Patterns and Privacy Preservation with Prior Knowledge for Classification

by

Shaofeng Bu

B.Eng., Xi’an Jiaotong University China, 1998
M.Eng., Xi’an Jiaotong University China, 2001
M.Sc., The University of British Columbia, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

in
The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)
September 2010
© Shaofeng Bu 2010
Abstract

Privacy preservation is a key issue in outsourcing of data mining. When we seek approaches to protect the sensitive information contained in the original data, it is also important to preserve the mining outcome. We study the problem of privacy preservation in outsourcing of classifications, including decision tree classification, support vector machine (SVM), and linear classifications. We investigate the possibility of guaranteeing no-outcome-change (NOC) and consider attack models with prior knowledge.

We first conduct our investigation in the context of building decision trees. We propose a piecewise transformation approach using two central ideas of breakpoints and monochromatic pieces. We show that the decision tree is preserved if the transformation functions used for pieces satisfy the global (anti-)monotonicity. We empirically show that the proposed piecewise transformation approach can deliver a secured level of privacy and reduce disclosure risk substantially.

We then propose two transformation approaches, (i) principled orthogonal transformation (POT) and (ii) true negative point (TNP) perturbation, for outsourcing SVM. We show that POT always guarantees no-outcome-change for both linear and non-linear SVM. The TNP approach gives the same guarantee when the data set is linearly separable. For linearly non-separable data sets, we show that no-outcome-change is not always possible and propose a variant of the TNP perturbation that aims to minimize the change to the SVM classifier. Experimental results show that the two approaches are effective to counter powerful attack models.

In the last part, we extend the POT approach to linear classification models and propose to combine POT and random perturbation. We conduct a detailed set of experiments and show that the proposed combination approach could reduce the change on the mining outcome while still providing high level of protection on privacy by adding less noise. We further investigate the POT approach and propose a heuristic to break down the correlations between the original values and the corresponding transformed values of subsets. We show that the proposed approach could significantly improve the protection level on privacy in the worst cases.
# Table of Contents

Abstract .............................................. ii

Table of Contents ..................................... iii

List of Tables ........................................ vi

List of Figures ......................................... vii

Acknowledgements ..................................... ix

Dedication ............................................. x

Statement of Co-Authorship ........................... xi

1 Introduction ......................................... 1
   1.1 The Data-Mining-As-a-Service Model .................. 1
      1.1.1 Three Pillars to Privacy Preservation .......... 2
      1.1.2 Data Custodian Scenario ......................... 3
   1.2 Attack Models ...................................... 4
      1.2.1 General Attack Models .......................... 5
      1.2.2 Attack Models with Prior Knowledge ........... 5
   1.3 Research Problems and Contributions ................. 6
      1.3.1 Decision Tree .................................. 6
      1.3.2 Support Vector Machine ........................ 9
      1.3.3 Combining POT and Random Perturbation ........ 11
   1.4 Literature Review .................................. 13
      1.4.1 Privacy Preserving Data Mining .................. 13
      1.4.2 Secure Data Mining On Distributed Data Sets .... 16
      1.4.3 K-anonymity ................................... 17
   1.5 Roadmap ........................................... 17
   1.6 Bibliography ...................................... 19
# Table of Contents

## 2 Preservation Of Patterns and Input-Output Privacy

2.1 Introduction .................................................. 25  
2.2 Related Studies ............................................. 28  
2.3 Preliminaries ............................................... 28  
  2.3.1 Monotone and Anti-monotone Functions .......... 28  
  2.3.2 Domain, Subspace Association and Pattern Disclosure 29  
  2.3.3 Attack Models Possibly with Prior Knowledge ... 30  
2.4 The No-Outcome-Change Guarantee .................... 31  
2.5 A Piecewise Framework .................................... 33  
  2.5.1 ChooseBP: Choosing Breakpoint Locations Randomly 33  
  2.5.2 ChooseMaxMP: Exploiting Monochromatic Pieces ... 35  
  2.5.3 Selecting Transformations Randomly ............. 38  
  2.5.4 Defense Against Sorting Attacks ................. 40  
2.6 Empirical Evaluation ...................................... 41  
  2.6.1 Experimental Setup ................................... 41  
  2.6.2 Domain Disclosure Risk ............................. 42  
  2.6.3 Subspace Association Risk ......................... 45  
  2.6.4 Output Privacy: Pattern Disclosure Risk ........ 47  
2.7 Future Work ............................................... 47  
2.8 Bibliography ............................................... 48

## 3 Patterns and Privacy Preservation with Prior Knowledge for SVM Classification

3.1 Introduction .................................................. 49  
3.2 Related Work ............................................... 52  
3.3 Knowledge Points and Attack Models ................... 53  
  3.3.1 Knowledge Points .................................... 53  
  3.3.2 Attack Models with Prior Knowledge ............ 54  
3.4 POT: Principled Orthogonal Transformation .......... 57  
  3.4.1 Preliminaries ......................................... 57  
  3.4.2 Limitations of Random Transformations .......... 58  
  3.4.3 Reducing Correlations by POT .................... 59  
  3.4.4 Technical Details and Equations ................. 61  
  3.4.5 Properties of the Algorithm ....................... 63  
3.5 Providing Extra Protection for Linear SVM: True Negative Point (TNP) ......................... 64  
  3.5.1 Linearly Separable Data Sets ..................... 66  
  3.5.2 Linearly Non-Separable Data Sets ................ 70  
3.6 Experiments ............................................... 71  
  3.6.1 ICA Attack ........................................... 72
# Table of Contents

3.6.2 Knowledge Points .................................................. 73
3.6.3 Effectiveness of POT Against Curve Fitting Attacks ............. 74
3.6.4 Effectiveness of TNP Against Global Attacks .................... 77
3.6.5 Minimizing the Outcome Change Against Global Attacks ...... 79
3.6.6 Intended Usage of POT and TNP ................................. 81
3.7 Conclusion and Future Work ......................................... 82
3.8 Bibliography ............................................................ 84

4 Extension to POT ......................................................... 87
4.1 Introduction ............................................................. 87
4.2 Related Works .......................................................... 88
4.3 Preliminaries ........................................................... 89
  4.3.1 Attack Models with Prior Knowledge ............................. 89
  4.3.2 POT: Principled Orthogonal Transformation .................... 90
4.4 Optimizing POT: POT for Linear Sub-groups ....................... 91
  4.4.1 A Motivating Example ............................................ 92
  4.4.2 A Heuristic: POT for Linear Sub-Groups ....................... 92
  4.4.3 Properties of Algorithm POT-LS ............................... 100
4.5 POT for Other Classification Methods ................................ 101
4.6 Combination of POT and Random Perturbation .................... 105
4.7 Experiments .................................................................. 106
  4.7.1 Optimizing POT: POT-LS ....................................... 107
  4.7.2 Combination Of POT and Random Perturbation ................ 112
4.8 Conclusions .................................................................. 120
4.9 Bibliography ............................................................. 123

