tasks, but a later section will demonstrate that an attacker must sacrifice one known input-output pair in order to account for it.

This foundational perturbation method involving random projection and random translation will hereafter be referred to as RP.

### 7.1.2 Random Projection with Independent Noise

To add an additional degree of control over the accuracy/privacy trade-off, two different forms of additive noise are proposed to extend the RP method. The first of these is to add i.i.d. Gaussian noise to every value in the perturbed dataset. This extension of the foundational perturbation method to include “independent” noise will be referred to as RPIN. The variance of the noise should be proportional to the range of each feature:

$$\begin{aligned}Y &= \frac{1}{\sqrt{k}\sigma_r}RX + \Psi + \Delta \\ \delta_{i,j} &= \mathcal{N}(0, \sigma_\delta^2 \cdot \mathcal{R}(F_i))\end{aligned}\tag{7.2}$$

The motivation for this additive noise is to add a degree of uncertainty to any recovery attempts made with attacks that attempt to reverse the random projection on

the basis of its approximately distance-preserving properties. However, this distortion comes at a proportional cost to data mining tasks that are dependent on pairwise distances. This accuracy/privacy trade-off is controlled through the  $\sigma_\delta$  parameter. While this form of additive noise has previously been combined with random rotation (K. Chen et al., 2007), it has not been combined with random projection as far as could be determined from an extensive literature review.

### 7.1.3 Random Projection with Cumulative Noise

The second proposed additive noise extension to RP is an entirely novel method specifically designed for a data stream mining context. This extension of the foundational perturbation method to include “cumulative” noise will be referred to as RPCN. As with RPIN, i.i.d. Gaussian values are added to each record, but each random value is also added to every subsequent record in the stream, such that the noise accumulates:

$$Y = \frac{1}{\sqrt{k}\sigma_r}RX + \Psi + \Gamma$$

$$\gamma_{i,j} = \begin{cases} \mathcal{N}(0, \sigma_\gamma^2 \cdot \mathcal{R}(F_i)) & j = 1 \\ \mathcal{N}(0, \sigma_\gamma^2 \cdot \mathcal{R}(F_i)) + \gamma_{i,j-1} & j > 1 \end{cases} \quad (7.3)$$

where  $i, j$  are the feature and record indexes respectively.

Because the sum of  $n$  i.i.d. variables drawn from a Gaussian distribution is itself a Gaussian distribution with the same mean and a variance of  $n\sigma^2$ , the difference in  $\Gamma$  noise between two records in a stream is proportional to the number of records separating them. In essence, the successive values along each row of  $\Gamma$  can be considered a Gaussian random walk. This property is particularly useful for resisting known input-output attacks, where an attacker will face increasing levels of noise when attempting to recover records further away from known records. It also means that using a small  $\sigma_\gamma$  can result in the same equivalent noise as independent noise with a much larger  $\sigma_\delta$ ,

as explained in the next section. By using a small  $\sigma_\gamma$ , the effect on pairwise distances among nearby records in the stream is also minimised, allowing for more accurate data mining. The gradual motion of the random walk over time can then be considered a form of concept drift, which many stream mining algorithms are designed to adapt to as they learn (Bifet & Kirkby, 2009). Therefore, RPCN is designed to achieve a similar privacy benefit to that of RPIN, but with less impact on the accuracy of data stream mining algorithms.

#### 7.1.4 Comparison of Independent and Cumulative Noise

In order to perform a fair comparison between independent and cumulative noise, the relationship between the  $\sigma_\delta$  and  $\sigma_\gamma$  parameters that will result in an equivalent total amount of noise must be established. To simplify the equations involved in the comparison, dataset features are assumed to be min-max normalised such that  $\mathcal{R}(F_i) = 1$ . This allows the Gaussian distributions in the definitions of  $\delta_{i,j}$  and  $\gamma_{i,j}$  to be simplified to  $\mathcal{N}(0, \sigma_\delta^2)$  and  $\mathcal{N}(0, \sigma_\gamma^2)$  respectively.

Finding the total independent noise over a stream of  $n$  records is then straightforward, as the amount of noise added to each record is i.i.d. The amount of noise added to each feature  $F_i$  can be considered to be the absolute difference between each record and its original value, which is modelled by the half-normal distribution  $|\mathcal{N}(0, \sigma_\delta^2)|$  (the mean of which is  $\sigma_\delta \cdot \sqrt{\frac{2}{\pi}}$ ). Therefore, the expected value of the independent noise  $E_\delta$  added over a stream of  $n$  records is:

$$\begin{aligned} E_\delta &= \sum_{i=1}^n \left( \sigma_\delta \cdot \sqrt{\frac{2}{\pi}} \right) \\ &= n \cdot \sigma_\delta \cdot \sqrt{\frac{2}{\pi}} \end{aligned} \tag{7.4}$$

Finding the total cumulative noise ( $E_\gamma$ ) over a stream of  $n$  records can be found similarly, but the fact that the variance of the accumulated noise added to each record

increases linearly for subsequent records must be accounted for:

$$\begin{aligned}
 E_{\gamma} &= \sum_{i=1}^n \left( \sqrt{i \cdot \sigma_{\gamma}^2} \cdot \sqrt{\frac{2}{\pi}} \right) \\
 &= \sum_{i=1}^n \left( \sqrt{i} \cdot \sigma_{\gamma} \cdot \sqrt{\frac{2}{\pi}} \right) \\
 &= \sigma_{\gamma} \cdot \sqrt{\frac{2}{\pi}} \cdot \sum_{i=1}^n \sqrt{i}
 \end{aligned} \tag{7.5}$$

Given the above expressions for  $E_{\delta}$  and  $E_{\gamma}$ , the equivalence point  $\gamma_y$  can be found from:

$$\begin{aligned}
 E_{\delta} &= E_{\gamma} \\
 n \cdot \sigma_{\delta} \cdot \sqrt{\frac{2}{\pi}} &= \sigma_{\gamma} \cdot \sqrt{\frac{2}{\pi}} \cdot \sum_{i=1}^n \sqrt{i} \\
 n \cdot \sigma_{\delta} &= \sigma_{\gamma} \cdot \sum_{i=1}^n \sqrt{i} \\
 \sigma_{\gamma} &= \sigma_{\delta} \cdot \frac{n}{\sum_{i=1}^n \sqrt{i}}
 \end{aligned} \tag{7.6}$$

Figure 7.1 demonstrates this relationship by plotting the amount of noise expected to be added to each of 1000 records when  $\sigma_{\delta} = 0.1$  and the total cumulative noise is equivalent ( $\sigma_{\gamma} \approx 0.0047$ ).

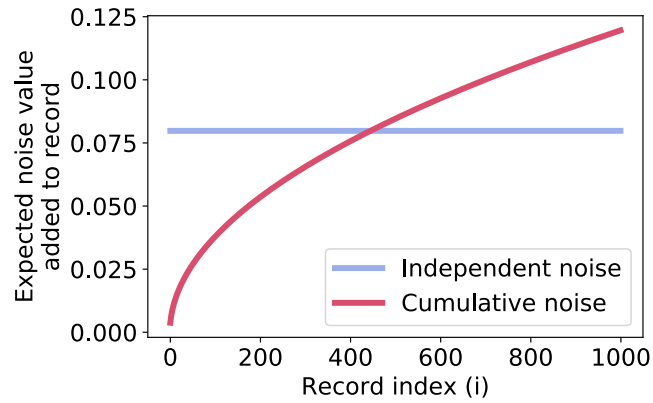


Figure 7.1: Comparison of independent and cumulative noise when the total noise of each approach is equal.

One drawback to cumulative noise that is apparent in Figure 7.1 is that the difference in noise between records that are nearby in the data stream is much less than that achieved with equivalent independent noise. The impact of this depends on the real-world context in which the masking is applied. For some data streams, an attacker with some set of known input-output records may already have some knowledge of other nearby records. For example, an individual may publicly share their location at some times, but wish for it to remain private at other times. The attacker's goal will therefore be to attack records distant from the ones they already know. However, as the noise accumulates over a period of time, this may be a much more difficult attack to perform than one on independent noise.

### 7.1.5 Interpolating Cumulative Noise Between Known Points

It is worth noting that if the cumulative noise added to two records ( $\gamma_a$  and  $\gamma_c$ ) is known, then the cumulative noise added to a record between them in the stream ( $\gamma_b, a < b < c$ ) can be estimated by modelling the joint probability distribution  $P(\gamma_b|\gamma_a \wedge \gamma_c)$  as a product of the two Gaussian probability distributions  $P(\gamma_b|\gamma_a)$  and  $P(\gamma_b|\gamma_c)$  (Smith, 2011)<sup>1</sup>:

---

<sup>1</sup>[https://ccrma.stanford.edu/~jos/sasp/Product\\_Two\\_Gaussian\\_PDFs.html](https://ccrma.stanford.edu/~jos/sasp/Product_Two_Gaussian_PDFs.html)

$$\begin{aligned}
P(\gamma_b|\gamma_a) &= \mathcal{N}(\gamma_a, (b-a)\sigma_\gamma^2) \\
P(\gamma_b|\gamma_c) &= \mathcal{N}(\gamma_c, (c-b)\sigma_\gamma^2) \\
P(\gamma_b|\gamma_a \wedge \gamma_c) &= \mathcal{N}(\mu, \sigma^2) \\
\mu &= \frac{\gamma_a(c-b)\sigma_\gamma^2 + \gamma_c(b-a)\sigma_\gamma^2}{(c-b)\sigma_\gamma^2 + (b-a)\sigma_\gamma^2} \\
&= \frac{\gamma_a(c-b) + \gamma_c(b-a)}{c-a} \\
\sigma^2 &= \frac{(b-a)\sigma_\gamma^2(c-b)\sigma_\gamma^2}{(b-a)\sigma_\gamma^2 + (c-b)\sigma_\gamma^2} \\
&= \frac{(b-a)(c-b)\sigma_\gamma^2}{c-a}
\end{aligned} \tag{7.7}$$

Note that the effective variance decreases as  $b$  comes closer to either  $a$  or  $c$  because  $(b-a)(c-b)$  is maximised when  $b = \frac{c+a}{2}$ . While knowing noise values on either side of an unknown record reduces the variance of the effective cumulative noise of the unknown record, it is unlikely to occur in practice. Assuming the pairwise distance is perfectly preserved so that  $|x_c - x_a| = |y_c - y_a|$  (which it is on expectation, because random projection preserves pairwise distance approximately), then the magnitude of the difference between  $\gamma_c$  and  $\gamma_a$  can be found:  $|\gamma_c - \gamma_a| = |y_c - y_a| - |x_c - x_a|$ . However, it is still not possible to determine the difference vector  $\gamma_c - \gamma_a$  because it is in the transformed feature space produced by the random projection.

### 7.1.6 Data Perturbation Efficiency

As the computational complexity of the matrix multiplication involved in the random projection of each record is determined by the number of features in the original and projected datasets ( $O(km)$ ,  $k \leq m$ ), the additional operations required for the random translation and additive noise do not significantly affect performance. In the case of RPIN, the translation and generated noise must be added to each of the  $k$  features of the projected record, resulting in a total complexity still proportional to  $O(km)$ . For

RPCN, the generated noise can simply be added to the translation before it is added to the projected record, resulting in the same overall complexity as independent noise. By adding the noise to the random translation stored in memory, it will also be applied to all subsequent records. The only other performance consideration for additive noise is the requirement of an entropy source from which the noise for each record can be efficiently drawn.

### 7.1.7 Applying Noise to the Random Projection Matrix

While applying additive noise to the random projection matrix itself was considered, this avenue was not pursued because it would result in smaller changes to records with low magnitude (closer to the origin). Applying additive noise to the projected records instead ensures the degree of perturbation is independent of the record's magnitude.

## 7.2 Known Input-Output Attacks

This section describes the proposed known input-output attacks that are designed to breach the privacy of the data perturbation methods described in the previous section. The basis for any known input-output attack is that the attacker has prior knowledge of a small subset of input records and the perturbed records they correspond to. This can commonly occur when each record of a data stream represents an individual, in which case one or more individuals may share their private records with each other in order to breach the privacy of other individuals represented within the data stream.

More formally, it is assumed the attacker knows  $p$  columns of the input stream  $X$  (represented as  $X_p$ ) and their corresponding columns in the output stream  $Y$  (represented as  $Y_p$ ). The attacker will then use this prior knowledge to attack other output columns ( $y_i \in Y \setminus Y_p$ ) in order to produce a recovered estimate ( $\hat{x}_i$ ) of the original input column ( $x_i$ ). Following existing convention (K. Liu et al., 2008), it is also assumed that the

attacker knows the perturbation method that was applied to  $X$ , and that they know any variances involved ( $\sigma_r^2$ ,  $\sigma_\delta^2$ , and  $\sigma_\gamma^2$ ). This information may be leaked, or may need to be shared by multiple organisations following the same perturbation procedure before sharing and merging data. As in the previous section, dataset features are assumed to be min-max normalised ( $\mathcal{R}(F_i) = 1$ ) in order to simplify the expressions involved in each attack.

### 7.2.1 Notation

Throughout the rest of this section, the following notation is adopted:

- $[A, a]$  represents the matrix produced by inserting vector  $a$  as a new column in matrix  $A$ . When the columns are related to records in a data stream, it is assumed the columns are sorted in the relative order they appear in the data stream (i.e. sorted by stream-index).
- $\overline{A}$  represents the matrix produced by concatenating the columns of  $m \times n$  matrix  $A$  to form a single column vector of length  $mn$ .
- $\delta_i$  and  $\gamma_i$  represent the vectors of independent or cumulative noise (respectively) that are components of the perturbed record  $y_i$ .
- $a$  l.i.  $A$  indicates vector  $a$  is linearly independent to the columns of matrix  $A$ .

### 7.2.2 Known Input-Output MAP Attack on Random Projection

The Maximum A Posteriori (MAP) attack has previously been proposed as a way to breach the privacy of randomly projected datasets given known input-output pairs (K. Liu, 2007; K. Liu et al., 2008). This attack requires that all columns in  $X_p$  are linearly independent (i.e.  $X_p$  has full column rank), and that the target  $x_i$  is linearly



independent to all columns in  $X_p$ . This implies that  $p$  must be less than  $m$ , as the number of rows in  $X$  limits the maximum rank of  $[X_p, x_i]$  to  $m$ . Attacks for the case when  $p \geq m$  are discussed in Section 7.2.7.

The basic premise of the attack is to estimate  $\hat{x}_i$  as  $\hat{x}$  such that the probability of  $[Y_p, y_i]$  being the result of a random projection of  $[X_p, \hat{x}]$  is maximised:

$$\hat{x}_i = \arg \sup_{\hat{x}} \phi_r(\overline{[Y_p, y_i]}) \quad (7.8)$$

where  $\hat{x} \in \mathbb{R}^m$  l.i.  $X_p$  and  $\phi_r$  is the probability density function for the output of the random projection:

$$\overline{[Y_p, y_i]} = \frac{1}{\sqrt{k}\sigma_r} \overline{R[X_p, \hat{x}]} \quad (7.9)$$

The distribution of  $\phi_r$  is a multi-variate Gaussian (of  $(p+1)k$  dimensions) with a zero mean vector and a block-diagonal covariance matrix (K. Liu, 2007):

$$\Sigma_{\phi_r} = I_k \otimes \frac{1}{k} [X_p, \hat{x}]^T [X_p, \hat{x}] \quad (7.10)$$

While finding an analytic solution to the maximisation problem in Equation (7.8) is infeasible, it is possible to find approximate solutions numerically via optimisation (K. Liu, 2007). The computational complexity of performing such an attack is dominated by the  $O(m(p+1))$  matrix multiplication of  $X^T X$ , and the  $O((k(p+1))^3)$  Cholesky decomposition and matrix inversion operations performed in serial to compute the probability density of  $\phi_r$ . As these operations must be performed once for each of  $i$  iterations of a numeric optimisation algorithm, the combined computational complexity of the attack can be simplified to  $O(i(m(p+1) + (k(p+1))^3))$ .

### 7.2.3 Extended MAP Attack for Random Translation

The above attack on random projection does not account for the additional random translation used in the RP perturbation method, which extends the attack's target perturbation model from Equation (7.9) to become:

$$\overline{[Y_p, y_i]} = \frac{1}{\sqrt{k}\sigma_r} \overline{R[X_p, \hat{x}] + \Psi} \quad (7.11)$$

However, the translation can be accounted for by translating both  $[Y_p, y_i]$  and  $[X_p, \hat{x}]$  so that one pair of corresponding input and output records are aligned at the origin (which is the zero vector:  $\mathbf{0}$ ). Because any projection of a zero vector is itself the zero vector ( $\frac{1}{\sqrt{k}\sigma_r} \overline{R\mathbf{0}} = \mathbf{0}$ ), such a pair must represent the result of a random projection without any additional translation component. Such a reversing translation can be performed on the inputs ( $[X_p, \hat{x}]$ ) and outputs ( $[Y_p, y_i]$ ) by subtracting the first column of each matrix from all other columns in that matrix. However, because the zero vector is linearly dependent to all other vectors, this pair can no longer be used as part of the attack on random projection, and therefore the first column of each matrix (now zero vectors) must be removed. This entire alignment operation is defined as the function  $\alpha$ :

$$\alpha \left( \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \dots & \dots & \dots & \dots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{bmatrix} \right) = \begin{bmatrix} a_{1,2} - a_{1,1} & \dots & a_{1,n} - a_{1,1} \\ \dots & \dots & \dots \\ a_{m,2} - a_{m,1} & \dots & a_{m,n} - a_{m,1} \end{bmatrix} \quad (7.12)$$

such that  $\alpha(A) = \alpha(A + \Psi)$  for any matrix  $A$  and translation  $\Psi$  with corresponding dimensions.

$\alpha$  can be used to update the MAP attack formulae in Equations (7.8) and (7.10) to

account for the translation introduced in Equation (7.11):

$$\begin{aligned}\hat{x}_i &= \arg \sup_{\hat{x}} \phi_r(\overline{\alpha([Y_p, y_i])}) \\ \Sigma_{\phi_r} &= I_k \otimes \frac{1}{k} \alpha([X_p, \hat{x}])^T \alpha([X_p, \hat{x}])\end{aligned}\quad (7.13)$$

This extended attack that accounts for both the random projection and random translation of the RP perturbation method will hereafter be referred to as A-RP. Removing one of the known input-output pairs reduces the number of inputs used to build the covariance matrix of  $\phi_r$  such that the computational complexity of A-RP is  $O(i(mp + (kp)^3))$ .

#### 7.2.4 Extended MAP Attack for Independent Noise

This subsection describes an attack named A-RPIN, which is an extension of A-RP that accounts for the independent noise of the RPIN perturbation method:

$$\overline{[Y_p, y_i]} = \frac{1}{\sqrt{k}\sigma_r} \overline{R[X_p, \hat{x}] + \Psi + [\Delta_p, \delta_i]} \quad (7.14)$$

A-RPIN involves two stages with separate optimisation problems.<sup>2</sup>

The first stage produces an estimate ( $\widehat{\Delta}_p$ ) of the independent noise added to each of the known records. The optimisation problem balances the probabilities of the estimated noise values and the probability that the estimated known output records without noise ( $\alpha(Y_p - \widehat{\Delta}_p)$ ) resulted from a random projection of the known inputs:

$$\widehat{\Delta}_p = \arg \sup_{\widehat{\Delta}_p} \frac{1}{kp + 1} \left( \frac{\phi_r(\overline{\alpha(Y_p - \widehat{\Delta}_p)})}{\sum \phi_\delta(\widehat{\Delta}_p)} \right) \quad (7.15)$$

where  $\widehat{\Delta}_p \in \mathbb{R}^{k,p}$  and  $\phi_\delta$  is the probability density function for an independent noise value

<sup>2</sup>An approach based on a single optimisation problem was initially experimented with, but it was found to be marginally less accurate than the two-stage method.

$(\mathcal{N}(0, \sigma_\delta^2))$ . The entire expression is divided by  $kp + 1$  to produce a mean probability density that is comparable to the outputs of the other attack types (such as A-RP). Also, because there is no estimate for  $\hat{x}$  in this problem, the covariance matrix for  $\phi_r$  is simplified to:

$$\Sigma_{\phi_r} = I_k \otimes \frac{1}{k} \alpha(X_p)^T \alpha(X_p) \quad (7.16)$$

The second stage of A-RPIN simultaneously optimises for the estimated record  $\hat{x}_i$ , and the independent noise that was applied to it ( $\hat{\delta}_i$ ). This optimisation problem utilises the  $\widehat{\Delta}_p$  estimated in the first stage, and balances the probability of generating the noise vector  $\hat{\delta}$  with the probability that the estimated outputs without noise ( $\alpha([Y_p, y_i] - [\widehat{\Delta}_p, \hat{\delta}])$ ) resulted from a random projection of the inputs:

$$\hat{x}_i, \hat{\delta}_i = \arg \sup_{\hat{x}, \hat{\delta}} \frac{1}{2} \left( \phi_r(\alpha([Y_p, y_i] - [\widehat{\Delta}_p, \hat{\delta}])) + \frac{1}{k} \sum \phi_\delta(\hat{\delta}) \right) \quad (7.17)$$

where  $\hat{x} \in \mathbb{R}^m$  i.i.  $X_p$  and  $\hat{\delta} \in \mathbb{R}^k$ . Once again, the probability densities are reduced to a mean density, but here the density of  $\phi_r$  is given equal weight to the combined mean of all  $\phi_\delta$ . Initial experimentation found that balancing the densities in this way improved the accuracy of the attack.

A special consideration must be made when  $p = 1$ , because this will cause  $\alpha(X_p)$  in Equation (7.15) to be an empty matrix. In this case, the first stage of the attack is skipped and  $\widehat{\Delta}_p$  is estimated as a zero vector. However, this will result in the independent noise still being present on the single known record during the second stage. To compensate for this, the variance of the distribution  $\phi_\delta(\hat{\delta})$  can be doubled to  $2\sigma_\delta^2$ , which will account for the independent noise added to both the unknown and single known record.

As A-RPIN involves two numeric optimisation problems that are dominated by the same operations as A-RP, its computational complexity can be expressed as  $O(i_1(pm +$

$(kp)^3) + i_2(pm + (kp)^3))$ , where  $i_1$  and  $i_2$  are the number of iterations for each numeric optimisation.

### 7.2.5 Extended MAP Attack for Cumulative Noise

This subsection adapts A-RPIN to produce A-RPCN, which is a MAP attack that can account for the perturbation model of RPCN:

$$\overline{[Y_p, y_i]} = \frac{1}{\sqrt{k}\sigma_r} \overline{R[X_p, \hat{x}] + \Psi + [\Gamma_p, \gamma_i]} \quad (7.18)$$

To facilitate the description of A-RPCN, a new matrix  $\Omega_p$  is defined that represents the differences in cumulative noise between successive columns in  $Y_p$ . Therefore, each column in  $\Gamma_p$  can be reconstructed by summing the corresponding column in  $\Omega_p$  and all columns prior to it:

$$\Gamma_p = \begin{bmatrix} \omega_{1,1} & \sum_{i=1}^{i \leq 2} \omega_{1,i} & \dots & \sum_{i=1}^{i \leq p} \omega_{1,i} \\ \dots & \dots & \dots & \dots \\ \omega_{k,1} & \sum_{i=1}^{i \leq 2} \omega_{k,i} & \dots & \sum_{i=1}^{i \leq p} \omega_{k,i} \end{bmatrix} \quad (7.19)$$

As with A-RPIN, A-RPCN is split into two stages. In the first stage, the cumulative noise ( $\Gamma_p$ ) is estimated similarly to the independent noise:

$$\widehat{\Omega}_p = \arg \sup_{\widehat{\Omega}_p} \frac{1}{kp+1} \left( \phi_r(\overline{\alpha(Y_p - \widehat{\Gamma}_p)}) + \sum \phi_\omega(\widehat{\Omega}_p) \right) \quad (7.20)$$

where  $\widehat{\Omega}_p \in \mathbb{R}^{k,p}$ . The key difference is in the use of the probability density function for the distribution of cumulative noise differences between known records:

$$\phi_\omega \sim \mathcal{N}(0, (i-h)\sigma_\gamma^2) \quad (7.21)$$

where  $i$  is the stream-index of the record the noise is being estimated for, and  $h$  is

the stream-index of the previous record in  $X_p$ . Note that the stream-indexes represent the positions of records within the entire stream, not within  $X_p$ . When  $i$  is the first column in  $X_p$  (i.e.  $h$  is not defined), the noise is always fixed to a zero vector with a probability of 1. This is done because the cumulative noise on the first known record can be considered a part of the random translation, which is already accounted for by the  $\alpha$  transformation. This also avoids the need for special handling of the case when  $p = 1$ , as was required for A-RPIN.

$\widehat{\Gamma}_p$  can then be derived from  $\widehat{\Omega}_p$  (using Equation (7.19)) for use in the second stage of the attack, which simultaneously optimises  $\hat{x}_i$  and  $\hat{\gamma}_i$ :

$$\hat{x}_i, \hat{\gamma}_i = \arg \sup_{\hat{x}, \hat{\gamma}} \frac{1}{2} \left( \phi_r(\alpha([Y_p, y_i] - [\widehat{\Gamma}_p, \hat{\gamma}])) + \frac{1}{k} \sum \phi_\gamma(\hat{\gamma}) \right) \quad (7.22)$$

where  $\hat{x} \in \mathbb{R}^m$  i.i.  $X_p$  and  $\hat{\gamma} \in \mathbb{R}^k$ . Equation (7.7) can be used to derive the probability density function ( $\phi_\gamma$ ) for the distribution of  $\hat{\gamma}_i$  given the cumulative noise estimates for the known records ( $\widehat{\Gamma}_p$ ):

$$\begin{aligned} \phi_\gamma &\sim \mathcal{N}(\mu, \sigma^2) \\ \mu &= \frac{\hat{\gamma}_h(j-i) + \hat{\gamma}_j(i-h)}{j-h} \\ \sigma^2 &= \frac{(i-h)(j-i)\sigma_\gamma^2}{j-h} \end{aligned} \quad (7.23)$$

where  $h$  and  $j$  are the stream-indices of the prior and subsequent records (respectively) relative to  $i$  in  $X_p$ . If  $i$  is earlier in the stream than all records in  $X_p$ , then  $\phi_\gamma \sim \mathcal{N}(\hat{\gamma}_j, (j-i)\sigma_\gamma^2)$ . Similarly, if  $i$  is later in the stream than all records in  $X_p$ , then  $\phi_\gamma \sim \mathcal{N}(\hat{\gamma}_h, (i-h)\sigma_\gamma^2)$ .

As the complexity of A-RPCN is dominated by the same matrix operations as A-RPIN, its computational complexity can also be expressed as  $O(i_1(pm + (kp)^3) + i_2(pm + (kp)^3))$ , where  $i_1$  and  $i_2$  are the number of iterations for each numeric optimisation.

## 7.2.6 Numerical Optimisation

As mentioned previously, numerical optimisation can be used to find approximate solutions for Equations (7.8), (7.13), (7.15), (7.17), (7.20), and (7.22). The Nelder-Mead method was used for this purpose (Nelder & Mead, 1965), as it is a nonlinear, unconstrained, optimisation algorithm that has been used previously for known input-output MAP attacks on random projection (K. Liu, 2007).

The Nelder-Mead method requires seeds for all variables to be optimised. For  $\hat{\delta}$ ,  $\hat{\omega}$ , and  $\hat{\gamma}$  variables, random Gaussian values are generated according to the distributions of  $\phi_{\delta}$ ,  $\phi_{\omega}$ , and  $\phi_{\gamma}$  respectively. Initial values for  $\hat{x}$  are randomly generated from a uniform distribution with a range equal to that of each feature centred around the median of each feature (where the range and median are estimated from the known inputs  $X_p$ ). Except where noted, experiments in Chapter 8 perform optimisation runs with three different random seeds for each attack, and the result that produced the highest probability density is taken as the overall winner.

The implementations of the optimisation problems as objective functions used for experimentation return probability densities in logarithmic space, which prevents underflow issues when working with very small probabilities. Furthermore, every individual log-probability-density produced during the evaluation of an objective function is limited to a maximum value of 10,000, which prevents any single value from overly skewing the optimisation (such as in cases where a density is computed as positive infinity). The maximum number of iterations was set to 200 times the number of variables to optimise, and the relative score threshold for ending the optimisation early was set to  $0.0001^3$ . No absolute score threshold was set, as probability densities were produced at different orders of magnitude depending on the number of variables

---

<sup>3</sup>Both of these are the same configuration values used in the default configuration of SciPy's Nelder-Mead optimisation implementation: <https://docs.scipy.org/doc/scipy/reference/optimize.minimize-neldermead.html>.

to optimise.

It is possible that the optimisation process may produce a matrix of inputs ( $\alpha([X_p, \hat{x}])$ ) that does not fulfil the constraint that all columns be linearly independent (i.e. the matrix does not have full column rank). These variable combinations were penalised by causing the optimisation objective function to return negative infinity as the log-probability-density.

### 7.2.7 Attacks When $p \geq m$

As previously mentioned, all of the above attacks require that  $[X_p, x_i]$  has full column rank, so they cannot be used with more known records than there are features in the original data stream ( $p \geq m$ ). This raises the question, what attacks are possible when this is the case?

K. Liu et al. (2008) show that when at least  $m$  input-output pairs are known for a randomly projected dataset, any other record  $x_i$  must be linearly dependent on  $X_p$ , and therefore can be perfectly recovered via a linear combination of  $X_p$ . They also claim that an attacker can know when  $x_i$  is linearly dependent on  $X_p$ , because  $y_i$  will be linearly dependent on  $Y_p$ . However, this is not always true. Consider the case when  $[X_p, x_i]$  has full column rank. If  $k < p$ , then it is not possible for  $[Y_p, y_i]$  to also have full column rank. This shows that even though  $y_i$  will be linearly dependent on  $Y_p$  when  $x_i$  is linearly dependent on  $X_p$ , the opposite is not always true. Therefore, an attacker cannot be certain  $x_i$  is linearly dependent on  $X_p$  on the basis of their known input-output pairs.

On the other hand, K. Liu (2007) shows that it is possible to recover the random projection matrix  $R$  when  $p \geq m$ . Even though this does not allow any value in the original matrix to be perfectly recovered, Lyu et al. (2018) present an alternative formulation of the MAP optimisation problem that makes use of prior knowledge of  $R$ .



However, in either of the above cases, independent and cumulative noise will distort the relationship between the column ranks of  $[X_p, x_i]$  and  $[Y_p, y_i]$ , confounding attempts to perfectly recover records via a linear combination or to recover the projection matrix  $R$ . On this basis, more sophisticated attacks against RPIN and RPCN in cases when  $p \geq m$  can be considered an area for further research.

## Chapter 8

# Experimental Evaluation of Proposed Perturbation Methods

The following section details experimental evaluations of the proposed known input-output attacks and the three perturbation methods they apply to: random projection with random translation (RP), and the two extensions with independent noise (RPIN) and cumulative noise (RPCN) respectively. The effect cumulative noise has on accuracy and privacy over the lifetime of a data stream is also analysed. The implementations used in the experiments were implemented with Clojure and made use of the MOA framework (Bifet et al., 2010) for stream mining algorithms and Apache Commons Math 3.6.1<sup>1</sup> for Nelder-Mead optimisation. The source-code (including data preparation and experimentation) has been made available in a GitHub repository<sup>2</sup>. All experiments were performed with OpenJDK 1.8.0 on a 64-bit Ubuntu 16.04 installation running on a 4x2.60GHz Intel Core i5 CPU with 8GB of memory.

---

<sup>1</sup><http://commons.apache.org/proper/commons-math/>

<sup>2</sup><http://github.com/ben-denham/ppdsp>

## 8.1 Experimental Setup

Online classification is used as the setting for evaluating the impact of data perturbation on data utility; specifically, how it impacts classification accuracy. Accuracy is measured by applying the adaptive random forest (ARF) classifier (Gomes et al., 2017) in the setting of prequential evaluation (Bifet & Kirkby, 2009). The implementation of ARF provided by the MOA data mining framework is used (Bifet et al., 2010).

The measures of relative error and  $\varepsilon$ -privacy (K. Liu, 2007) are used to evaluate the efficacy of privacy-preservation methods. Relative error represents the degree of success achieved by a record recovery attempt. It is defined as the magnitude of the difference vector between the original record ( $x_i$ ) and its recovered counterpart ( $\hat{x}_i$ ), normalised by the magnitude of the original record vector:  $\frac{\|x_i - \hat{x}_i\|}{\|x_i\|}$ . An “ $\varepsilon$ -privacy breach” of a record occurs if the relative error of the recovered record is less than a specified  $\varepsilon$ . The probability of an  $\varepsilon$ -privacy breach with a known input-output attack under certain conditions is measured by performing a series of attacks using different sets of randomly selected “known” records to attempt to recover randomly selected “unknown” records. The proportion of attacks that resulted in an  $\varepsilon$ -privacy breach is then taken as the probability of such a breach.

Attack duration is measured in terms of CPU execution time. In order to account for JVM warm-up, all experiments where duration is measured were repeated immediately within the same process, with the duration of the second set of experiments being recorded.

As all three perturbation methods under evaluation are based on random projection, the random projection parameters are kept constant throughout all experiments. Specifically, data dimensionality is not reduced ( $k = m$ ), and  $\sigma_r$  is set to 1. Furthermore, as all known input-output attack types are able to perfectly remove the random translation component of the perturbation methods, the effect of random translation does not need

to be experimentally evaluated. The key comparison is of the effects of adding independent or cumulative noise to random projection. Therefore, no translation is applied during the experimentation in order to reduce the impact of numerical imprecision resulting from the additional floating point arithmetic operations.