5 Conclusions and Future Works .......................................... 125
5.1 Conclusions ............................................................... 125
5.2 Future Works ............................................................. 127
5.3 Bibliography ............................................................. 128
List of Tables

1.1 Three Pillars of Privacy Preservation . . . . . . . . . . . . . . 4
2.1 Statistics of Attributes . . . . . . . . . . . . . . . . . . . . . 41
2.2 Sorting Attack: Worst Case . . . . . . . . . . . . . . . . . . 45
2.3 Output Privacy . . . . . . . . . . . . . . . . . . . . . . . . . 47

3.1 Three Pillars of Privacy Preservation . . . . . . . . . . . . . . 50
3.2 Data Sets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
3.3 Correlations . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
3.4 Cost of POT-GenMatrix . . . . . . . . . . . . . . . . . . . . . 77
3.5 Global Attacks on All Data Sets . . . . . . . . . . . . . . . . 78
3.6 Measurement of Classification . . . . . . . . . . . . . . . . . 80

4.1 The NOC Guarantee for Linear Classification . . . . . . . . . 105
4.2 Data Sets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107
4.3 Evaluation of Lga-Posterior . . . . . . . . . . . . . . . . . . 112
List of Figures

1.1 The Data Custodian Model ........................................ 3
1.2 An Example ..................................................... 10
1.3 Combination of POT and Random Perturbation for SVM .......... 12

2.1 A Training Data Set before and after transformation and their corresponding Decision Trees ........... 27
2.2 A Piecewise Framework ......................................... 33
2.3 Failing the Global-Monotone Invariant ......................... 34
2.4 Satisfying the Global-Monotone Invariant .................... 35
2.5 A Skeleton for ChooseBP ....................................... 35
2.6 A Skeleton for ChooseMaxMP .................................. 37
2.7 Maximizing Monochromatic Pieces ............................ 37
2.8 A Skeleton for Randomly Choosing Transformations Piecewise ... 39
2.9 Impact of Breakpoints and Monochromatic Pieces on Domain Disclosure ...................................... 43
2.10 The Venn Diagram of Cracks: the Combination Attack ....... 44
2.11 Subspace Association Disclosure Risk ....................... 46

3.1 Generating Orthogonal Matrix $Q$ ............................. 60
3.2 Finding True Negative Points .................................. 67
3.3 A Skeleton for the TNP Perturbation Approach ............... 68
3.4 Linearly Non-Separable Data Sets ............................... 70
3.5 ICA Attack ..................................................... 72
3.6 Sources of Knowledge Points (Kps) ............................. 73
3.7 Privacy Analysis of POT vs. Rotation .......................... 74
3.8 Impact of the Number of Knowledge Points .................... 75
3.9 WDBC : Radius $\delta$ ............................................. 76
3.10 Global Attack on WDBC ....................................... 77
3.11 Impact of Poor Knowledge Points (WDBC) .................... 79
3.12 Classification Accuracy : TNP v.s. Random Perturbation .... 81

4.1 Examples of Transformations .................................... 92
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Algorithm POT-LS : POT for Linear Sub-Groups</td>
<td>96</td>
</tr>
<tr>
<td>4.3</td>
<td>Examples of Transformations : Algorithm POT-LS</td>
<td>98</td>
</tr>
<tr>
<td>4.4</td>
<td>Lga-Posterior : Identifying More Linear Sub-Groups</td>
<td>99</td>
</tr>
<tr>
<td>4.5</td>
<td>Finding Linear Sub-Groups : Lga-Posterior</td>
<td>99</td>
</tr>
<tr>
<td>4.6</td>
<td>The Combination of POT and Random Perturbation</td>
<td>106</td>
</tr>
<tr>
<td>4.7</td>
<td>POT-LS (in the Worst Cases) : Data Set WDBC</td>
<td>108</td>
</tr>
<tr>
<td>4.8</td>
<td>POT-LS (in the Worst Cases) : Data Set Housing</td>
<td>110</td>
</tr>
<tr>
<td>4.9</td>
<td>POT-LS (in the Worst Cases) : Data Set Credit</td>
<td>110</td>
</tr>
<tr>
<td>4.10</td>
<td>POT-LS (in the Worst Cases) : IBM Synthetic Data Set</td>
<td>111</td>
</tr>
<tr>
<td>4.11</td>
<td>The Combination Approach for SVM : WDBC</td>
<td>113</td>
</tr>
<tr>
<td>4.12</td>
<td>The Combination Approach for SVM : Housing</td>
<td>114</td>
</tr>
<tr>
<td>4.13</td>
<td>The Combination Approach for SVM : IBM Synthetic Data</td>
<td>115</td>
</tr>
<tr>
<td>4.14</td>
<td>The Combination Approach for SVM : Credit</td>
<td>115</td>
</tr>
<tr>
<td>4.15</td>
<td>The Combination Approach for Linear Models : WDBC</td>
<td>116</td>
</tr>
<tr>
<td>4.16</td>
<td>The Combination Approach for Linear Models : Housing</td>
<td>117</td>
</tr>
<tr>
<td>4.17</td>
<td>The Combination Approach for Lasso : WDBC</td>
<td>118</td>
</tr>
<tr>
<td>4.18</td>
<td>The Combination Approach for Elastic Net : WDBC</td>
<td>119</td>
</tr>
<tr>
<td>4.19</td>
<td>The Combination Approach for Lasso : Housing</td>
<td>120</td>
</tr>
<tr>
<td>4.20</td>
<td>The Combination Approach for Linear Models : Housing</td>
<td>121</td>
</tr>
</tbody>
</table>
Acknowledgements

I would like to express my deep thanks to my supervisors Dr. Raymond T. Ng and Dr. Laks V.S. Lakshmanan for their valuable advices and constant encouragement through the long journey of the doctoral program. I sincerely appreciate their continuous guidance, inspiration, and friendship for years.

I would like to thank Dr. Chen Greif, who is in my supervisory committee. Without his excellent advice, this research work would not be done in time. I would also like to thank the university examiners, Dr. Alan Wagner and Dr. Ruben H. Zamar, and the external examiner, Dr. Flip Korn, to review my thesis.

I appreciate the helpful advice from Dr. Rachel Pottinger. I would also like to thank all the members of the Data Management and Mining Lab, including Ed Knorr, Ganesh Ramesh, Hongrae Lee, Michael Lawrence, Jian Xu, Min Xie, and all others who made my research in DMM lab an enjoyable experience.