## 8.2 Datasets

A variety of datasets that can be considered streams and/or containing sensitive data were selected for the experimental evaluation. Table 8.1 describes the properties of each of these twelve datasets. To ensure all tested levels of noise represented comparable amounts of distortion across datasets, all dataset features were min-max normalised to a range of  $[0, 1]$ .

Table 8.1: Properties of datasets used for experimental evaluation.

Dataset	Feature Count	Class Count	Record Count	Real-world?	Stream?	Private?
SEA	3	2	100,000	No	Yes	No
RBF	10	5	50,000	No	Yes	No
ELEC	8	2	45,312	Yes	Yes	No
WFR	4	4	5,456	Yes	Yes	No
AREM	6	32	35,999	Yes	Interleaved	One Individual
TAXI	7	3	50,000	Yes	Yes	Many Individuals
POWUSG	10	3	19,735	Yes	Yes	One Individual
P2PLNS	10	2	12,682	Yes	Yes	Many Individuals
PREG	5	2	4,082	Yes	Yes	Many Individuals
BRCNCR	9	2	10,000	Yes	No	Many Individuals
ADULT	6	2	32,561	Yes	No	Many Individuals
HTRU2	8	2	17,898	Yes	No	No

Eight of the datasets represent real-world examples of data streams: Electricity (ELEC; Gama, Medas, Castillo and Rodrigues (2004)), Wall-Following Robot Navigation (WFR; Freire et al. (2009)), Activity Recognition system based on Multisensor

data fusion (AREM; Palumbo, Gallicchio, Pucci and Micheli (2016)), New York City Taxi Trip Duration (TAXI; Kaggle (2017)), Individual Household Electric Power Consumption (POWUSG; Hebrail (2012)), Prosper Peer-to-Peer Loans (P2PLNS; Prosper Marketplace, Inc. (2014))<sup>3</sup>, and 2002 Pregnancy survey data (PREG; Centers for Disease Control and Prevention (2005)). These datasets are considered streams because their records can be ordered according to the time they were produced, though the presence or nature of concept drift within these datasets is unknown. Feature sets were reduced to a subset of numeric features for the TAXI, POWUSG<sup>4</sup>, P2PLNS, and PREG datasets. Two datasets also required the creation of a classification target by applying equal-frequency binning to a target feature: TAXI (trip duration) and POWUSG (power usage amount). Two datasets were sub-sampled without replacement to achieve balance between classes, which allowed classification accuracy to be used as a valid performance measure: P2PLNS (completed vs. defaulted and charged-off loans) and PREG (live-birth vs. still-birth and miscarriage). For testing efficiency, only the first 50,000 records of the TAXI dataset were used. Finally, all values of the “parity” feature in the PREG dataset were decremented by 1 to remove knowledge of the outcome of the pregnancy represented by each record.

The SEA (Street & Kim, 2001) and radial basis function (RBF; Bifet and Kirkby (2009)) synthetic stream generators were selected as they have known concept drift properties and were used in the original experimental evaluation of the ARF classifier (Gomes et al., 2017). The SEA stream was generated with three drift points at 25,000 record intervals and an abrupt rate of drift (drift-width = 1). The RBF stream was generated with 5 classes and 50 centroids drifting at a fast rate (speed-of-change = 0.001). The MOA implementations of SEA and RBF were utilised (Bifet et al., 2010).

Testing was also performed with static (non-streaming) datasets: Breast Cancer

<sup>3</sup>Accessing the original Prosper loan data is described at: <https://prosper.zendesk.com/hc/en-us/articles/210013083-Where-can-I-download-Prosper-loan-data->.

<sup>4</sup>Only temperature values were used.

Wisconsin (BRCNCR; Mangasarian, Street and Wolberg (1995)), ADULT (Kohavi, 1996), and HTRU2 (R. Lyon, 2017; R. J. Lyon, Stappers, Cooper, Brooke & Knowles, 2016)). ADULT and HTRU2 have been used in previous studies on data perturbation (Lyu et al., 2018; Okkalioglu et al., 2015). The ADULT dataset was limited to its six numeric features, as has previously been performed for studying numeric data perturbation (Guo, Wu & Li, 2008), and BRCNCR was increased from 699 to 10,000 records via the Synthetic Minority Over-sampling Technique (SMOTE; Chawla, Bowyer, Hall and Kegelmeyer (2002)) in order to produce enough records for mining in an online setting.

WFR, AREM, POWUSG, BRCNCR, ADULT, and HTRU2 were retrieved from the UCI Machine Learning Repository (Dheeru & Karra Taniskidou, 2017).

Many of the datasets described above represent potentially sensitive data, either because each record within the dataset represents personal information relating to an individual (TAXI, P2PLNS, PREG, BRCNCR, ADULT), or because the stream may come from a sensor monitoring an individual or group of individuals (AREM and POWUSG). For example, consider the case of the TAXI data stream, where each record represents a single taxi trip. Each record includes the start time and the coordinates of the origin and destination of the trip. For each individual who took a taxi trip, the record may reveal private information, particularly if the origin is their home address and the destination is sensitive (such as a medical surgery). Furthermore, this data stream would be considered private by the taxi company that collected it, as it reveals the operational behaviours of their taxi service. Obtaining such data could provide a competitive advantage to a rival company. However, taxi companies may wish to share such data streams with each other in order to improve their respective models for predicting trip duration, providing mutual benefit for both companies. This makes the taxi trip data stream environment a prime candidate for privacy-preserving data sharing. However, a known input-output attack is highly likely, as an individual or group

of individuals may attempt to use their knowledge of their own taxi trips to recover knowledge of the trips of other individuals.






### 8.3 Attack Type Comparison

Before the three perturbation methods (RP, RPIN, and RPCN) can be compared, the attack type that will achieve the best record recovery against each method must be established. The best attack types can then be used as benchmarks when comparing the privacy achieved by each perturbation method.

For the RP perturbation method based on random projection and translation, the A-RP known input-output attack described in Section 7.2.3 is the only applicable known input-output attack, and a comparison of attack types for this perturbation method is not necessary. However, when the independent noise (RPIN) or cumulative noise (RPCN) methods are used, there are several possible attack types. One option is to use the respective attacks that account for independent noise (Section 7.2.4; A-RPIN) or cumulative noise (Section 7.2.5; A-RPCN). Alternatively, as the expected mean value is zero for either form of additive noise in the absence of additional prior knowledge, A-RP may still be a viable approach when additive noise is present. Furthermore, initial experimentation showed that the relative error of records recovered by A-RPIN and A-RPCN tends to increase with the number of known input-output pairs. This may be due to the rapidly increasing numbers of variables in the optimisation problems of the first stage in each attack, which grow with  $O(kp)$ . Therefore, variations of A-RPIN and A-RPCN that only use one known input-output pair were also tested (A-RPIN-1 and A-RPCN-1). When more than one pair is available in an experiment, the closest record in the stream to the unknown record is used. Finally, as initial experimentation showed that A-RP and the attack types that take noise into consideration outperform each other under different conditions (e.g. depending on the amount of additive noise or the number

of known input-output pairs), the performance of combinations of A-RP with the other attack types was also evaluated. These combinations are represented as MAX(A-RP, A-RPIN), MAX(A-RP, A-RPIN-1), MAX(A-RP, A-RPCN), and MAX(A-RP, A-RPCN-1) respectively. To combine two attack types, each attack is applied separately, and the attack that achieves the highest objective function score is taken as the overall winner. If an attack type involves multiple stages, the score of the final stage's objective function is compared. The scores produced by different objective functions are comparable as they are all normalised to represent a mean probability density. Table 8.2 summarises the full list of attack types to be compared for perturbation methods RPIN and RPCN.

Table 8.2: Attack types evaluated for each perturbation method.

Perturbation	Attack Types				
					
RPIN	A-RP	A-RPIN	A-RPIN-1	MAX(A-RP, A-RPIN)	MAX(A-RP, A-RPIN-1)
RPCN	A-RP	A-RPCN	A-RPCN-1	MAX(A-RP, A-RPCN)	MAX(A-RP, A-RPCN-1)

The attack types were compared according to their effectiveness in recovering the original records from perturbed versions of the TAXI dataset (experiments with other datasets presented similar trends, and are therefore omitted for brevity). For each perturbation method, three perturbed datasets were produced with different levels of noise. For independent noise, the three levels of noise were achieved by setting  $\sigma_\delta$  to: 0.05, 0.1, and 0.25. The cumulative noise levels were achieved by setting  $\sigma_\gamma$  to values that would achieve equivalent levels of noise according to Equation (7.6) (given the number of records in the TAXI stream):  $\sim 3.4E - 4$ ,  $\sim 6.7E - 4$ , and  $\sim 1.7E - 3$ . Additionally, experiments were performed with different numbers of known input-output pairs ( $p$ ): 1, 4 ( $\lceil \frac{m}{2} \rceil$ ), and 6 ( $m - 1$ ). 500 attacks were simulated with each attack type for each combination of noise level and  $p$ , and these attack simulations were used



to estimate the probability of an  $\varepsilon$ -privacy breach ( $\varepsilon = 0.2$ ).

Figure 8.1 shows the results of the experiments with different attack types on RPIN and RPCN perturbation methods. Note that all attack types on both RPIN and RPCN other than A-RP achieve identical or nearly identical performance when  $p = 1$  (these points overlap in the charts). This is due to the fact that the known record limit of A-RPIN-1 and A-RPCN-1 makes no difference when only one record is known. Furthermore, when  $p = 1$ , the combined attack types rely solely on the attacks that account for additive noise because A-RP produces lower probability density scores. It can be seen that as  $p$  increases, the effectiveness of each attack increases initially, but may decrease slightly at the maximum value of  $p$ . This is likely due to the increased difficulty in optimising the objective functions that involve many more variables. For both RPIN and RPCN perturbation, A-RP is the most effective attack type at lower noise levels. However, the combined attack types MAX(A-RP, A-RPIN-1) and MAX(A-RP, A-RPCN-1) are generally found to be the second most effective attack types for their respective perturbation methods, and they improve in performance with increasing noise levels such that they are the most effective attacks at the highest noise levels. Because high noise levels represent the most challenging scenarios to attack, MAX(A-RP, A-RPIN-1) and MAX(A-RP, A-RPCN-1) were selected as the best attack types to use for privacy evaluation in subsequent experiments.

### 8.3.1 Attack Execution Time Comparison

As the established computational complexity of each attack type is dependent on the number of iterations performed during one or two numeric optimisations, the execution time of each attack type (excluding combined types) was also evaluated experimentally. A total of 500 attacks on randomly selected records were made against the TAXI dataset at the highest noise level, with a single set of initial optimisation variables. Figure 8.2

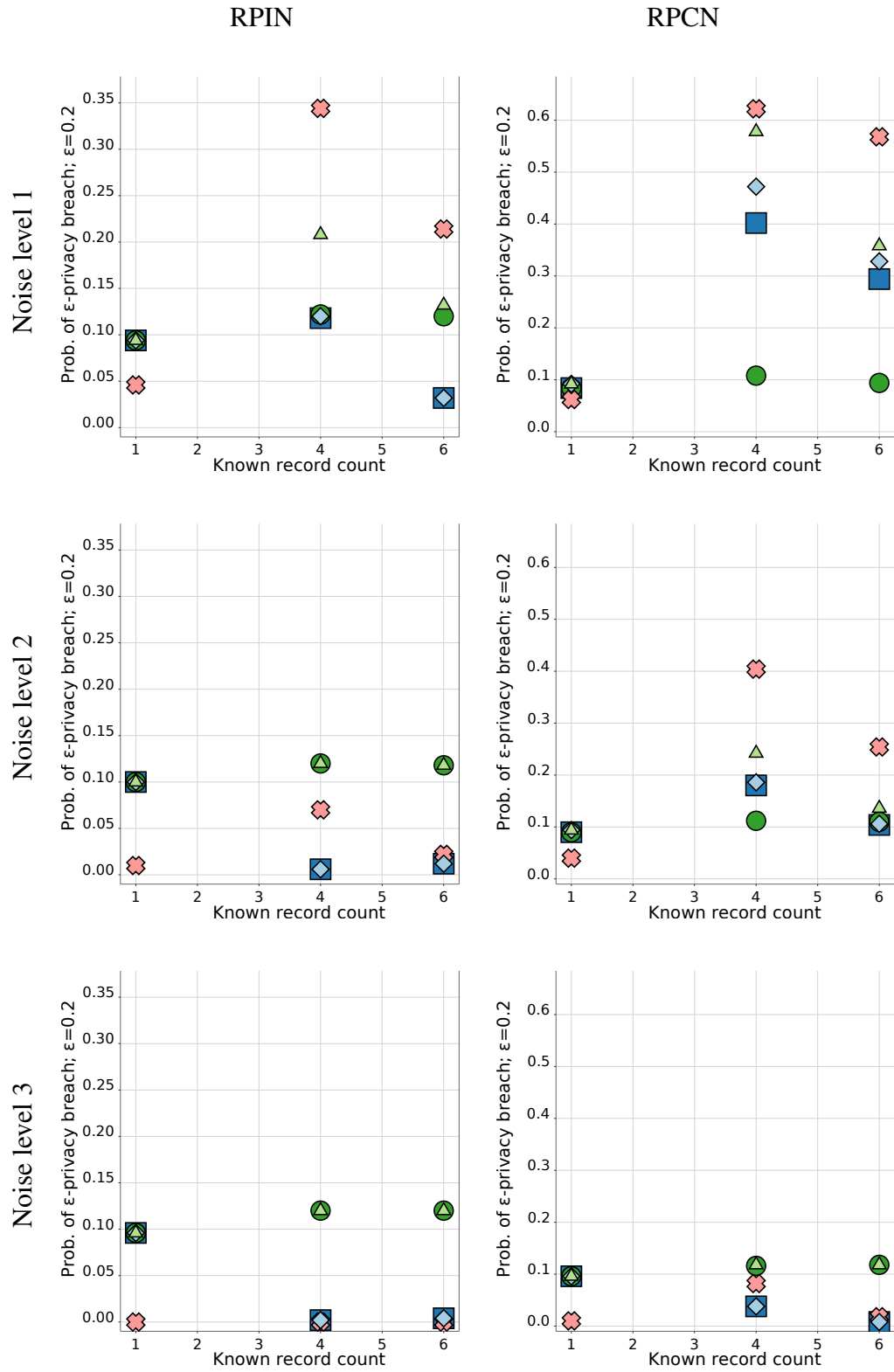


Figure 8.1: Comparison of MAP attacks on perturbation with RPIN and RPCN applied to the TAXI dataset. Legend in Table 8.2.

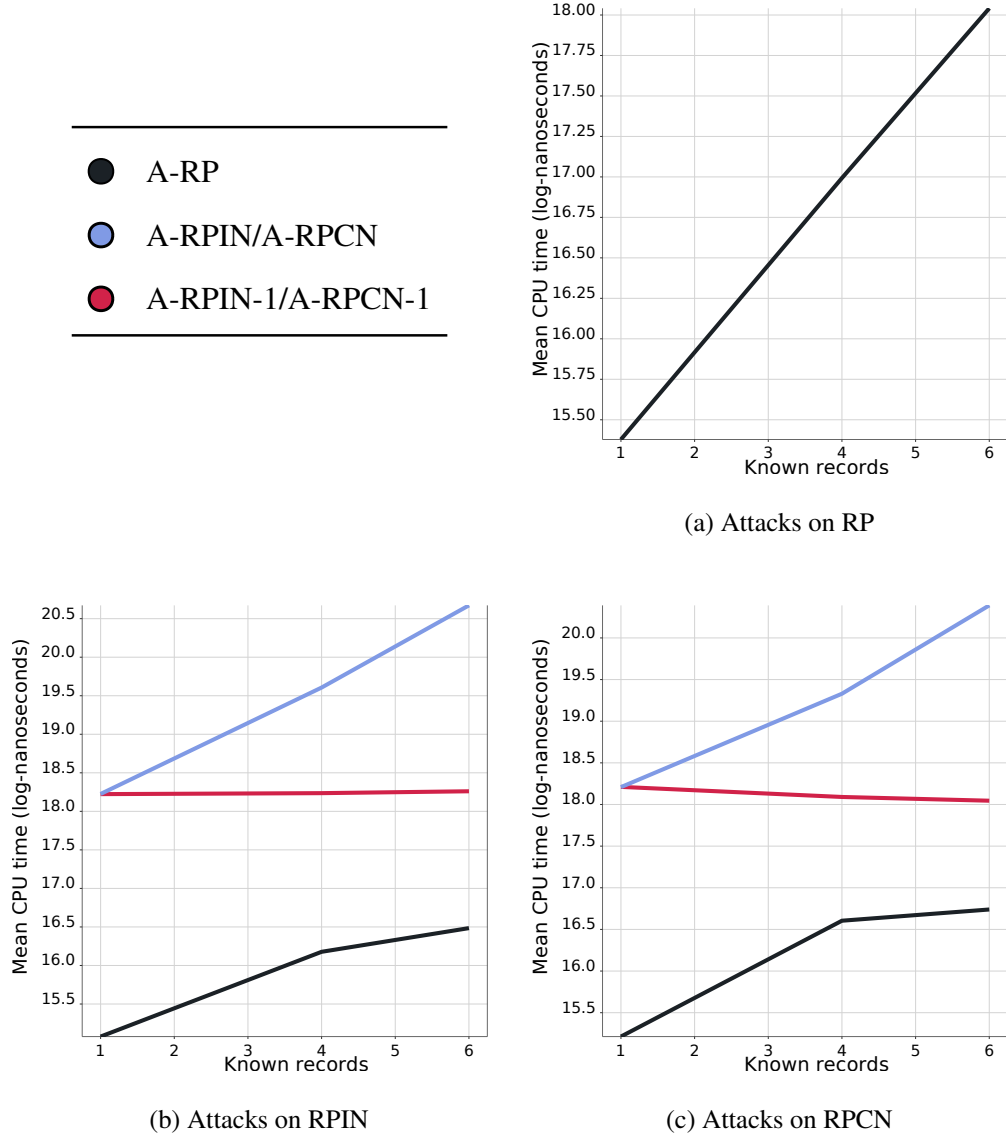


Figure 8.2: Comparison of the execution time of MAP attacks on the TAXI dataset.