I would also like to thank all my friends who made my studies at UBC a wonderful experience. I owe my special thanks to Xiaodong Zhou, Fahong Li, Ying Su, Juan Li, Hui Wang, and Kan Cai, for their generous help and support during my hard time.

Finally, I would like to thank my parents, parents-in-law, and other family members for their endless support. I owe my deepest gratitude to my wife, Ming Huo, for her selfless love. They encouraged me to overcome the difficulties and helped me to walk through the darkest time in my life.
Dedication

To my families.
Statement of Co-Authorship

This thesis contains materials that are work and results of joint research under the supervision of Dr. Raymond T. Ng, Dr. Laks V.S. Lakshmanan (chapter two, chapter three, and chapter four), under the instruction of Dr. Chen Greif (chapter three), and in collaboration with Dr. Ganesh Ramesh (chapter two). As the first author of the three manuscript chapters (chapter two, chapter three and chapter four), I took the main responsibility to design the approaches, finished the proofs, implemented the experiments, and conducted the data analysis under the instruction of my supervisors (Dr. Raymond T. Ng and Dr. Laks V.S. Lakshmanan) and Dr. Chen Greif. I also prepared the manuscripts, which are always reviewed by the three supervisory committee members.
Chapter 1

Introduction

Data mining aims to extract predictive information and patterns from the huge amount of data stored in databases or data warehouses to predict future trends and support decision making [22]. In recent years, more and more companies, institutes, organizations would like to conduct data mining tasks on their data to gain useful knowledge. Instead of conducting data mining tasks by data owners themselves, data-mining-as-a-service model has appeared and is becoming more and more popular [29]. There are two main reasons to motivate data owners to outsource data mining tasks to data mining service providers in the data-mining-as-a-service model. First, data owners might not have the required expertise, including parameter tuning and model selection, to conduct data mining tasks [22]. Outsourcing the data mining tasks can save data owners the need to acquire the required expertise or to hire extra data mining experts for mining tasks [45]. Second, data mining tasks might incur heavy computation cost. Instead of purchasing expensive computational resources, data owners might want to outsource data mining tasks to outside mining service providers [55, 64]. Especially with the emergence of cloud computing [59], data mining tasks could be submitted to high performance cloud computing environments [21]. It will become more and more popular to outsource data mining tasks since data-mining-as-a-service could be also provided by cloud computing services [61], e.g., Oracle Data Mining on cloud [68] and IBM Cloud Computing [69].

1.1 The Data-Mining-As-a-Service Model

There are at least two scenarios in the data-mining-as-a-service model. They are the data collector scenario and the data custodian scenario.

In the data collector scenario, a data collector collects data from individual data owners and conducts data mining tasks. For example, all individual users submit their data to a collector for data analysis in an online survey project. The collector might not be trusted by individual data owners. The privacy of the input data should be protected before it is submitted to the data collector.
In the data custodian scenario, the data custodian is trusted. A data custodian either owns the data or is responsible to protect the data. For example, a hospital or medical research institute keeps the patient data and has the responsibility to protect the patients’ private information.

1.1.1 Three Pillars to Privacy Preservation

Privacy preserving data mining is the problem of protecting sensitive information inherent in data while conducting data mining tasks [13, 54]. In order to protect the private information, a perturbation or transformation mechanism is used to disguise the sensitive information. As discussed in [9], there are three pillars to the privacy preservation and they are:

- **input privacy [46]**, which aims to protect the privacy of the input data to data mining tasks.

- **output privacy [46]**, which means to protect the data mining outcome. There are cases that the mined patterns should be protected. For example, a certain disease prediction pattern mined from a medical data set might need to be protected since malicious users could use the pattern to make a prediction on other individuals.

- **minimization of the outcome change.** In order to protect the private information, the original data is always perturbed or transformed [6]. The data mining tasks will not be applied on the original data anymore. Instead, the data mining tasks are applied on the transformed or perturbed data. The discrepancy between the data mining patterns from the original data and the patterns from the perturbed data is called *outcome change*. We say a transformation or perturbation approach provides no outcome change (NOC) guarantee if the patterns mined from the original data and from the perturbed data are identical. There are circumstances under which it may be impossible to guarantee NOC. In this case, a desirable property is minimization of outcome change.

In the data collector scenario, a data collector gathers individual data from different owners and conducts the data mining tasks. The assumption is that the data collector might not be trusted. The individual data, i.e., the input data to the data mining tasks, should be protected. The dominant approach is random perturbation [4, 16], which transforms the original values by adding random noise in a principled way. There is a trade off between
privacy protection and outcome change [4]. In order to provide more protection we need more perturbation on the original data. Meanwhile, more protection means more change to the data mining outcome. The change of data mining outcome is inevitable.

In the data custodian scenario, the data custodian owns the data or is authorized to keep the data. The data custodian is trusted. However, when the data custodian wants to outsource data mining tasks to outside data mining service providers, she needs to transform the original data before she sends the data to the data mining service providers.

1.1.2 Data Custodian Scenario

Figure 1.1 shows the framework of the data custodian model. When a data custodian wants to outsource data mining tasks, she transforms the original data set $D$ to the transformed data set $D'$ by a transformation $f$. The transformed data set $D'$ is submitted to the data mining service provider, who will conduct the data mining tasks on the transformed data set $D'$.

After the service provider gets the data mining pattern $P'$, she will send $P'$ back to the data custodian. So far the data mining pattern $P'$ is from the transformed data set $D'$, which means $P'$ is in the encrypted form. The data custodian needs to decode the encrypted pattern $P'$ to get pattern $P$, which is the final mining outcome the data custodian could obtain. Let us suppose the pattern $P_0$ be the pattern we could discover directly from the original data set $D$. The question is whether the decoded pattern $P$ equals to the original pattern $P_0$. If they are different, then the change between $P$ and $P_0$ should be minimized. This is the third pillar of privacy preserving data mining, i.e., minimizing the outcome change.

Figure 1.1: The Data Custodian Model
### 1.2. Attack Models

The data mining service provider only has the transformed data set \( D' \) and cannot access the original data set \( D \). Therefore, the original data \( D \) is protected by the transformation \( f \). This is how we achieve the input-privacy. Meanwhile, the data mining service provider only has the pattern \( P' \), which is in encrypted form and is different from the true pattern \( P_0 \). Thus the pattern is also protected. This is output-privacy.