(where the y-axis is a log scale) shows that the mean time (taken over 500 attacks) required for all attack types appears to increase either polynomially or exponentially with  $p$  (with the exception of A-RPIN-1 and A-RPCN-1, which only ever use a single known input-output pair;  $p = 1$ ). Therefore, even if an attacker has more known input-output pairs, they may not be able to improve the effectiveness of their attacks if they are attempting to breach the privacy of a stream in real-time.

## 8.4 Perturbation Method Comparison

After establishing the benchmark privacy attacks, the three perturbation methods (RP, RPIN, and RPCN) were compared across all twelve datasets. Experiments were performed with combinations of three noise levels and three values of  $p$  (the number of known input-output pairs). The three independent noise levels used  $\sigma_\delta$  values equal to 0.05, 0.1, and 0.25, and the cumulative noise levels used the corresponding  $\sigma_\gamma$  to achieve equivalent levels of noise according to Equation (7.6) (given the number of records in each dataset). Tests were performed with 1,  $\lceil \frac{m}{2} \rceil$ , and  $m - 1$  known input-output pairs (where  $m$  is the number of features in each dataset).

To evaluate privacy, attacks were performed on 500 randomly chosen records for each perturbed dataset using the corresponding benchmark attack: A-RP for RP, MAX(A-RP, A-RPIN-1) for RPIN, and MAX(A-RP, A-RPCN-1) for RPCN. The probability of an  $\varepsilon$ -privacy breach was evaluated for  $\varepsilon$  values of 0.2 (representing a recovery of a broad estimate of the original value) and 0.1 (representing an accurate recovery of the original value). Accuracy was measured by attempting to perform online classification on each perturbed data stream with an ARF classifier using Naive Bayes leaf prediction in a prequential classify-then-train context.

Figures 8.3, 8.4, and 8.5 present the trade-off of the resistance of each perturbation method to privacy-breaching attacks against the accuracy that can be achieved when learning from the perturbed data (legend in Table 8.3).

Table 8.3: Perturbation method legend for Figures 8.3, 8.4, and 8.5.

Noise	RP	RPIN	RPCN
None	●		
Level 1		○	●
Level 2		△	△
Level 3		□	■

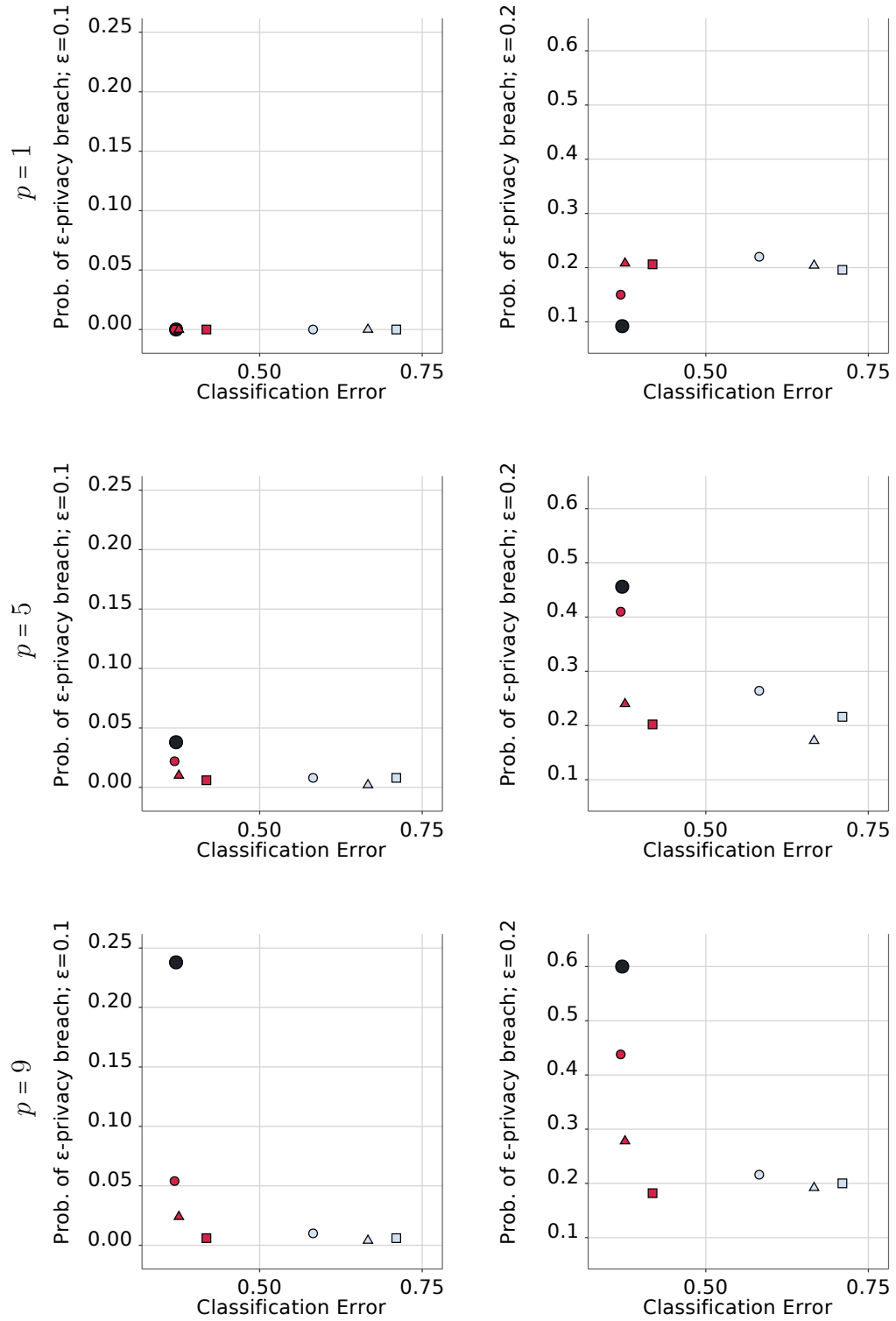


Figure 8.3: Trade-off between accuracy and privacy for different perturbation methods applied to the RBF dataset.

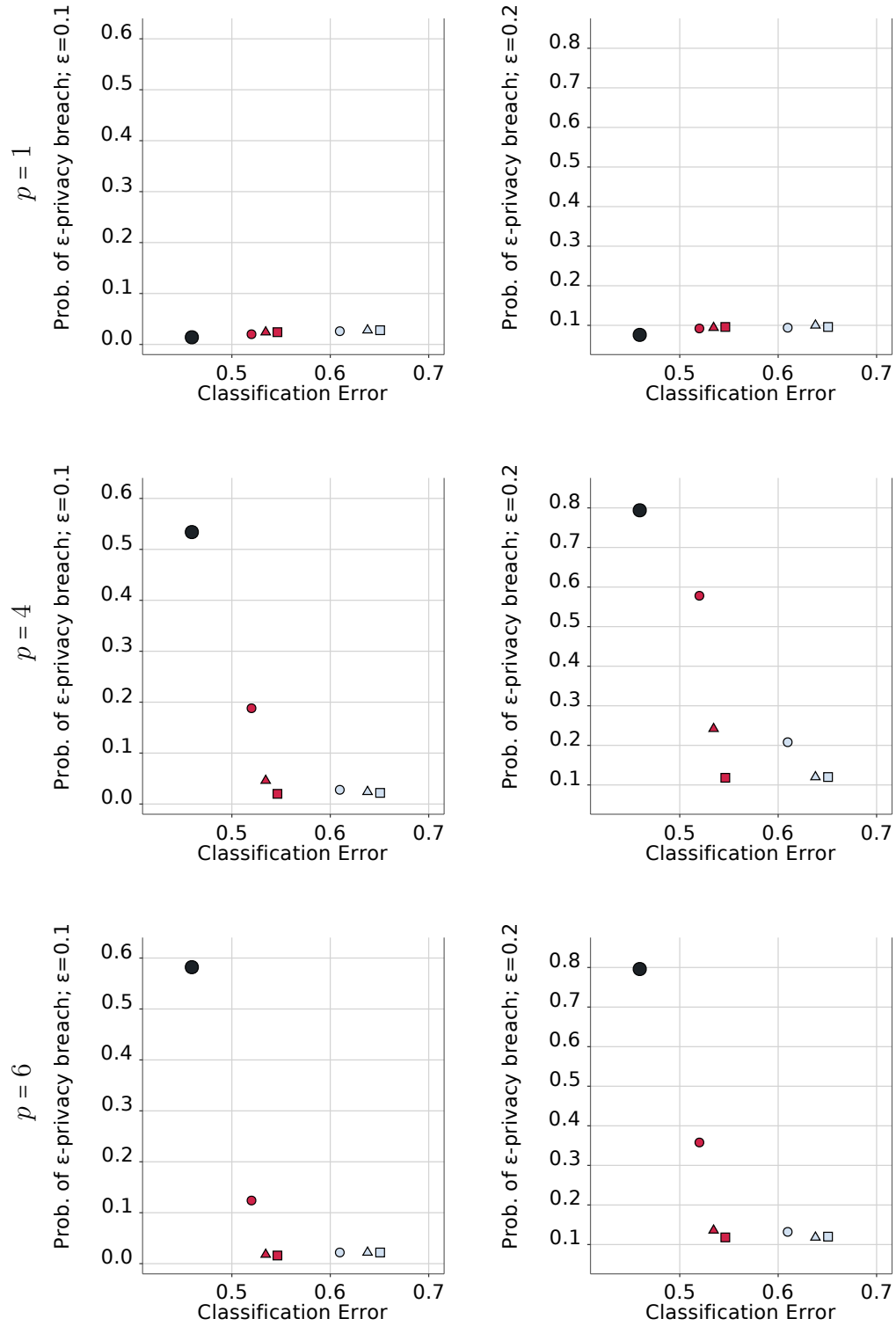


Figure 8.4: Trade-off between accuracy and privacy for different perturbation methods applied to the TAXI dataset.

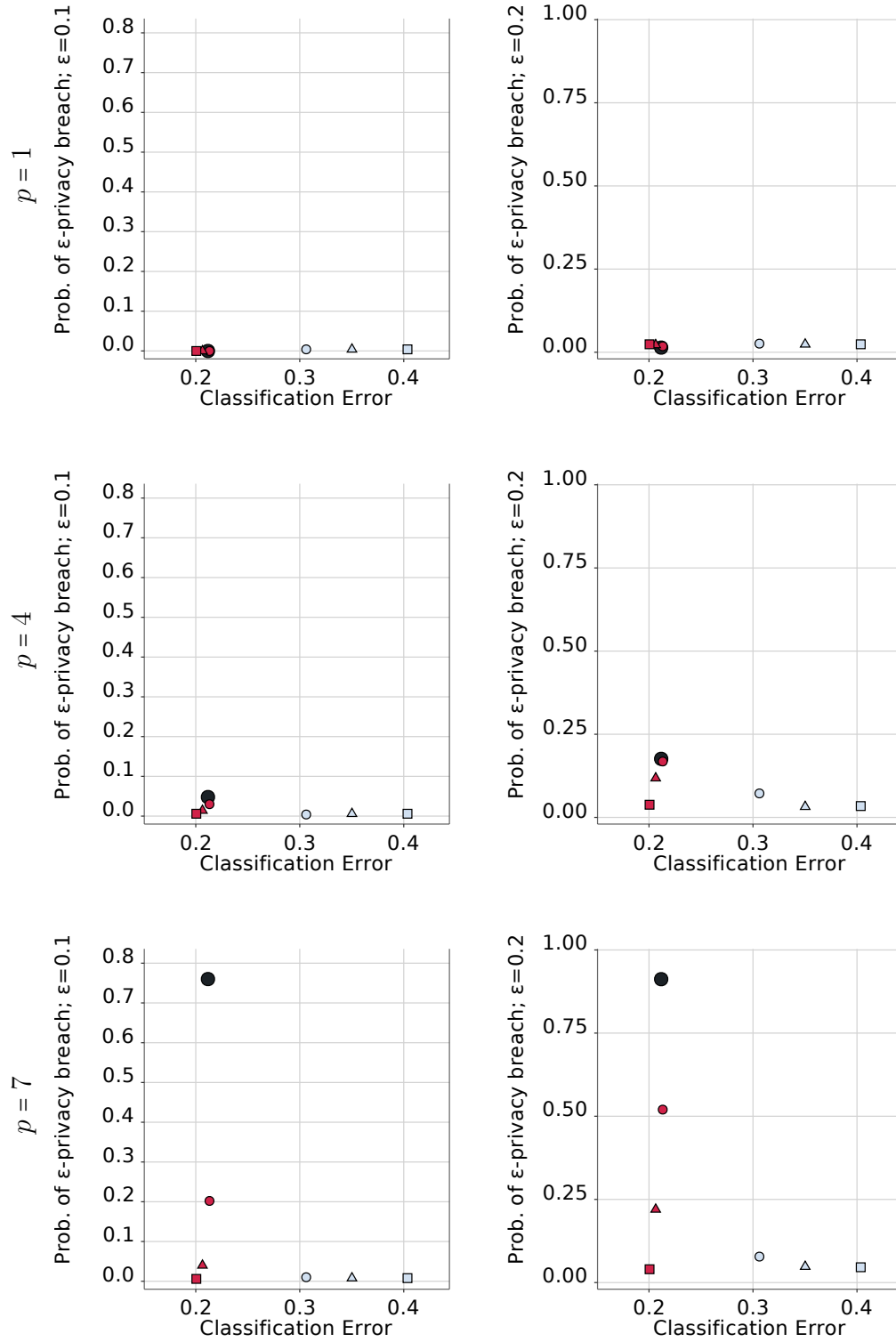


Figure 8.5: Trade-off between accuracy and privacy for different perturbation methods applied to the ELEC dataset.

Points closer to the origin represent perturbation methods that achieve better accuracy and privacy, so it can be seen that as the level of either type of additive noise increases, the privacy generally improves at the expense of accuracy. A consistent trend that is observed across all three datasets is that RP performs poorly in comparison to both perturbation methods that use additive noise (RPIN and RPCN) when the number of known input-output pairs is greater than 1. These are the scenarios of greatest interest in practice, as they challenge the robustness of the data perturbation mechanism. Furthermore, this decrease in privacy when only one input-output pair is known (shown in the top bands of Figures 8.3-8.5) is much smaller than the privacy gain when more pairs are known. Another important and consistent trend observed across Figures 8.3-8.5 is the clear superiority of RPCN over RPIN as it achieves a far better privacy/accuracy trade-off (as its points are much closer to the origin). The advantage of RPCN over RPIN is its higher accuracy at any given level of privacy, as shown most clearly in the bottom bands of Figures 8.3-8.5. RPCN is able to maintain higher levels of accuracy because its noise injection is gradual, unlike RPIN. This gradual injection of noise enables ARF to adapt its model over time to mitigate against noise and thereby return higher levels of accuracy than RPIN. The trends presented in these figures are representative of the trends observed in other datasets, which have been omitted for brevity. Also note that these trends are consistent for both tested values of  $\epsilon$ .

In order to identify statistically significant differences in the performance of the tested perturbation methods with varying noise levels, a Friedman test and Nemenyi post-hoc analysis were performed in a manner similar to that recommended for comparing classifier performance (Demšar, 2006). To evaluate the trade-off between privacy and accuracy, a comparison was made based on the sum of the squares of classification error and the probability of an  $\epsilon$ -privacy breach, defined as the Privacy-Accuracy Magnitude (PAM):



$$\text{PAM} = (\text{error})^2 + P(\varepsilon\text{-privacy breach})^2 \quad (8.1)$$

Performing a sum-of-squares favours methods that balance both privacy and accuracy over methods that achieve one at the expense of the other. Attacks were evaluated for  $\varepsilon = 0.2$  and  $p = m - 1$  (the highest number of known pairs, which is the most difficult scenario for preserving privacy of those tested). The null-hypothesis that “all seven perturbation methods achieve equivalent PAM” was rejected at the 99% confidence level. The Nemenyi post-hoc analysis was then applied to find the statistically significant differences between different perturbation methods. A critical difference of  $\sim 2.60$  was found, producing the critical difference diagram in Figure 8.6 (where noise levels are appended to the names of perturbation methods). This analysis demonstrates that RP represents the worst trade-off between privacy and accuracy. RPCN-2 provided the best trade-off, although it was not significantly better than RPCN-3 and RPIN-1. This demonstrates that RPCN is the most effective method overall for balancing privacy and accuracy.