We could also use random perturbation approach to perturb the original data in the data custodian scenario. Since the outcome change is inevitable for the random perturbation approach, the question is whether we could design any other transformation that could provide the NOC guarantee while not sacrificing the input and output privacy. Several recent studies show that it is possible to provide the no-outcome-change guarantee while also providing input and output privacy in the data custodian scenario. For example, Wong et al. studied the problem of secure outsourcing of association rule mining in [55] and proposed an algorithm to give the NOC guarantee while also providing input and output privacy.

Table 1.1 highlights the differences between the data collector scenario and the data custodian scenario regarding the three pillars of privacy preservation.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Studies</th>
<th>Input Privacy</th>
<th>Output Privacy</th>
<th>NOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Collector</td>
<td>[2, 4, 5, 16]</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Custodian</td>
<td>[10, 11, 35, 55]</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>K-Anonymity</td>
<td>[25, 39, 49]</td>
<td>√</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1: Three Pillars of Privacy Preservation

### 1.2 Attack Models

Privacy protection is always discussed together with attack models, which define how much information a hacker could have to crack the encrypted data [18].

\(^1\)Molly et al. show that the proposed approach in [55] could not counter attacks with known frequency knowledge in a recent paper [42].
1.2.1 General Attack Models

Generally, there are four types of attack models [18], and they are:

- Ciphertext-only attack: which means a hacker could only access the ciphertext;

- Known-plaintext attack: which means a hacker could have samples of the plaintext and the corresponding encrypted version of those samples, i.e., the ciphertext. She could use those samples to further learn other secret information;

- Chosen-plaintext attack: which means a hacker could freely select plaintexts to encrypt and get the encrypted version of the plaintexts;

- Chosen-ciphertext attack: which means a hacker could choose a ciphertext and get the corresponding decryption version (i.e., the plaintext) of this ciphertext.

In the data custodian model, as shown in Figure 1.1, a data custodian only sends the transformed data set $D'$ to data mining service providers. Both the mining service providers or other outside hackers could only access the transformed data set $D'$. Thus the third attack model and the forth attack model, i.e., chosen-plaintext attack and chosen-ciphertext attack, are not suitable for the data custodian model.

Moreover, the ciphertext-only attack is the simplest attack model. It is difficult to attack a transformation with only transformed data set [6, 48]. In this thesis, we mainly consider the second attack model, i.e., the known-plaintext attack, since it is practical for the data custodian model.

1.2.2 Attack Models with Prior Knowledge

In the known-plaintext attack model, a hacker knows samples of the plaintext and the corresponding ciphertext. In the data custodian scenario, it means that a hacker knows a group of pairs of the original data point and the corresponding transformed data point. Those pairs of data points are regarded as prior knowledge owned by the hacker.

A hacker could gain this kind of prior knowledge from various sources. For example, a hacker could know statistic values (e.g., the minimum, the maximum, or the mean, etc) from published statistics or she could get some samples from similar data sets (e.g., a rival company targeting the same market). In order to capture all these forms of prior knowledge, in this thesis
we define an abstract notion knowledge point that is a pair of an original data point and the corresponding transformed data point. This attack model is also called input-output point based attack in [6, 11]. However, in practice, it might be difficult for a hacker to know the exact values in the original data set \( D \). More practically, a hacker’s knowledge about the original data might fall in a small range around the true values. For example, given the minimum of the age attribute is 19, a hacker might think the minimum is 18 from published information, which is close enough to the true minimum value to be regarded as a threat. This definition does not weaken the abilities of a hacker to crack the transformed data. Indeed, it makes the attack more practical and more meaningful.

In this thesis, we evaluate our proposed transformations in the context of attack models with prior knowledge (i.e., knowledge points). A common attack method is to use the knowledge points to fit a curve to crack the transformed data, which is called curve fitting attack. Such methods include linear fitting, polyline fitting, spline fitting, etc. Another possible attack method with knowledge points is to crack the transformation matrix [10, 35] if the original data is transformed by a matrix.

1.3 Research Problems and Contributions

Classification is a data mining task that builds classifiers from a training data set for future prediction [22]. In this thesis we focus our study on privacy protection under attack models with prior knowledge, for outsourcing of classification, in the data custodian scenario. Our goal is to design transformation methods to transform the original data to provide the input and output privacy while minimizing the outcome change.

1.3.1 Decision Tree

We start our study on investigating in privacy preserving decision tree classification. We propose monotone and anti-monotone functions to transform the original data set \( D \). Given a monotone or anti-monotone function \( f \) and an attribute \( A \) in \( D \), the transformed data is \( A' = f(A) \). Because (anti-)monotone functions are invertible, decoding the mining outcome (i.e., the pattern \( P' \) in Figure 1.1) is straightforward. The decoded pattern is \( P = f^{-1}(P') \). Moreover, (anti-)monotone functions preserve the decision tree in an exact sense (shown in Chapter 2), which means the decoded pattern \( P \) is identical to the pattern \( P_0 \) discovered from the original data set \( D \) directly.
1.3. Research Problems and Contributions

For example, let us suppose \( D \) contains an attribute ‘age’, whose domain is from 18 to 80. We transform the attribute ‘age’ by a monotone function

\[
f(x) = 0.5 \times x + 30.
\]

Suppose there is a pattern \( P_0 \) from \( D \) as following:

\[
P_0 = \{(\text{age} \leq 30) \lor (\text{age} > 50), \text{Class} = H; (30 < \text{age} \leq 50), \text{Class} = L\}.
\]

After transformation, the mined pattern will be

\[
P' = \{(\text{age} \leq 45) \lor (\text{age} > 55), \text{Class} = H; (45 < \text{age} \leq 55), \text{Class} = L\}.
\]

The values have been transformed by the function \( f \). Once the data custodian gets this pattern, she decodes it by the invert function

\[
f^{-1}(x) = 2 \times (x - 30)
\]

and obtains the decoded pattern

\[
P = \{(\text{age} \leq 30) \lor (\text{age} > 50), \text{Class} = H; (30 < \text{age} \leq 50), \text{Class} = L\},
\]

which is identical to the true original pattern \( P_0 \).

Meanwhile, the data service provider only have the transformed data set \( D' \) and the encrypted pattern \( P' \). The input and output privacy are also achieved.

There are several key advantages that these (anti-)monotone functions enjoy over random perturbations.

- First, both the original data and the decision tree (mining outcome) are expressed in the transformed domain. In other words, both the data and the mining outcome are protected, which means the two pillars (i.e., input and output privacy) are provided.

- Second, with monotone and anti-monotone functions, the decision tree is exactly preserved. Also, decoding of the mining outcome using the reverse transformations by the data owner is easy. Undoubtedly, the data owner would like to minimize the effort required for the decoding process.