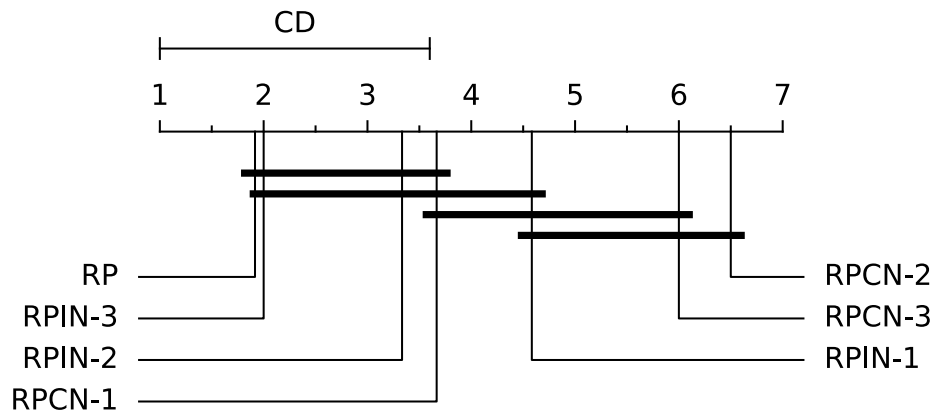


Figure 8.6: Critical difference diagram for the accuracy/privacy trade-off evaluation of perturbation methods at different noise levels.

## 8.5 Trend Analysis for Cumulative Noise Perturbation

Given that the cumulative noise of the RPCN perturbation method continues to grow over time, it is important to understand the trends of how privacy and accuracy are expected to be affected over the lifetime of a stream. Figures 8.7 and 8.8 visualise these trends for the RBF, TAXI, and ELEC datasets, which are representative of the trends in other datasets (omitted for brevity).

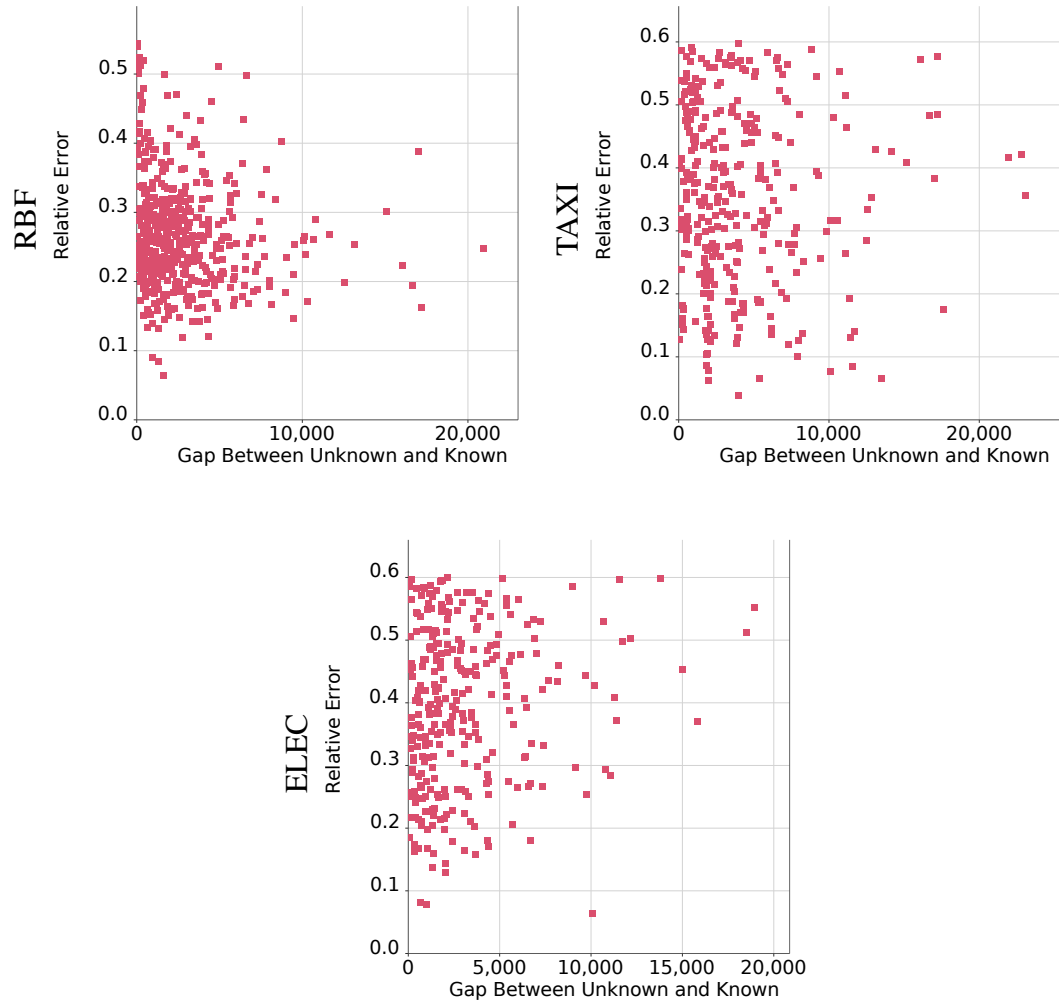


Figure 8.7: Trends of record recovery as the distance between known and unknown records increases in a data stream perturbed with RPCN.

Figure 8.7 plots the results of 500 MAX(A-RP, A-RPCN-1) attacks<sup>5</sup> with  $m - 1$  random known input-output pairs to recover a randomly selected unknown record from a data stream perturbed with RPCN at the highest noise level of 3. Note that as the distance in the stream between the unknown record and the closest known record increases, the minimum relative error of recovery attempts also tends to increase. This shows that known input-output pairs will become less useful for attacking new records as the stream progresses over time, i.e. RPCN's privacy level increases with time.

Figure 8.8 plots the accuracy of the online ARF classifier over time for the three perturbation methods (with RPIN and RPCN at the highest noise level of 3). As expected from the trade-off plots, RP generally produces the most accurate models, followed by cumulative noise and finally independent noise. Importantly, the accuracy in the presence of cumulative noise is stable over time: it is not continuously degrading as more noise is added. Of particular note is that this behaviour is observed for the RBF dataset, which has a known pattern of continuous drift. This demonstrates that the ARF classifier is able to adapt to concept drift as well as the injected noise.

Figure 8.9 provides some insight into how the accuracy is maintained by plotting a line that tracks the depth of each tree in the ARF ensemble over the course of the TAXI data stream when it has been perturbed by each of the three methods (once again, using the highest noise level of 3 for RPIN and RPCN). The mean tree depth was taken over intervals of 100 records, and a tree size dropping to zero indicates the removal and replacement of a tree (triggered by concept drift). It can be seen that RPCN results in the highest rate of tree removal, demonstrating that ARF is successfully replacing trees in response to the accumulating noise, allowing it to maintain a stable level of accuracy.

This trend analysis shows that cumulative noise is able to improve privacy over the lifetime of the stream while still perturbing data slowly enough that online classifiers may continuously adapt to maintain a stable level of accuracy.

---

<sup>5</sup>Attacks with relative error greater than 0.6 are omitted to highlight trends in minimum relative error.

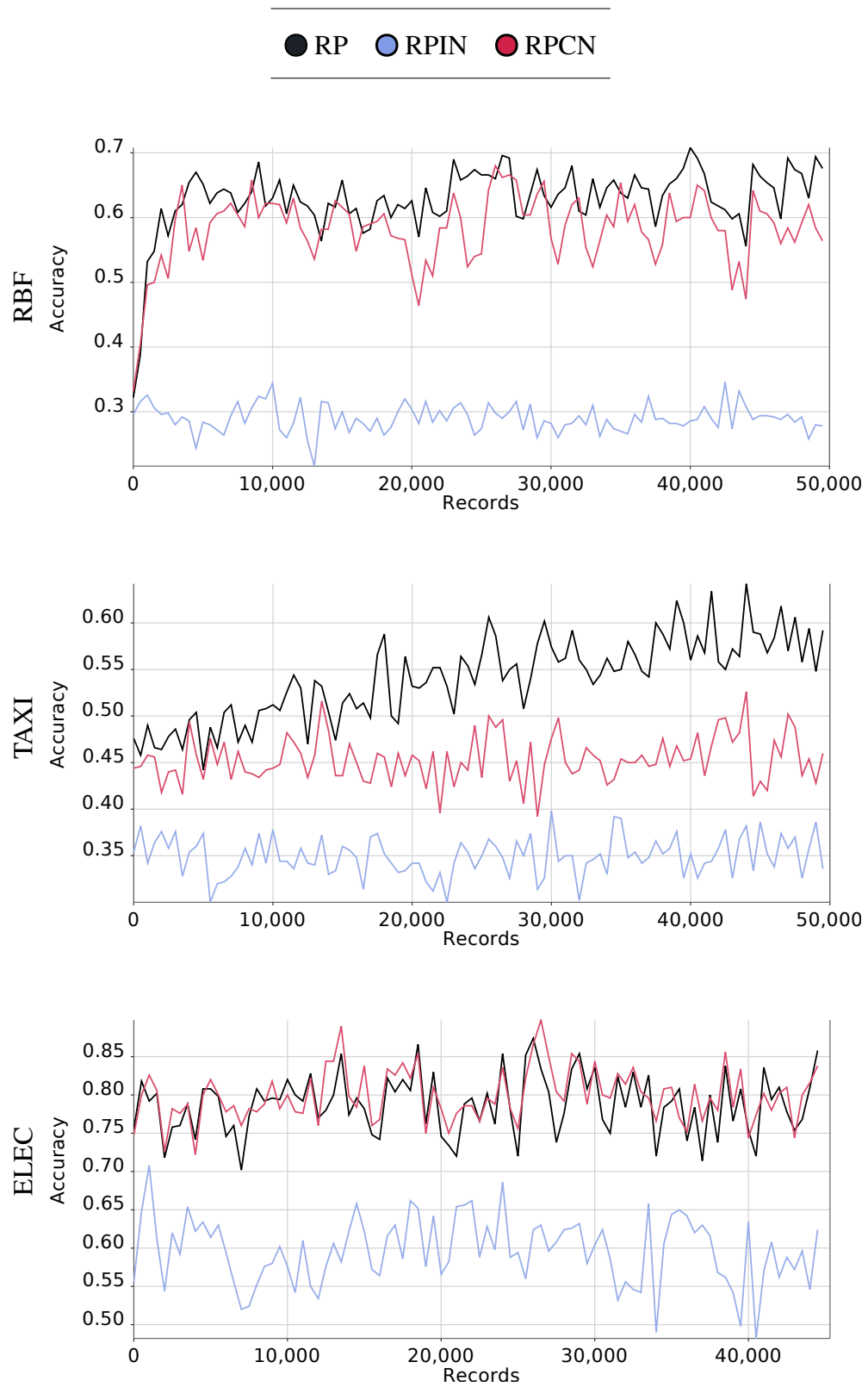


Figure 8.8: Trends of accuracy over the course of data streams perturbed with RP, RPIN, and RPCN.

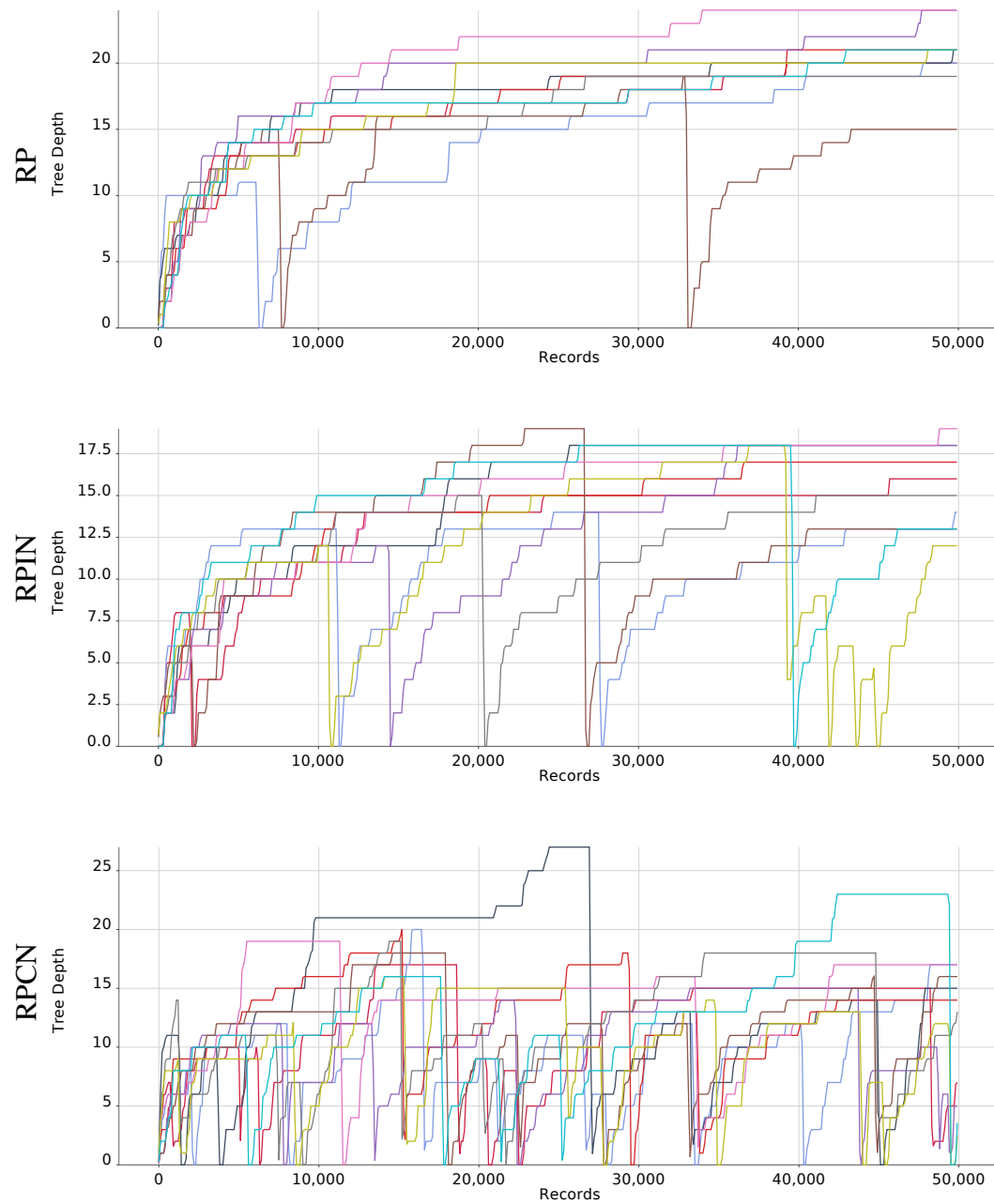


Figure 8.9: Trends of ARF tree depth over the course of the TAXI data stream perturbed with RP, RPIN, and RPCN.

## **Chapter 9**

### **Integrating Data Perturbation**

### **Methods into Distributed Stream**

### **Mining**

This chapter considers the case where data streams must be mined while taking into account the aspects of both distribution and privacy-preservation. Using the terminology of the HDSM architecture, this can occur when each primary site provides data from a different individual or organisation, or when trouble sites that merge data from primary sites are operated by a third-party, such as a data mining service provider. If the distributed streams contain sensitive data, then the individuals or organisations controlling them may not wish to share such data in their raw form.

While HDSM minimises the number of records that are transmitted from primary sites to trouble sites, it still involves the transmission of records in their original form. However, in principle it is possible to apply any of the perturbation methods from Chapter 7 to these records before they are transmitted from their primary site. This is made possible by the fact that the data perturbation methods preserve the identity of each record (even if the number and meaning of its features are altered), thus allowing

fragments from different primary sites to be merged at trouble sites. While the perturbed records will in general result in less accurate models at trouble sites, the records can still be mined in their original form at the primary sites to ensure the maximum possible accuracy of local models. It should also be noted that once a record has been perturbed at a primary site before transmission to a trouble site, it does not need to be perturbed again before transmission to higher-order trouble sites.

## 9.1 Secure Multiparty Data Perturbation

However, in the case of vertically-distributed datasets, it is possible for each primary site to only provide a small number of features. Perturbations applied to such small feature sets will be less privacy-preserving than those applied to larger feature sets, considering:

- Gaining as many known input-output pairs as there are features ( $p \geq m$ ) will become easier, potentially enabling attackers to perform more sophisticated attacks based on recovery of the random projection matrix (as described in Section 7.2.7).
- Each feature in a projected record will be based on a linear combination of fewer features. In the extreme case, if a primary site has a single feature, then the projected record (for  $k = m = 1$ ) would simply be a scalar multiplication of the original feature.
- Having a larger initial feature set allows the dimensionality to be reduced to a greater degree by random projection, thus leading to stronger privacy (K. Liu, 2007).