- Third, with the proposed transformations, every data value is transformed. In contrast, with random perturbations, there is a chance that a data value is not changed and the true value is revealed. For example, in [9, 17], many situations examined leave a significant percentage (e.g., 25%) of items unchanged.
1.3. Research Problems and Contributions

A key advantage of (anti-)monotone functions is that they provide no-outcome-change while also providing input and output privacy. However, if we only use one (anti-)monotone function to transform an attribute of a data set, the input or output privacy might not be effectively protected, especially when a hacker has prior knowledge. For example, for the above transformation function $f = 0.5 \times x + 30$, if a hacker happens to know that the transformed value 40 corresponds to 22 and 50 corresponds to 42, then she could make a guess on the transformation function and guesses $g = 0.5 \times x + 29$, which is too close to the true transformation $f = 0.5 \times x + 30$.

In order to enhance the protection on privacy, we propose a transformation schema that generalizes from (anti-)monotone functions to piecewise (anti-)monotone functions. The new transformation schema provides two extra levels of protection against attacks with prior knowledge. First, we use breakpoints to break the domain of an attribute into several pieces. Each piece of the domain has its own (anti-)monotone function. Second, we identify the monochromatic pieces in a domain. A monochromatic piece is a piece whose elements are associated with the same class label. The monochromatic piece gives us more freedom to choose transformation functions. We show that we could even use any arbitrary function to transform a monochromatic piece while still guaranteeing no-outcome-change in chapter two.

The new transformation framework with piecewise (anti-)monotone functions and monochromatic pieces injects three extra levels of uncertainty to enhance the protection on privacy:

- **The number of breakpoints**: The data custodian can arbitrary select the number of breakpoints to break an attribute into pieces.

- **The location of breakpoints**: How to break an attribute into pieces is not determined. That is the location of breakpoints could also be randomly selected.

- **The function used in each piece**: After breaking an attribute into pieces, the data custodian could select different (anti-)monotone function for each piece.

Finally, even if a hacker can crack one piece, he may not be able to crack the remaining pieces. However, while breakpoints enhance input and output privacy, the challenge is to add breakpoints in a way that the no-outcome-change guarantee is provided, which is presented in chapter two.
1.3. Support Vector Machine

In chapter three, we study the problem of outsourcing support vector machine (SVM) classification. SVM aims to find the optimal classifier that minimizes the classification errors and separates the rest of the elements with maximal margin on the train data set in the feature space [8].

For decision tree classification, we start investigating (anti-)monotone functions because those functions provide the NOC guarantee. Same to the SVM classification, we begin with transformations that could preserve support vectors, which determines the optimal classifier.

Figure 1.2 is a simple example of the transformation that can preserve the support vectors. The table shown in Figure 1.2(a) is the original dataset $D$. There are two attributes #A and #B. $p_i$ is tuple ID. Each tuple is associated with a class label. We represent $D$ as a $2 \times 6$ matrix, i.e., each tuple is a column vector that can be regarded as a point in two-dimensional space. Figure 1.2(c) shows the data points. $h$ is the optimal classifier and points $p_3, p_4$ and $p_5$ are support vectors.

We transform the original data $D$ with an orthogonal matrix $Q = \begin{pmatrix} 0.8660 & 0.5000 \\ -0.5000 & 0.8660 \end{pmatrix}$, which means $D' = Q \times D$. $Q$ is also a rotation matrix and the geometric meaning is to rotate the whole dataset $\pi/6$ clockwise. The transformed dataset is shown in Figure 1.2(b). Figure 1.2(d) shows the transformed data points and the mining result found by the mining service provider. $h'$ is the optimal plane and $p'_3, p'_4$ and $p'_5$ are support vectors in $D'$. To decode each support vectors in $D'$, the data custodian uses the inverse transformation $p_i = Q^{-1} \times p'_i$.

By comparing Figure 1.2(c) and Figure 1.2(d), we can see that a transformed data point is a support vector in the transformed data set $D'$ if and only if its corresponding original data point is a support vector in the original data set $D$. In other words, the support vectors are preserved by the orthogonal transformation $Q$.

Orthogonal transformation includes rotation and reflection. Both rotation and reflection preserve the geometric shape of the data set and preserve the distance between any two points. Intuitively, orthogonal transformation will preserve the support vectors.

In previous work, Chen et al. propose to use random rotation to transform the original data and show that rotation provides the NOC guarantee for outsourcing SVM [10, 11]. However, random rotation could not provide enough protection on privacy especially when a hacker has prior knowledge. The reason is that the original values and the transformed values may
1.3. Research Problems and Contributions

(a) Original Data $D$

<table>
<thead>
<tr>
<th>id</th>
<th>#A</th>
<th>#B</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$p_2$</td>
<td>5</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>$p_3$</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$p_4$</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$p_5$</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>$p_6$</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) Transformed Data $D'$

<table>
<thead>
<tr>
<th>id</th>
<th>#A</th>
<th>#B</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p'_1$</td>
<td>1.366</td>
<td>0.366</td>
<td>0</td>
</tr>
<tr>
<td>$p'_2$</td>
<td>6.330</td>
<td>0.964</td>
<td>1</td>
</tr>
<tr>
<td>$p'_3$</td>
<td>3.232</td>
<td>1.598</td>
<td>0</td>
</tr>
<tr>
<td>$p'_4$</td>
<td>3.598</td>
<td>0.232</td>
<td>0</td>
</tr>
<tr>
<td>$p'_5$</td>
<td>5.464</td>
<td>1.464</td>
<td>1</td>
</tr>
<tr>
<td>$p'_6$</td>
<td>5.964</td>
<td>2.330</td>
<td>1</td>
</tr>
</tbody>
</table>

(c) Original SV

(d) Transformed SV

Figure 1.2: An Example

have high correlation after rotation transformation. Intuitively, the higher the correlation between the original values and the transformed values, the smaller the fitting error for curve fitting methods (e.g., linear regression, polyline, spline fitting, etc.), which means the transformation is more vulnerable to curve fitting attacks.

In order to enhance the privacy protection level, we propose an approach, i.e., principled orthogonal transformation (POT), to generate orthogonal transformations in a principled way. POT is based on the Gram-Schmidt procedure [50] and could iteratively and monotonically reduce the linear correlation between the original values and the transformed values. We will provide empirical results to show that POT can effectively counter curve fitting attacks.

Besides curve fitting attacks, a hacker could conduct a more powerful
attack on the transformation matrix if she has enough knowledge points. Given a $d$-dimensional data set, the orthogonal transformation is a $d \times d$ matrix. If a hacker has $d$ knowledge points, then she could make a guess on the transformation matrix by matrix computation. We call this type of attack global attack.

In order to further enhance the protection on privacy to counter global attacks, we propose an algorithm that identifies the data points that are guaranteed not to be support vectors. We call those data points True Negative Points (TNP). In support vector machine classification, support vectors determine the optimal classifier. Perturbation on the true negative points will not affect the support vectors and thus will not change the optimal classifier. Therefore, we can could enhance the privacy level while not sacrificing the classification accuracy by perturbing the TNP points.

In chapter three, we show that the NOC guarantee is provided for a linearly separable data set. However, for a linearly non-separable data set, because it is not clearly about the geometric meaning of the support vectors and the optimal classifier [8], the points identified by the algorithm might be false negative points. Thus, the NOC guarantee may be violated but the change is minimized.

1.3.3 Combining POT and Random Perturbation

Random perturbation is a well studied method for privacy preserving data mining. There is always trade off between the privacy protection level and data mining outcome change for random perturbation, i.e., the more perturbation on the data, the more protection on the privacy, the more change on the data mining outcome [4]. On the other hand, as discussed above, the POT approach has two advantages in privacy preservation of SVM classification. First, POT could enhance protection on privacy by reducing the correlation between the original values and the transformed values. Second, POT provides no-outcome-change guarantee. Therefore, the combination of the POT approach and random perturbation should be able to increase the protection level while minimizing the outcome change. The last part of this thesis proposes a transformation schema that transforms the original data sets in two steps, which is shown in Figure 1.3.

First, the random perturbation approach is applied. Random noise $N$ is added to the original data set $D$. The perturbed data set is $D' = D + N$. Second, we use POT to generate an orthogonal transformation $Q$ to transform the perturbed data $D'$ and get the transformed data set $D'' = Q \times D'$. Then the data mining service provider will conduct the data mining
1.3. Research Problems and Contributions

Figure 1.3: Combination of POT and Random Perturbation for SVM

tasks on the transformed data set $D''$ and discover pattern $P'$. Finally, after decoding, we could get pattern $P_2$, which is identical to the pattern $P_1$ mined from the perturbed data set $D'$. Let us still use $P_0$ to represent the pattern mined directly from the original data set $D$. The goal of the proposed transformation schema is to enhance the protection on privacy while minimize the difference between the pattern $P_2$ (equals to $P_1$) and the pattern $P_0$.

In order to further enhance the protection on privacy, we execute a worst case analysis on the POT approach. We notice that even though the POT approach provide high level protection in average cases, there are cases in which a hacker could still guess out a substantial number of the original values based on her prior knowledge. We found that although POT has reduced the correlation for an attribute to be below a given threshold, there might exist a subset of values that still have high correlation with the corresponding transformed values. When a hacker’s knowledge points happen to be located in this vulnerable subset, this hacker might be able to have a good guess on the original values in this subset. In order to enhance the protection level in the worst cases, we propose a heuristic approach to break down the correlation between the original values and the transformed values among those vulnerable subsets.

We will provide empirical results to show that the proposed transformation scheme, i.e., the combination of the POT approach and the random perturbation, could effectively improve the protection level on privacy and minimize the outcome change.
1.4 Literature Review

1.4.1 Privacy Preserving Data Mining

In order to protect private information about individuals while answering queries in statistical databases, several approaches have been proposed, including access restriction, query set restriction, micro-aggregation, data perturbation, output perturbation, auditing, random sampling, etc [1]. In the data-mining-as-a-service model, privacy also needs to be protected since the service provider, who conducts the data mining tasks, might not be trusted by the data owner [13, 54].

Random Perturbation

Random perturbation is a widely studied approach to transform the original values by adding random noise in a principled way [2, 4, 15, 16, 17, 23, 30, 67]. As discussed in Section 1.1.2, the random perturbation approach is designed for the data collector scenario and mainly focuses on input privacy (Table 1.1).

Agrawal and Srikant propose a random perturbation approach for privacy preserving decision tree classification [4]. Random noise drawn from a given distribution is added to the original data. The data miner could build the classifier by reconstructing the distribution of the original data from the perturbed data and the distribution of the noise. Privacy level corresponds to the width \((x_2 - x_1)\) if the value \(x\) can be estimated within the interval \([x_1, x_2]\). In [2], Agrawal and Aggarwal consider a different privacy measure based on differential entropy. They propose an EM algorithm for computing the reconstructed distribution which converges to the maximum-likelihood estimate of the original distribution.

Random perturbation has also been applied to association rule mining by Evfimievski et al. in [16, 17]. In [46], Rizvi and Haritsa propose a variant perturbation approach based on probabilistic distortion. Instead of adding random noise on the original values, they apply the Bernoulli principle of keeping an item in a transaction with probability \(p\), but deleting the item with probability \((1 - p)\).

It is shown that random perturbation could be vulnerable to matrix-based attacks. In [30], Kargupta et al. show that the added random noise could be separated from the real values by using spectral analysis based matrix decomposition techniques. In [23], Huang et al. present a method based on principal component analysis that could reconstruct the original data by exploiting correlations among attributes.
1.4. Literature Review

Aggarwal and Yu [5] propose a condensation approach, which uses a anonymized data set to replace the original data set, to protect the private information. The anonymized data set has similar statistic characteristics to the original data set. The data mining task is conducted on the anonymized data set. Same as random perturbation, the mining outcome is changed.

In [66], Zhang et al. present a new scheme based on algebraic-technique to build a more accurate classifier with less disclosure. This study is also based on the data collector scenario. Like the studies based on random perturbation, the mining results are changed by the perturbation.

In summary, the random perturbation approach is designed for the data collector scenario. Its goal is to protect the private information inherent in the original data, i.e., the input privacy. It is shown that it is inevitable that there is a tradeoff of privacy and classification accuracy. Generally, more noise added, more protection on the private information, but meanwhile less accuracy of the classifier. Meanwhile, the data miner and the data owner have the same forms of the data mining outcome. The output privacy is not a stated design objective. In the data custodian scenario, we aim to propose transformation approaches that could deliver the three pillars of privacy preservation, whereas the perturbation approach cannot.

Geometric Data Transformations

Chen et al. propose a rotation-based transformation approach for SVM [10]. In order to enhance the protection level, the proposed approach swaps the rows of a rotation matrix. The approach selects the permutation that gives the highest protection level, which is defined by a unified column privacy metric. For a \(d\) dimensional data set, the rotation matrix is a \(d \times d\) matrix. Therefore, the complexity to select the best permutation of the rows is \(d!\). Meanwhile, the selection of the best permutation could only find the best solution for the current rotation matrix, which is called local optimization in the paper. There is no guarantee on the privacy protection level. The independent component analysis (ICA) is used as attack model to evaluate the protection level. However, ICA assumes that the hacker has no prior knowledge. Therefore, ICA belongs to the ciphertext-only attack [18]. In their following work [11], except the ICA attack model, they introduce more powerful attack models, such as distance-inference attacks, in which a hacker has prior knowledge to crack the rotation matrix. Their proposed approach amounts to adding noise to the transformed data; but this sacrifices classification accuracy.