Instead of perturbing the vertical partition of each primary site individually, this section describes a novel approach that allows a single perturbation to be applied across

the partitions of multiple primary sites. This approach is based on two steps. Firstly, the perturbation is decomposed so that primary sites can produce partial perturbed records that when summed produce the same result as a centralised perturbation. Secondly, a secure multiparty computation protocol is used to allow the partial perturbed records to be summed at the trouble site without any site gaining knowledge of the partial perturbed record of any other individual site.

### 9.1.1 Decomposing the Perturbation Process for Distributed Computation

This subsection will demonstrate how data perturbation can be decomposed and distributed across a set of  $o$  primary sites such that the sum of individually computed partial perturbed records results in the same perturbed record that would have been produced had the entire dataset been centralised before perturbation. More formally, each primary site ( $i$ ) has a sub vector containing  $l_i$  of the  $m$  features of record  $x$ . Each primary site must be able to produce a partial perturbed record ( $y_i$ ) such that the sum of partial perturbed records is the fully perturbed record  $y$  with  $k$  features:

$$y = \sum_{i=1}^o y_i \quad (9.1)$$

A perturbation based on random projection, random translation, and cumulative noise (RPCN) will be demonstrated, but the same method is applicable in the case of independent noise (RPIN) by replacing all  $\gamma$  terms with  $\delta$  terms, or by ignoring  $\gamma$  terms in the absence of any additive noise (RP). Recall that for RPCN, perturbed record column-vector  $y$  is produced from record column-vector  $x$  by the following perturbation:



$$y = \frac{1}{\sqrt{k}\sigma_r} Rx + \psi + \gamma \quad (9.2)$$

Let  $R$  be the column-wise concatenation of two sub matrices  $A$  and  $B$  such that  $R_{*,j} = A_{*,j}$  whenever  $1 \leq j \leq l_a$  and  $R_{*,j} = B_{*,j-l_a}$  when  $l_a + 1 \leq j \leq l_a + l_b$ . The sub matrices  $A$  and  $B$  are randomly generated at two source sites that transmit their record fragments to a given trouble site. Now consider the composite data vector  $x \in \mathbb{R}^m$  at the trouble site as being composed of two sub vectors  $v \in \mathbb{R}^{l_a}$  and  $w \in \mathbb{R}^{l_b}$  corresponding to the record fragments supplied by the pair of source sites. The composite vector  $x$  is then given by  $x_j = v_j$  whenever  $1 \leq j \leq l_a$  and  $x_j = w_{j-l_a}$  when  $l_a + 1 \leq j \leq l_a + l_b$ .

Now it is easy to see that the multiplication  $Rx$  at the trouble site is distributive across the source sites:

$$\begin{aligned}
 A &= \begin{bmatrix} a_{1,1} & \dots & a_{1,l_a} \\ \dots & \dots & \dots \\ a_{k,1} & \dots & a_{k,l_a} \end{bmatrix} \quad B = \begin{bmatrix} b_{1,1} & \dots & b_{1,l_b} \\ \dots & \dots & \dots \\ b_{k,1} & \dots & b_{k,l_b} \end{bmatrix} \\
 R &= \begin{bmatrix} a_{1,1} & \dots & a_{1,l_a} & b_{1,1} & \dots & b_{1,l_b} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{k,1} & \dots & a_{k,l_a} & b_{k,1} & \dots & b_{k,l_b} \end{bmatrix} \\
 v &= \begin{bmatrix} v_1 \\ \dots \\ v_{l_a} \end{bmatrix} \quad w = \begin{bmatrix} w_1 \\ \dots \\ w_{l_b} \end{bmatrix} \quad x = \begin{bmatrix} v_1 \\ \dots \\ v_{l_a} \\ w_1 \\ \dots \\ w_{l_b} \end{bmatrix} \\
 Rx &= \begin{bmatrix} a_{1,1}v_1 + \dots + a_{1,l_a}v_{l_a} + b_{1,1}w_1 + \dots + b_{1,l_b}w_{l_b} \\ \dots \\ a_{k,1}v_1 + \dots + a_{k,l_a}v_{l_a} + b_{k,1}w_1 + \dots + b_{k,l_b}w_{l_b} \end{bmatrix} \\
 &= \begin{bmatrix} a_{1,1}v_1 + \dots + a_{1,l_a}v_{l_a} \\ \dots \\ a_{k,1}v_1 + \dots + a_{k,l_a}v_{l_a} \end{bmatrix} + \begin{bmatrix} b_{1,1}w_1 + \dots + b_{1,l_b}w_{l_b} \\ \dots \\ b_{k,1}w_1 + \dots + b_{k,l_b}w_{l_b} \end{bmatrix} \\
 &= Av + Bw
 \end{aligned} \tag{9.3}$$

Therefore, Equation (9.2) can be decomposed as follows:

$$\begin{aligned}
 y &= \sum_{i=1}^o y_i \\
 y_i &= \frac{1}{\sqrt{k}\sigma_r} R_i x + \psi_i + \gamma_i
 \end{aligned} \tag{9.4}$$

In Equation (9.4),  $R_i$  is a  $k \times l_i$  matrix with random elements generated in the same

way as the  $R$  matrix, essentially representing a slice of  $l_i$  columns of the original  $R$ . The vector  $\psi_i$  can be generated in exactly the same way as the original random translation  $\psi$ , using Equation (7.1). Using multiple translations drawn from the same distribution simply changes the magnitude of the translation, which has no effect on the behaviour of common data mining operations (such as classification) to be performed with the perturbed dataset. Finally,  $\gamma_i$  can be generated using the definition in Equation (7.3) (the same as  $\gamma$ ), but  $\sigma_\gamma^2$  must be replaced with  $\frac{\sigma_\gamma^2}{o}$  to account for the fact that the summed multiple  $\gamma_i$  values will have a variance equal to the sum of the variances used to generate the  $\gamma_i$  values.

Another desirable property of this decomposition is that each primary site can autonomously generate its own  $R_i$ ,  $\psi_i$ , and  $\gamma_i$  without needing to disclose these values to any other primary or trouble sites.

### 9.1.2 Secure Sum Protocol for Partial Perturbed Records

Producing a partial perturbed record at each primary site does not enable privacy-preserving distributed stream mining on its own. This is because transmitting a partial perturbed record in its present form may still be insecure, due to the fact that it is primarily the result of a projection to a higher dimensional feature space, which could lead to attacks that treat the projection as a complete system of linear equations (K. Liu, 2007). Therefore, this subsection describes a protocol that allows a trouble site to receive the sum of all partial perturbed records from its source sites without it or any primary site gaining any knowledge of the partial perturbed record ( $y_i$ ) of any other primary site.

To simplify the protocol, all features must be normalised such that they have a minimum value of 0, and so that a value  $\theta$  can be found that is known to be greater than the maximum possible value of any feature. If all features are min-max normalised,

then  $\theta = 1$ .

The protocol is based on the secure sum protocol of Clifton, Kantarcioglu, Vaidya, Lin and Zhu (2002), but accounts for the fact that the trouble site must receive the final sum without it providing any of the addends, rather than the sum being produced at an arbitrary party. Consider the case where two primary sites ( $a$  and  $b$ ) are acting as source sites to a trouble site. The protocol can be performed as follows:

1. Site  $a$  produces values  $z_a$  and  $\zeta_a$  according to:

$$\begin{aligned} z_a &= (y_a + \zeta_a) \mod \theta \\ \zeta_{a,i} &= \mathcal{U}(0, \theta) \end{aligned} \tag{9.5}$$

where  $\mathcal{U}(a, b)$  is a uniform distribution ranging from  $a$  to  $b$ .

The addition of  $y_a$  and  $\zeta_a$  with a modulus of  $\theta$  essentially hides  $y_a$  uniformly in the range  $[0, \theta]$ , preventing anyone with access to  $z_a$  from gaining any knowledge of  $y_a$  if they do not know  $\zeta_a$ .

2. Site  $a$  transmits  $z_a$  to primary site  $b$ , and  $\zeta_a$  to the trouble site.
3. Site  $b$  transmits  $z_a + y_b$  to the trouble site.
4. The trouble site uses  $z_a + y_b$  and  $\zeta_a$  to recover  $y$  as follows:

$$(z_a + y_b - \zeta_a + \theta) \mod \theta = y_a + y_b = y \tag{9.6}$$

The addition of  $\theta$  before the modulo operation accounts for the case when  $z_a - \zeta_a$  is negative because of the earlier modulo operation when producing  $z_a$ .

Note there is no need for primary site  $b$  to produce  $z_b$  for transmission to the trouble site as it would also need to transmit  $\zeta_b$  to the trouble site at the same time.

Note also that this protocol requires one more stage of communication than is normally required in HDSM: the communication from primary site  $a$  to primary site  $b$ . However, this extra communication cost can be partially offset by the fact that primary site  $b$  can filter out trouble records that sites  $a$  and  $b$  did not both agree on, meaning that only agreed upon trouble records will be transmitted to the trouble site (similar to the two-phase transmission protocol described in Section 5.2).

When there are more than two source sites (which may be required to produce a feature set large enough for privacy-preservation if each site only has a very small number of features), then the most computationally efficient way to implement the protocol is for it to be performed between pairs of sites. If there is an odd number of source sites, then one group of three source sites ( $a$ ,  $b$ , and  $c$ ) must perform the protocol as follows:

1. Primary site  $a$  transmits  $z_a$  to primary site  $b$  and  $\zeta_a$  to the trouble site.
2. Primary site  $b$  transmits  $z_a + z_b$  to primary site  $c$  and  $\zeta_b$  to the trouble site.
3. Primary site  $c$  transmits  $z_a + z_b + y_c$  to the trouble site.
4. The trouble site recovers  $y_a + y_b + y_c$  as:

$$(z_a + z_b + y_c - \zeta_a - \zeta_b + \theta) \mod \theta \quad (9.7)$$

Note that this variation of the protocol for three source sites requires one more round of communication than required for a pair of source sites: the transmission from primary site  $b$  to primary site  $c$ .

However, performing the secure sum protocol with pairs of source sites will result in the trouble site learning the partial sum of each pair of source sites, which may still not acceptably preserve privacy if the projection in the partial sum forms a complete

system of linear equations. Furthermore, there is a risk of collusion between the trouble site and any of the primary sites. This collusion problem exists for any protocol where the aim is for the trouble site to learn the sum:  $y = \sum_{i=1}^o y_i$ . If primary sites  $a$  and  $b$  perform the protocol such that the trouble site learns  $y_a + y_b$ , then the primary site  $a$  could collude with the trouble site by also transmitting  $y_a$  to the trouble site, which would allow it to recover  $y_b$  (and vice-versa for collusion with primary site  $b$ ). Involving more parties in the secure computation (i.e. extending the protocol for three source sites to include even more source sites) would reduce the completeness of the system of linear equations and necessitate collusion between the trouble site and a greater number of primary sites to achieve a privacy breach. For example, recovering  $y_a$  from  $y_a + y_b + y_c$  requires collusion between the trouble site and both sites  $b$  and  $c$ . However, as the number of communication rounds involved in the secure computation increases linearly with the number of source sites involved, extending the secure computation to include a large number of sites would likely be infeasible in the context of data stream mining. In practice, the number of source sites involved in each instance of the protocol would need to be selected to trade-off privacy and efficiency.

# Chapter 10

## Conclusions and Future Work

### 10.1 Research Achievements

This research has made several novel contributions.

Firstly, a novel Hierarchical Distributed Stream Miner (HDSM) for mining vertically-distributed data streams has been proposed. Experimentation showed that HDSM was able to achieve significant accuracy improvements with minimal data transmission to trouble sites while remaining competitive in terms of resource time with other distributed stream mining approaches. The experiments also demonstrated that HDSM is robust to the presence of concept drift as it adapts to changing cross-terms that manifest in the streams. It has also been shown that HDSM can be used for anytime classification to improve classification response time with minimal impact on accuracy, and that architecture variations based on batch record transmission and a probabilistic global view of site confidence can be used to further improve efficiency and accuracy.

Secondly, two novel data perturbation methods for privacy-preserving data stream mining and publishing have been proposed. The methods are based on a combination of random projection, random translation, and additive noise that is either generated completely independently for each record (RPIN), or accumulated over the course of the

stream (RPCN). Variations of the known input-output MAP attack were also developed for use against the proposed perturbation methods.

It was expected that the injection of additive noise would help to significantly boost the level of data privacy over the combination of random projection and random translation alone (RP). The cumulative noise addition scheme of RPCN was inspired by the notion of concept drift that is commonly present in data streams. It was hypothesised that the injection of an external signal that parallels natural concept drift would be recognisable by the classifier, hence triggering model adaptation periodically. This in turn would help to stabilise accuracy over time, and the overall expected outcome was the production of a better trade-off between data privacy and accuracy.

Experimentation showed that the perturbation methods involving additive noise achieved better trade-offs between data privacy and accuracy (as measured by the PAM metric) than their RP counterpart. Comparative tests between RPCN and RPIN also revealed a clear difference in accuracy between the two, with cumulative noise (RPCN) achieving an overall better trade-off between data privacy and accuracy over the independent noise (RPIN) scheme. The advantage of RPCN is that it injects noise gradually over time, while RPIN must distort each record by a much greater amount in order to achieve an equivalent level of noise. Both perturbation methods offered similar levels of data privacy, but the gradual drift of the cumulative noise allowed the classifier to adapt to the injected noise signal and maintain a higher accuracy, resulting in a better overall privacy/accuracy trade-off with RPCN.

It has also been shown that the best type of known input-output MAP attack against RPCN involved a combination of two attacks that did and did not account for the cumulative noise, where the attack accounting for cumulative noise was limited to a single known input-output pair. Importantly, this attack ( $\text{MAX}(\text{A-RP}, \text{A-RPCN-1})$ ) was shown to be less effective against records that are further away from the known records in the stream, demonstrating the data privacy of RPCN increases over time.



Finally, an approach has been described that integrates HDSM and data perturbation to achieve the goal of privacy in distributed data stream mining environments. This includes an algorithm for secure, multiparty, distributed data perturbation.

## 10.2 Limitations

There are a number of limitations to this research that should be recognised.

Firstly, the algorithm descriptions and experimental evaluations have focused on online classification in a prequential evaluation setting. However, the proposed methods are amenable to other machine learning problems. The prediction of any supervised learning problem can be used in place of the classification in HDSM. Likewise, a perturbed dataset can be used for a variety of data mining tasks, other than classification. Further experimentation in the context of other machine learning problems is an area for future work.

For the data perturbation methods, only attacks based on known input-output prior knowledge have been experimentally evaluated. The development and evaluation of different attack types based on different forms of prior knowledge is an important area for future research. Other forms of prior knowledge could include a sample of records drawn from the same distribution, feature means and covariances, and the random projection matrix  $R$ .

Experimental evaluation has also been limited in terms of the datasets used. In some cases, finding datasets that represented specific problems was not possible; publicly available (and therefore reproducible) datasets seldom have truly sensitive values that provide a realistic context for evaluating privacy-preservation. Additionally, all datasets have been relatively well balanced in the distribution of records among target classes. This was an intentional choice to remove the confounding factor of imbalanced learning while attempting to compare algorithm accuracy.

Finally, all distributed algorithms were tested with simplified implementations designed to run on a single machine. The development and evaluation of truly distributed implementations can be seen as an area for future work. Such an implementation of HDSM could also include the variations that take advantage of the batch communication protocol proposed in Chapter 5, and the approach for distributed privacy-preserving stream mining described in Chapter 9.

### **10.3 Future Work in Distributed Data Stream Mining**

There are still many opportunities to improve the accuracy and efficiency of HDSM, such as those described in Chapter 5 and Appendices B and C. Other classification result aggregation methods could potentially boost accuracy, including weighting votes by site accuracy (as measured by sliding windows). Accuracy could also be improved with alternative rules for triggering trouble site creation and removal, such as taking into account the degree to which the agreement threshold is exceeded when deciding between different candidate trouble sites involving a particular source site. The mechanism proposed in Section 5.4 may also improve trouble record selection by considering a probabilistic view of global site confidence. Improved trouble site creation rules could reduce the inefficient churn that can occur if many trouble sites are created and then removed soon after for lack of utility. To avoid the overhead of re-learning patterns in the case of recurrent concepts, models of removed trouble sites could be retained for later re-use, as in the work of Sakthithasan, Pears, Bifet and Pfahringer (2015).

## 10.4 Future Work in Privacy-Preserving Data Stream Mining

The research on privacy-preserving stream mining also presents opportunities for future research. While the proposed data perturbation methods can be applied to nominal data by treating them as ordinal integer values, the privacy guarantees are relatively weaker when an attacker knows that the original attribute is nominal (K. Liu, Kargupta & Ryan, 2006). Improvements to the methods that will provide stronger privacy guarantees for nominal attributes are still required. In general, future research on privacy-preservation methods that take advantage of the streaming nature of many datasets appears to be a promising direction.

While this research has shown the online ARF classifier is able to maintain stable accuracy in the presence of cumulative noise for the datasets tested, this may not be the case for less adaptive algorithms that may be more computationally efficient (and therefore more suitable for use in resource-constrained situations). To handle these cases, a drift detector could be used to monitor accuracy relative to the amount of cumulative noise. When the drift detector encounters a statistically significant difference in the ratio of the probability of privacy breach to accuracy over two contiguous sub-windows of a fixed-length window, a method of resetting the cumulative noise (i.e. to a zero vector) could be executed. Such a method could involve replacing the learned model with a new model based on a stream with a fresh cumulative noise generator. A transition period would be necessary to train the new model before using it for predictions. Ensuring that publishing two perturbed versions of the same stream in parallel does not leak additional information to an attacker would be a challenge of implementing this approach.

As previously mentioned, another particularly important area for future research lies in exploring different attack vectors to verify the privacy guarantees of the proposed perturbation methods. This includes the development of a known input-output attack

for the case when there are more known records than there are features in the original dataset ( $p \geq m$ ). Additionally, an attack method that makes use of spectral filtering (which has previously been used against additive noise) may be possible. The method of Giannella et al. (2013) to match known records with their perturbed versions could also be adapted for use against the perturbation methods proposed, which would provide understanding of the privacy achieved when the correspondence between input and output records is not known a priori by an attacker.

## References

- Abdelhameed, S. A., Moussa, S. M. & Khalifa, M. E. (2019). Restricted sensitive attributes-based sequential anonymization (RSA-SA) approach for privacy-preserving data stream publishing. *Knowledge-Based Systems*, 164, 1–20.
- Adhikari, A., Adhikari, J. & Pedrycz, W. (2016). *Data analysis and pattern recognition in multiple databases*. Springer.
- Aggarwal, C. C. & Philip, S. Y. (2004). A condensation approach to privacy preserving data mining. In *International Conference on Extending Database Technology* (pp. 183–199).
- Aggarwal, C. C. & Philip, S. Y. (2008). Privacy-preserving data mining: a survey. In *Handbook of Database Security* (pp. 431–460). Springer.
- Agrawal, R. & Srikant, R. (2000). Privacy-preserving data mining. In *ACM Sigmod Record* (Vol. 29, pp. 439–450).
- Baldi, P., Sadowski, P. & Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5, 4308.
- Basak, J. & Kothari, R. (2004). A classification paradigm for distributed vertically partitioned data. *Neural Computation*, 16(7), 1525–1544.
- Basu, K., Debusschere, V. & Bacha, S. (2013). Residential appliance identification and future usage prediction from smart meter. In *39th Annual Conference of the IEEE Industrial Electronics Society, IECON 2013* (pp. 4994–4999).
- Bifet, A. & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining* (pp. 443–448).
- Bifet, A., Holmes, G., Kirkby, R. & Pfahringer, B. (2010). MOA: massive online analysis. *Journal of Machine Learning Research*, 11, 1601–1604. Retrieved from <http://portal.acm.org/citation.cfm?id=1859903>
- Bifet, A. & Kirkby, R. (2009). *Data stream mining a practical approach*.
- Candanedo, L. M. & Feldheim, V. (2016). Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112, 28–39.
- Cao, Y. & Yoshikawa, M. (2016). Differentially private real-time data publishing over infinite trajectory streams. *IEICE TRANSACTIONS on Information and Systems*, 99(1), 163–175.
- Centers for Disease Control and Prevention. (2005, February). *National survey of family growth data*. Retrieved February 12, 2019, from <http://>

- [www.greenteapress.com/thinkstats/nsfg.html](http://www.greenteapress.com/thinkstats/nsfg.html)
- Chamikara, M., Bertok, P., Liu, D., Camtepe, S. & Khalil, I. (2018). Efficient data perturbation for privacy preserving and accurate data stream mining. *Pervasive and Mobile Computing*, 48, 1–19.
- Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.
- Chen, K., Sun, G. & Liu, L. (2007). Towards attack-resilient geometric data perturbation. In *Proceedings of the 2007 SIAM International Conference on Data Mining* (pp. 78–89).
- Chen, R., Sivakumar, K. & Kargupta, H. (2001). An approach to online Bayesian learning from multiple data streams. In *Proceedings of Workshop on Mobile and Distributed Data Mining, PKDD* (Vol. 1, pp. 31–45).
- Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X. & Zhu, M. Y. (2002). Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2), 28–34.
- Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan), 1–30.
- Devi, S. (2014). A survey on distributed data mining and its trends. *International Journal of Research in Engineering & Technology (IMPACT: IJRET)*, 2(3), 107–120.
- Dheeru, D. & Karra Taniskidou, E. (2017). *UCI machine learning repository*. Retrieved from <http://archive.ics.uci.edu/ml> (University of California, Irvine, School of Information and Computer Sciences)
- Dwork, C. (2008). Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation* (pp. 1–19).
- Dwork, C., Naor, M., Pitassi, T. & Rothblum, G. N. (2010). Differential privacy under continual observation. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing* (pp. 715–724).
- Freire, A. L., Barreto, G. A., Veloso, M. & Varela, A. T. (2009). Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In *6th Latin American Robotics Symposium (LARS), 2009* (pp. 1–6).
- Gama, J., Medas, P., Castillo, G. & Rodrigues, P. (2004). Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence* (pp. 286–295).
- Giannella, C. R., Liu, K. & Kargupta, H. (2013). Breaching Euclidean distance-preserving data perturbation using few known inputs. *Data & Knowledge Engineering*, 83, 93–110.
- Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., ... Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10), 1469–1495.
- Guo, S., Wu, X. & Li, Y. (2008). Determining error bounds for spectral filtering based reconstruction methods in privacy preserving data mining. *Knowledge and Information Systems*, 17(2), 217–240.
- Hebrail, G. (2012). *Individual household electric power consumption data*

- set. Retrieved from <https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption>
- Huang, Z. & Du, W. (2008). OptRR: Optimizing randomized response schemes for privacy-preserving data mining. In *Proceedings of the IEEE 24th International Conference on Data Engineering, 2008* (pp. 705–714).
- Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9), 1098–1101.
- Hulten, G., Spencer, L. & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 97–106).
- Jiang, W. & Clifton, C. (2006). A secure distributed framework for achieving k-anonymity. *The VLDB Journal—The International Journal on Very Large Data Bases*, 15(4), 316–333.
- Johnson, W. B. & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(189-206), 1.
- Kaggle. (2017, September). *New york city taxi trip duration*. Retrieved February 12, 2019, from <https://www.kaggle.com/c/nyc-taxi-trip-duration/data>
- Kargupta, H., Datta, S., Wang, Q. & Sivakumar, K. (2003). On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the Third IEEE International Conference on Data Mining, 2003. ICDM 2003.* (pp. 99–106).
- Kellaris, G., Papadopoulos, S., Xiao, X. & Papadias, D. (2014). Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12), 1155–1166.
- Khodaparast, F., Sheikhalishahi, M., Haghighi, H. & Martinelli, F. (2018). Privacy preserving random decision tree classification over horizontally and vertically partitioned data. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)* (pp. 600–607).
- Kohavi, R. (1996). Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In *KDD* (Vol. 96, pp. 202–207).
- Kourtellis, N., Morales, G. D. F., Bifet, A. & Murdopo, A. (2016). VHT: Vertical hoeffding tree. In *IEEE International Conference on Big Data (Big Data), 2016* (pp. 915–922).
- Li, F., Sun, J., Papadimitriou, S., Mihaila, G. A. & Stanoi, I. (2007). Hiding in the crowd: Privacy preservation on evolving streams through correlation tracking. In *Proceedings of the IEEE 23rd International Conference on Data Engineering, 2007* (pp. 686–695).
- Li, T., Li, J., Liu, Z., Li, P. & Jia, C. (2018). Differentially private naive Bayes learning over multiple data sources. *Information Sciences*, 444, 89–104.
- Li, Y., Jiang, Z. L., Yao, L., Wang, X., Yiu, S. & Huang, Z. (2017). Outsourced privacy-preserving C4.5 decision tree algorithm over horizontally and vertically partitioned dataset among multiple parties. *Cluster Computing*, 1–13.

- Liu, K. (2007). *Multiplicative data perturbation for privacy preserving data mining* (Unpublished doctoral dissertation). University of Maryland, Baltimore County.
- Liu, K., Giannella, C. & Kargupta, H. (2006). An attacker's view of distance preserving maps for privacy preserving data mining. In *European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 297–308).
- Liu, K., Giannella, C. & Kargupta, H. (2008). A survey of attack techniques on privacy-preserving data perturbation methods. In *Privacy-preserving data mining* (pp. 359–381). Springer.
- Liu, K., Kargupta, H. & Ryan, J. (2006). Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering*, 18(1), 92–106.
- Liu, L., Wang, J. & Zhang, J. (2008). *Privacy vulnerabilities with background information in data perturbation* (Tech. Rep.). Davis Marksbury Building, 329 Rose Street, Lexington, KY 40506-0633, USA: Technical report. Department of Computer Science, University of Kentucky.
- Liu, Y., Xu, Z. & Li, C. (2018). Distributed online semi-supervised support vector machine. *Information Sciences*, 466, 236–257.
- Lyon, R. (2017). HTRU2. *UCI Machine Learning Repository*.
- Lyon, R. J., Stappers, B., Cooper, S., Brooke, J. & Knowles, J. (2016). Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1), 1104–1123.
- Lyu, L., Bezdek, J. C., Law, Y. W., He, X. & Palaniswami, M. (2018). Privacy-preserving collaborative fuzzy clustering. *Data & Knowledge Engineering*, 116, 21–41.
- Mangasarian, O. L., Street, W. N. & Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4), 570–577.
- Mendes, R. & Vilela, J. P. (2017). Privacy-preserving data mining: methods, metrics, and applications. *IEEE Access*, 5, 10562–10582.
- Moghadam, A. N. & Ravanmehr, R. (2018). Multi-agent distributed data mining approach for classifying meteorology data: case study on Iran's synoptic weather stations. *International Journal of Environmental Science and Technology*, 15(1), 149–158.
- Narayanan, A. & Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy, 2008. SP 2008*. (pp. 111–125).
- Nelder, J. A. & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7(4), 308–313.
- Nin, J., Herranz, J. & Torra, V. (2008). Rethinking rank swapping to decrease disclosure risk. *Data & Knowledge Engineering*, 64(1), 346–364.
- Obenshain, M. K. (2004). Application of data mining techniques to healthcare data. *Infection Control & Hospital Epidemiology*, 25(8), 690–695.
- Okkalioglu, B. D., Okkalioglu, M., Koc, M. & Polat, H. (2015). A survey: deriving private information from perturbed data. *Artificial Intelligence Review*, 44(4),



- 547–569.
- Omer, M. Z., Gao, H. & Mustafa, N. (2017). Privacy-preserving of SVM over vertically partitioned with imputing missing data. *Distributed and Parallel Databases*, 35(3-4), 363–382.
- Otgonbayar, A., Pervez, Z., Dahal, K. & Eager, S. (2018). K-VARP: K-anonymity for varied data streams via partitioning. *Information Sciences*, 467, 238–255.
- Oza, N. (2011). *FLTz flight simulator*. Retrieved from <https://c3.nasa.gov/dashlink/resources/294/>
- Palumbo, F., Gallicchio, C., Pucci, R. & Micheli, A. (2016). Human activity recognition using multisensor data fusion based on reservoir computing. *Journal of Ambient Intelligence and Smart Environments*, 8(2), 87–107.
- Park, B.-H., Ayyagari, R. & Kargupta, H. (2001). A Fourier analysis based approach to learning decision trees in a distributed environment. In *Proceedings of the 2001 SIAM International Conference on Data Mining* (pp. 1–22).
- Parker, B., Mustafa, A. M. & Khan, L. (2012). Novel class detection and feature via a tiered ensemble approach for stream mining. In *IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI), 2012* (Vol. 1, pp. 1171–1178).
- Paschke, F., Bayer, C., Bator, M., Mönks, U., Dicks, A., Enge-Rosenblatt, O. & Lohweg, V. (2013). Sensorlose zustandsüberwachung an synchronmotoren. In *Proceedings. 23. Workshop Computational Intelligence, Dortmund*, 5 (p. 211).
- Prosper Marketplace, Inc. (2014, November). *Loan data from prosper*. Retrieved February 12, 2019, from [https://docs.google.com/document/d/1qEcwl\\_tBMlRYZT-1699-71TzInWfk4W9q5rTCSvDVMpc/pub](https://docs.google.com/document/d/1qEcwl_tBMlRYZT-1699-71TzInWfk4W9q5rTCSvDVMpc/pub)
- Recamonde-Mendoza, M. & Bazzan, A. L. (2016). Social choice in distributed classification tasks: Dealing with vertically partitioned data. *Information Sciences*, 332, 56–71.
- Rehman, M. H., Liew, C. S., Wah, T. Y. & Khan, M. K. (2017). Towards next-generation heterogeneous mobile data stream mining applications: Opportunities, challenges, and future research directions. *Journal of Network and Computer Applications*, 79, 1 - 24.
- Rodríguez, D. M., Nin, J. & Nuñez-del Prado, M. (2017). Towards the adaptation of SDC methods to stream mining. *Computers & Security*, 70, 702–722.
- Sakpere, A. B. & Kayem, A. V. (2014). A state-of-the-art review of data stream anonymization schemes. In *Information Security in Diverse Computing Environments* (pp. 24–50). IGI Global.
- Sakthithasan, S., Pears, R., Bifet, A. & Pfahringer, B. (2015). Use of ensembles of Fourier spectra in capturing recurrent concepts in data streams. In *International Joint Conference on Neural Networks (IJCNN), 2015* (pp. 1–8).
- Samuelsson, J. (2016). *Anomaly detection in consolelogs* (No. 16012). (Degree Project, Uppsala University, Department of Information Technology)
- Sang, Y., Shen, H. & Tian, H. (2012). Effective reconstruction of data perturbed by random projections. *IEEE Transactions on Computers*, 61(1), 101–117.
- Skillicorn, D. B. & McConnell, S. M. (2008). Distributed prediction from vertically

- partitioned data. *Journal of Parallel and Distributed computing*, 68(1), 16–36.
- Smith, J. O. (2011). *Spectral audio signal processing*. <http://ccrma.stanford.edu/~jos/sasp/>. (online book, 2011 edition)
- Street, W. N. & Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 377–382).
- Tumer, K. & Ghosh, J. (2000). Robust order statistics based ensemble for distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery* (pp. 185–210). AAAI/ MIT Press.
- Vergara, A., Vembu, S., Ayhan, T., Ryan, M. A., Homer, M. L. & Huerta, R. (2012). Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166, 320–329.
- Wang, J., Deng, C. & Li, X. (2018). Two privacy-preserving approaches for publishing transactional data streams. *IEEE Access*, 6, 23648–23658.
- Wang, J., Liu, C., Fu, X., Luo, X. & Li, X. (2019). A three-phase approach to differentially private crucial patterns mining over data streams. *Computers & Security*, 82, 30–48.
- Wang, K., Fung, B. C. & Dong, G. (2005). Integrating private databases for data analysis. In *International Conference on Intelligence and Security Informatics* (pp. 171–182).
- Wang, Q., Zhang, Y., Lu, X., Wang, Z., Qin, Z. & Ren, K. (2018). Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy. *IEEE Transactions on Dependable and Secure Computing*, 15(4), 591–606.
- Wang, T., Guan, S.-U., Man, K. L. & Ting, T. (2014). EEG eye state identification using incremental attribute learning with time-series classification. *Mathematical Problems in Engineering*, 2014.
- Wang, T., Zheng, Z., Rehmani, M. H., Yao, S. & Huo, Z. (2018). Privacy preservation in big data from the communication perspective—a survey. *IEEE Communications Surveys & Tutorials*, 1–1. (Early Access)
- Witten, I. H., Frank, E., Hall, M. A. & Pal, C. J. (2016). Moving on: applications and beyond. In *Data mining: practical machine learning tools and techniques*. (p. 510–511). Cambridge, MA : Morgan Kaufmann Publisher, [2016].
- Zhao, J., Yang, J. & Zhang, J. (2014). Privacy properties of random projection perturbation when random matrix is leaking. *Journal of Computational Information Systems*, 10(8), 3465–3472.

# Appendix A

## Quantile Threshold Examples and Proofs

This appendix provides examples and proofs for a number of the claimed properties of the terms used in calculating quantile thresholds for trouble record transmission.

### A.1 Agreement Extrema

This section provides proofs that  $0 \leq a_t \leq 1$  for the definition of  $a_t$  in Equation (3.1) where  $t$  is a trouble site.