Liu et al. study random orthogonal transformations from an attacker’s
1.4. Literature Review

view for k-means and k-NN clustering [35]. However, like random rotation, random orthogonal transformation is vulnerable to attacks with prior knowledge.

In [44], Oliveira et al. use affine transformations, such as scaling, rotation and translation, to transform the original values for privacy preserving clustering. However, different attributes are subjected to different transformations. The geometric properties of the data set, e.g., the distance between two points, are not preserved. Therefore, the data mining outcome, i.e., the clusters, are changed. Meanwhile, like random rotation, random affine transformation is vulnerable to attacks with prior knowledge as well.

Comparing to the random rotation transformation, the POT approach proposed in this thesis could iteratively and monotonically reduce the correlation between the original values and the transformed values in a principled way. Empirical study shows that POT could effectively counter attacks with prior knowledge.

Other Methods

Wong et al. study the problem of secure outsourcing of association rule mining [55]. The propose an encryption approach to protect the original values. The approach is based on a one-to-n mapping and ensures correct decryption. They share with us the goal of providing the NOC guarantee. However, Molly et al. show that the encryption approach in [55] is not secure and can be broken if a hacker knows the frequency knowledge [42].

In [56], Wong et al. propose an audit approach for secure outsourcing frequent item mining. The approach protects the true item values by inserting false items.

Wong et al. also study the problem of privacy protection about k-nearest neighbor (kNN) computation on encrypted databases in [57]. Their approach preserves the orders of the distance between data points instead of the exact the distance. Therefore, the proposed approach is not suitable for classification methods, such as the support vector machine classification.

Mozafari and Zaniolo propose an algorithm for secure view publishing that could preserve naive Bayesian classifiers while protecting privacy[43].

All the above work have little overlap with this thesis. We mainly focus our study on privacy preservation in outsourcing of classifications, including decision tree classification and the SVM classification.

Lin and Chen study how to secure publish the SVM classifier while not breaching sensitive information [34]. Their approach modifies the SVM classification algorithm so that the mining outcome is different from but
1.4. Literature Review

close to the true support vectors. This work assumes the data owner is able to conduct the SVM computation and wants to publish the SVM classifier, which is different from the goal of this thesis.

1.4.2 Secure Data Mining On Distributed Data Sets

When multi-parties have their own copies of data and would like to conduct data mining tasks on the integrated data set, secure mechanism should be designed to make sure that each site cannot gain additional information about the data via the collaboration [12]. The dominant approaches is to design security protocols based on secure multi-party computation (SMC) [20, 52, 62, 63]

Kantarcioglu and Clifton propose security protocol for association rule mining on horizontally partitioned data [28]. Secure association rule mining on vertically partitioned data is studied and security protocol is proposed in [51] by Vaidya and Clifton.

Security protocols for privacy preserving decision tree classification have been proposed for vertically partitioned data by Vaidya and Clifton in [53] and for horizontally partitioned data by Samet and Miri in [47].

Vaidya and Clifton propose security protocol for $k$-means clusters mining over vertically partitioned data [52]. Inan et al. propose an approach to compute dissimilarity matrix of objects for privacy preserving cluster over horizontally partitioned data [24]. Jagannathan and Wright propose a new security protocol for arbitrarily partitioned data, which is a generalization of vertically and horizontally partitioned data [26].

Yu et al. study the problem of secure SVM computation on distributed sites. They propose secure and scalable protocols to train SVM classifier on horizontally partitioned data in [62] and on vertically partitioned data in [63]. Laur et al. [31], Mangasarian et al. [37, 38], and Zhan et al. [65] also propose different security protocols to build support vector machine on distributed sites.

Yang et al. use a SMC-based cryptographic approach to protect the customer’s privacy [60] for decision tree classification or frequent item mining. The proposed approach preserves the data mining outcome (i.e., the NOC guarantee) by preserving the frequency of items.

The security protocols designed for data mining on distributed data sets are not suitable for outsourcing data mining in the data custodian model. The relationship between the data custodian and the data mining service provider is not a collaboration. Meanwhile, the data custodian is the only site which has the data.
1.4.3 K-anonymity

Privacy is also needed to be protected while publishing data for public benefit or research. The notion of $k$-anonymity is designed to protect the sensitive information inherent in the original data for data publishing [49]. The idea is to use suppression or generalization to anonymize items so that a tuple could not be distinguished from a group of size $k$. There are variants of $k$-anonymity, e.g., $l$-diversity [39], $t$-closes [36].

Meyerson and Williams prove that $k$-anonymity is NP-hard in general and propose a couple of approximations [41]. Bayardo and Agrawal study the complexity of finding optimal $k$-anonymity in [7] and propose an algorithm to reach the optimal anonymization in certain circumstance. LeFevre et al. propose a set of more efficient algorithms that reach $k$-anonymity by full-domain generalization [32]. Full domain generalization maps the entire domain of the original attributes to a more general domain. Jiang and Clifton study the $k$-anonymity problem for vertically partitioned data and propose a security protocol to ensure each site could not learn anything that violates $k$-anonymity about other data sites [27].

Iyengar uses generalization and suppression to anonymize original values to protect the sensitive information in the context of data mining [25]. While protecting the privacy, the proposed approach aims to preserve the usefulness of the anonymized data in the context of classification and regression.

Wang et al. propose a bottom-up method for generalization [58] and Fung et al. propose a top-down specialization approach [19] to anonymize the original data set while trying to remain the classification models. Lefevre et al. [33] also propose anonymity approaches while trying to provide good data mining results. The proposed methods provide different anonymization for different data mining tasks.

Similar to the perturbation approach in the data collector scenario, those notions and approaches in k-anonymity are mainly designed to protect input privacy (Table 1.1). The mining outcome changes when data mining methods are applied on the anonymized data.

1.5 Roadmap

The rest of this thesis is organized as follows. Chapter two presents the transformation schema of piece-wise (anti-)monotone functions for secure outsourcing of decision tree classification. Chapter three shows our work on how to provide the three pillars of privacy preservation for the SVM classi-
1.5. Roadmap

fication. The transformation framework that combines the POT approach and random perturbation is shown in chapter four. Finally, this thesis is concluded in chapter five.
1.6 Bibliography


1.6. Bibliography


1.6. Bibliography


1.6. Bibliography


1.6. Bibliography


Chapter 2

Preservation Of Patterns and Input-Output Privacy\textsuperscript{2}

2.1 Introduction

One of the key motivations for privacy preserving data mining is the mining-as-a-service model, in which there are at least two different scenarios. In [2], Agarwal and Srikan consider the data collector scenario where individual data owners selectively submit their data to a data collector, who may not be trusted, in exchange for the value coming from mining the data collection. Many existing studies center on this.