Firstly, note that the sum of the agreement window's contents ( $\sum_{i=1}^{|w|} w_i$ ) represents the total number of record fragments received, and can therefore also be expressed as the product of the number of source sites ( $s$ ) and the mean number of record fragments sent by each site ( $\mu_r$ ). Also note that  $\sum_{i=1}^{|w|} I(w_i = s)$  counts the number of unique records received that achieved full agreement. The maximum possible value for this expression is therefore  $\mu_r$ , as the highest number of agreeing records is achieved when all source sites send fragments for exactly the same set of records. Therefore, it follows:

$$\begin{aligned}
\max a_t &= \max s \left( \frac{\sum_{i=1}^{|w|} I(w_i = s)}{\sum_{i=1}^{|w|} w_i} \right) \\
&= \max s \left( \frac{\sum_{i=1}^{|w|} I(w_i = s)}{s\mu_r} \right) \\
&= \frac{\max \sum_{i=1}^{|w|} I(w_i = s)}{\mu_r} \\
&= \frac{\mu_r}{\mu_r} = 1
\end{aligned} \tag{A.1}$$

Furthermore, as the minimum agreement occurs when every record transmits different trouble records ( $\min \sum_{i=1}^{|w|} I(w_i = s) = 0$ ) and  $s \geq 2$ :

$$\begin{aligned}
\min a_t &= \min s \left( \frac{\sum_{i=1}^{|w|} I(w_i = s)}{\sum_{i=1}^{|w|} w_i} \right) \\
&= \min s \left( \frac{\min \sum_{i=1}^{|w|} I(w_i = s)}{\sum_{i=1}^{|w|} w_i} \right) \\
&= \min s \left( \frac{0}{\sum_{i=1}^{|w|} w_i} \right) = 0
\end{aligned} \tag{A.2}$$

## A.2 Example of Agreement in the Case of Unequal Record Transmissions

This section demonstrates that when source sites transmit different numbers of records, the agreement is computed as a proportion of the mean number of records transmitted by each site.

Consider the case where two source sites transmit 4 and 8 records respectively, and there is agreement on as many records as possible (4). The agreement window will contain [2, 2, 2, 2, 1, 1, 1, 1], and the agreement will be computed as 4 agreeing records out of a mean number of 6 transmitted records, according to the definition of agreement in Equation (3.1):

$$a = s \left( \frac{\sum_{i=1}^{|w|} I(w_i = s)}{\sum_{i=1}^{|w|} w_i} \right) = 2 \left( \frac{\sum_{i=1}^8 I(w_i = 2)}{\sum_{i=1}^8 w_i} \right) = 2 \left( \frac{4}{12} \right) = \frac{4}{6} \quad (\text{A.3})$$

### A.3 Reduction Coefficient Extrema

This section provides proofs that  $0 \leq r_s \leq 1$  for the definition of  $r_s$  in Equation (3.2) when  $s$  is a trouble site ( $r_s = 1$  by definition when  $s$  is a primary site). Given  $0 \leq a_s \leq 1$ :

$$\begin{aligned} \min r_s &= \min(a_s \times \min(1, k_s)) \\ &= 0 \times \min(1, k_s) = 0 \end{aligned} \quad (\text{A.4})$$

$$\begin{aligned} \max r_s &= \max(a_s \times \min(1, k_s)) \\ &= 1 \times 1 = 1 \end{aligned} \quad (\text{A.5})$$

### A.4 Two-Phase Reduction Coefficients Extrema

This section provides proofs that  $0 \leq r_x^{(1)} \leq 1$  and  $0 \leq r_x^{(2)} \leq 1$  for any trouble site  $x$  given the definitions of  $r_x^{(1)}$  and  $r_x^{(2)}$  for trouble sites in Equation (5.3). Given  $a_s \geq 0$ :

$$\begin{aligned}
\min(r_x^{(2)}) &= \min(a_x \times r_x^{(1)}) \\
&= \min a_x \times \min r_x^{(1)} \\
&= 0 \times \min r_x^{(1)} = 0 \\
\min(r_x^{(1)}) &= \min \frac{1}{|S_x|} \sum_{s \in S_x} (r_s^{(2)} \times q_{s \Rightarrow x}) \\
&= \min \frac{1}{|S_x|} \sum_{s \in S_x} (\min r_s^{(2)} \times \min q_{s \Rightarrow x}) \\
&= \min \frac{1}{|S_x|} \sum_{s \in S_x} (0 \times \min q_{s \Rightarrow x}) \\
&= \min \frac{1}{|S_x|} \sum_{s \in S_x} 0 \\
&= \min \frac{1}{|S_x|} 0 = 0
\end{aligned} \tag{A.6}$$

The proofs for the maximum values of  $r_x^{(1)}$  and  $r_x^{(2)}$  are first given for the base case where the source sites ( $S_x$ ) are all primary sites, and therefore  $r_s^{(2)} = 1 \forall s \in S_x$ . Then, given  $q_{s \Rightarrow x} \leq 1 \forall s \in S_x$  and  $0 \leq a_s \leq 1$ :

$$\begin{aligned}
\max r_x^{(1)} &= \max \frac{1}{|S_x|} \sum_{s \in S_x} (r_s^{(2)} \times q_{s \Rightarrow x}) \\
&= \max \frac{1}{|S_x|} \sum_{s \in S_x} (\max(r_s^{(2)}) \times \max(q_{s \Rightarrow x})) \\
&= \max \frac{1}{|S_x|} \sum_{s \in S_x} (1 \times 1) \\
&= \max \frac{1}{|S_x|} |S_x| = 1 \\
\max r_x^{(2)} &= \max(a_x \times r_x^{(1)}) \\
&= \max a_x \times \max r_x^{(1)} \\
&= 1 \times 1 = 1
\end{aligned} \tag{A.7}$$

Because  $r_x^{(1)}$  and  $r_x^{(2)}$  have maximum values of 1 for the base case when  $S_x$  only contains primary sites, the above equations can be also be applied inductively to show

this condition also holds for all higher-order trouble sites because  $\max(r_s^{(2)})$  will always be 1.

## A.5 First-Phase Transmission Equality

This section provides a proof that the number of records transmitted for  $r_s^{(2)} \times q_{s \Rightarrow d}$  is equal for all source sites ( $s$ ) of a trouble site  $d$  when there are enough records available to be transmitted from each source site. More formally,  $r_a^{(2)} \times q_{a \Rightarrow d} = r_b^{(2)} \times q_{b \Rightarrow d} \forall a, b \in S_d$  given  $r_a^{(2)}$  and  $r_b^{(2)}$  are sufficiently large:

$$\begin{aligned} r_s^{(2)} \times q_{s \Rightarrow d} &= r_s^{(2)} \times \min \left( 1, \frac{t_d^{(1)}}{l_d \times r_s^{(2)}}, \frac{t_d^{(2)}}{l_d \times r_s^{(2)} \times a_d} \right) \\ &= \min \left( r_s^{(2)}, \frac{t_d^{(1)}}{l_d}, \frac{t_d^{(2)}}{l_d \times a_d} \right) \end{aligned} \tag{A.8}$$

Note that the second two terms of the min operation are independent of  $s$ , and will therefore be the same for all source sites. The first term ( $r_s^{(2)}$ ) represents the extreme case when all records are transmitted from a trouble site. Therefore, the condition will only hold if  $r_s^{(2)}$  is large enough for it to not become the limiting factor in the above equation, i.e. when the source site is able to supply a quantity of records equal to that of all other source sites.

## A.6 Two-Phase Transmission Limits

This section provides proofs that the phase-specific reduction coefficients  $r_d^{(1)}$  and  $r_d^{(2)}$  at a trouble site  $d$  never exceed the ratio of the configured trouble factor for the respective phase ( $t_d^{(1)}$  or  $t_d^{(2)}$ ) and the size of the feature set at the trouble site ( $l_d$ ). More formally,  $r_d^{(1)} \leq \frac{t_d^{(1)}}{l_d}$  and  $r_d^{(2)} \leq \frac{t_d^{(2)}}{l_d}$ :

$$r_d^{(1)} = \frac{1}{|S_d|} \sum_{s \in S_d} (r_s^{(2)} \times q_{s \Rightarrow d})$$

Assuming the  $t_d^{(1)}$  term is limiting  $q_{s \Rightarrow d}$

$$\begin{aligned} &\leq \frac{1}{|S_d|} \sum_{s \in S_d} \left( r_s^{(2)} \times \frac{t_d^{(1)}}{l_d \times r_s^{(2)}} \right) \\ &\leq \frac{1}{|S_d|} \sum_{s \in S_d} \left( \frac{t_d^{(1)}}{l_d} \right) \\ &\leq \frac{1}{|S_d|} \times |S_d| \times \frac{t_d^{(1)}}{l_d} \\ &\leq \frac{t_d^{(1)}}{l_d} \end{aligned}$$

$$r_d^{(2)} = a_d \times r_d^{(1)} \tag{A.9}$$

$$= a_d \times \frac{1}{|S_d|} \sum_{s \in S_d} (r_s^{(2)} \times q_{s \Rightarrow d})$$

Assume the  $t_d^{(2)}$  term is limiting  $q_{s \Rightarrow d}$

$$\begin{aligned} &\leq a_d \times \frac{1}{|S_d|} \sum_{s \in S_d} \left( r_s^{(2)} \times \frac{t_d^{(2)}}{l_d \times r_s^{(2)} \times a_d} \right) \\ &\leq a_d \times \frac{1}{|S_d|} \sum_{s \in S_d} \left( \frac{t_d^{(2)}}{l_d \times a_d} \right) \\ &\leq a_d \times \frac{1}{|S_d|} \times |S_d| \times \frac{t_d^{(2)}}{l_d \times a_d} \\ &\leq \frac{t_d^{(2)}}{l_d} \end{aligned}$$



# Appendix B

## HDSM Data Compression

Another method to reduce the cost of data transmission in HDSM is to use a lossless compression scheme to reduce the size of data transmitted to trouble sites and the aggregator.

### B.1 Key Compression

Because record keys must be unique, the number of bits required to represent a different value for each record can become a significant communication overhead. In these cases, techniques to reduce the size of the key can be used.

One way to reduce key sizes is to relax the uniqueness constraint: the key only needs to be unique with regards to all other records being processed by the system at the same time (i.e. records with which there may be confusion in the case of a duplicate key). If the original unique keys are monotonically increasing integers (e.g. timestamps), then they can be compressed to a smaller representation by applying the modulo operator. The divisor ( $d$ ) should represent the maximum distance (number of records) that could feasibly exist between two records being processed simultaneously by the system. For example, if a record fragment may arrive out of order by  $x$  seconds in either direction

(i.e. arrive  $x$  seconds earlier or later than the mean arrival time for other fragments of the same record, resulting in a maximum distance of  $2x$  records between fragments for the same record), and the HDSM system processes  $y$  records per second, then the minimum required divisor ( $d$ ) can be expressed as:  $d = 2xy$ . The number of bits required for the new key ( $\text{size}(i)$ ) can then be computed from  $d$ :  $\text{size}(i) = \lceil \log_2 d \rceil$ . This key compression method is robust both to records failing to arrive at one or more sites and records arriving out of order by up to  $d$  records.

If two-phase and batch transmission are in use, then a compressed representation can also be used for the subset of “agreed” keys transmitted back from the trouble site to the source sites. Instead of the globally unique keys, the positional indices from each batch can be used to identify the subset of trouble records in that batch. As batches of trouble record keys may be ordered differently by different source sites, the positional indices to return must be determined separately for each source site. This would reduce the data transmission with the two-phase approach to:  $ng + pb + pf$  (where  $b$  is the size of the positional index, and the other terms are defined in Equation (5.1)). The size (in bits) of the positional index  $b$  can be computed from the size of the batch ( $m$ ) such that there can be one unique  $b$  value for each position in the batch:  $\text{size}(b) = \lceil \log_2 m \rceil$ . These new definitions can be used to update the inequality in Equation (5.2):

$$\begin{aligned}
 ng + pb + pf &< n(g + f) \\
 ng + anb + anf &< n(g + f) \\
 g + ab + af &< g + f \\
 ab &< f - af \\
 b &< f\left(\frac{1}{a} - 1\right) \\
 \lceil \log_2 m \rceil &< f\left(\frac{1}{a} - 1\right)
 \end{aligned} \tag{B.1}$$

Therefore, the size of the batch now impacts the decision of whether to use single-phase or two-phase transmission rather than the size of the globally unique key. This provides more flexibility for the user, as the size of the batch is a configurable parameter.

## B.2 Feature Compression

When transmitting a batch of record features, there will be redundancy when multiple records have the same feature values. This is a typical case when features are nominal or ordinal. This redundancy can be reduced by representing the batch as a mapping of each unique combination of feature values to the keys of records they represent, for example:

$$\left\{ \begin{array}{ll} \text{key} & \text{features} \\ \text{key(a)} & \mapsto (0,0,1) \\ \text{key(b)} & \mapsto (0,1,0) \\ \text{key(c)} & \mapsto (0,0,1) \end{array} \right\} \Rightarrow \left\{ \begin{array}{ll} \text{features} & \text{list of keys} \\ (0,0,1) & \mapsto \{\text{key(a), key(c)}\} \\ (0,1,0) & \mapsto \{\text{key(b)}\} \end{array} \right\} \quad (\text{B.2})$$

For single-phase transmission, using the inverted representation will always reduce the data size proportionally to the number of duplicate feature combinations. For two-phase transmission, the inverted representation is only useful for the second phase (when feature values are transmitted), though it may not be more efficient as it would require keys to be added to the second phase transmission. The original transmission cost for a second-phase batch is the product of the number of records in the batch ( $m$ ) and the size of the feature values for a record fragment ( $f$ ):  $mf$ . When the inverted representation is used, the cost also includes a key for each record in the batch (where the key size can represent each record in the batch uniquely:  $b = \lceil \log_2 m \rceil$ ), but feature values only need to be transmitted for the number of unique feature combinations within

the batch ( $u$ ):  $mb + uf$ . An inequality between these two costs can be used to find when the inverted representation is more efficient:

$$\begin{aligned}
 mb + uf &< mf \\
 b + \frac{uf}{m} &< f \\
 \frac{b}{f} + \frac{u}{m} &< 1
 \end{aligned} \tag{B.3}$$

Therefore, the inverted representation will be more efficient when the ratio of batch key size to feature set size ( $\frac{b}{f}$ ) and the ratio of unique records to the total number of records ( $\frac{u}{m}$ ) are both sufficiently small. It is worth noting that ratio  $\frac{u}{m}$  is expected to increase as the number of possible feature combinations increases, meaning that the inverted representation will be more efficient for nominal features with small cardinalities. Conversely, the ratio should decrease as  $m$  increases, given a larger batch size leads to a greater probability of duplicate feature values within the same batch.

Even greater compression could be possible by representing each batch using a tree structure similar to that formed by Huffman coding (Huffman, 1952), where branches represent different feature values (with branches nearer the root based on more common feature values) and leaves specify the keys of the records having the feature values along the path to the leaf.

## Appendix C

# Omitting Unconfident Classifications in HDSM

In the current architecture of HDSM, all classification results are transmitted from every primary and trouble site to the aggregator. This includes unconfident classifications that resulted in the site forwarding the record to a trouble site. It may be possible to omit transmitting these unconfident classifications to the aggregator and still achieve comparable accuracy.

Omitting these classifications still guarantees that at least one classification arrives at the aggregator for each record. In the extreme case where all primary sites forward a particular record as trouble (so none of them will forward their classification to the aggregator), then trouble sites will achieve full agreement for the record and be able to provide their own classifications. Even if multiple layers of trouble sites forward the record on as trouble, there must always be a final layer of trouble sites that will provide one or more classifications to the aggregator.

Initial experimentation with max-confidence aggregation indicated that omitting these classifications does not drastically impact overall accuracy, as they are very rarely selected as the final classification by the aggregator due to their low confidence. In cases

where they were selected, they more frequently resulted in an incorrect classification instead of correcting the next-most-confident classification.

However, the aggregator currently requires classification transmissions from all sites in order to determine which sites to not expect classifications from (as per Algorithm 3), and this requirement is not fulfilled with the omission of unconfident classifications. This could potentially be resolved by altering HDSM's behaviour in one of the following ways:

1. A source site could still notify the aggregator of the trouble sites it has forwarded to, even when it does not need to send a classification. Given the classification result is relatively small (it only contains the assigned class value and its confidence), this would not represent a significant saving in data transmission cost.
2. If a trouble site receives a trouble fragment from a source site but never processes it (e.g. it is dropped from the input buffer because all source site fragments didn't arrive for the record), then the trouble site could notify the aggregator that the source sites had finished processing the record. If a trouble site has more than two source sites, then this could be slightly more efficient than sending notifications from source sites individually, but a large amount of latency would be introduced due to waiting for an unprocessed record to drop off the trouble site's input buffer before it could be classified.
3. The aggregator could assume all classifiers are finished after a certain timeout period. This completely removes the need for communication in the case of unconfident results, but the timeout would add a degree of latency. If batch transmission is used for classification results, the timeout could be set as a number of batches to check from each site after a unique record is first seen at the

aggregator. If only a single batch is checked, then the latency would be no more than that already introduced by batch transmission.