In this chapter, we consider a different, yet equally prevalent, scenario of a data custodian which either owns the data (e.g., a company owns its data) or is explicitly given the trust and responsibility of protecting the privacy of the individuals. As an example of the latter, a medical research group (i.e., the custodian) obtains consent forms from patients to participate in a biomarker study. The data custodian consider hiring a company to help mine the data. It does not implicitly trust the company and wants to guard against possible privacy breaches. To do so, the data custodian needs to transform its data. To determine the appropriate transformation, there are two critical considerations: minimizing disclosure versus minimizing outcome change (i.e., the deviation between the outcome obtained by mining the original data and the outcome of mining the transformed data).

Among the transformations studied in the literature, random perturbation is a dominant approach, i.e., transforming data values by adding random noises in a principled way [2]. The more noises are added, the more disclosure is minimized – but the more the mining outcome is changed. It is implicitly assumed that to minimize disclosure, it is inevitable that there be outcome change. This may be inevitable for the data collector model.

\textsuperscript{2}A version of this chapter has been published. Bu, S., Lakshmanan, L.V.S., Ng, R.T. and Ramesh, G.: Preservation of Patterns and Input-Output Privacy. ICDE 2007, pp. 696–705.
2.1. Introduction

In this chapter, we show that it is possible to minimize disclosure while guaranteeing no outcome change for the data custodian model. We conduct our investigation in the context of decision tree classifiers. We propose classes of monotone and anti-monotone functions, that preserve the decision tree in an exact sense. For privacy preserving data mining, we claim there are three equally important pillars. The no-outcome-change guarantee is the first pillar. The second pillar is the protection of the input data – input privacy in [9]. The third pillar is the protection of the mining outcome – output privacy. Existing studies focus on input privacy, but ignore the other two pillars. We show that using (anti-)monotone transformations can satisfy all three pillars at the same time. Moreover, when (anti-) monotone transformations are used, decoding the mining outcome is straightforward. Finally, with the proposed transformations, every data value is transformed. In contrast, with random perturbations, there is a chance that a data value is not changed and the true value is revealed.

Figure 2.1 shows a simple example of a monotone transformation. The original data $D$ is shown in (a). The attributes $\text{age}$ and $\text{salary}$ are transformed with the linear monotone functions: \[ \text{age}' = 0.9 \times \text{age} + 10, \] and \[ \text{salary}' = 0.5 \times \text{salary}. \] The transformed data $D'$ is now given to the mining service provider (Figure 2.1(b)). The classifier $T'$ based on $D'$ is shown in (c). Note that $D'$ provides input privacy, whereas $T'$ provides output privacy. To decode $T'$ to get the real decision tree, the data custodian uses the inverse function per attribute to obtain $T$, shown in (d). The inverse transformations are \[ \text{age} = (\text{age}' - 10)/0.9, \] and \[ \text{salary} = \text{salary}'/0.5. \] Notice that this is exactly the same outcome if the decision tree algorithm were to be applied to $D$ directly without the transformation. Also note that $T'$ looks realistic enough that a hacker may not even know that it is encoded.

A key analytical result of this chapter is the theorem guaranteeing that there is no outcome change when using the proposed functions. But what about the other two pillars? It should be obvious that for many situations, simply using (anti-) monotone functions may not be too effective for privacy protection. One of the key technical contributions here is the generalization from (anti-) monotone functions to piecewise (anti-) monotone functions. The two central ideas are to introduce breakpoints and to exploit monochromatic pieces. By going piecewise, three levels of uncertainty are added: (i) the uncertainty of the number of breakpoints, (ii) the uncertainty of the locations of the breakpoints; and (iii) the uncertainty of the function used in each piece.

We analyze disclosure risk performance (second and third pillars) with a comprehensive framework. First, we explore different notions of input
and output disclosure. For input, we consider domain disclosure and subspace association disclosure. For output, we consider outcome disclosure, which protects paths of a decision tree. Second, we consider different attack models for the hacker, such as sorting, linear regression, and curve fitting techniques. Third, we consider a hacker’s prior knowledge, modeled as knowledge points in the attribute domain. Based on benchmark data sets, we provide empirical results showing the effectiveness of the proposed framework.
2.2 Related Studies

Extensive research has been done in statistical databases to provide statistical answers without breaching sensitive information about individuals. A commonly used technique is random perturbation [1]. The study by Agrawal and Srikant shows how a decision tree can be built on data perturbed with random noise added in a principled way [2]. However, preserving the exact decision trees is not their goal, and they show that there is a tradeoff between classification accuracy and privacy level.

Recall the distinction between the data collector and data custodian scenarios. The random perturbation approach is designed to deal with the former, but can be applied to the latter as well. The piecewise monotone framework proposed here is designed for the data custodian scenario. Under this scenario, the proposed model delivers the three pillars of privacy preservation, whereas the perturbation approach cannot.

Evfimievski et al. [5] apply the random perturbation approach to association rule mining. Rizvi and Haritsa propose a variant based on probabilistic distortion [9]. The mining outcome is changed. Output privacy is not a stated design objective.

Recently, Kargupta et al. showed that a hacker can use spectral analysis based matrix decomposition techniques to separate the random noises from the real values [7]. Huang et al. [6] use a principal component analysis based method to exploit possible correlations among attributes to reconstruct original data, demonstrating that more accurate individual data can be revealed than originally thought!

Clifton and others consider privacy preservation with vertically partitioned data [11]. The focus is on developing protocols to make sure that each site cannot gain additional information about the data via the collaboration. There are many studies on various other mining tasks with privacy preservation as the main goal, including clustering [11], order-preserving comparisons [3], and data exchange [10]. For data exchange, the notion of k-anonymity is designed for input privacy. If the transformed data were mined directly, the mining outcome could be significantly affected.

2.3 Preliminaries

2.3.1 Monotone and Anti-monotone Functions

The training data set is a relation instance \( D \) with \( m \) attributes \( A_1, \ldots, A_m \), and a categorical class label attribute \( C \). Throughout this chapter, we are
interested in active domains of attributes, which contain values of attributes appearing in a given data set. We denote the active domain of an attribute \( A \) by \( \delta(A) \). Often we will transform it into another active domain \( \delta'(A) \).

Let \( A \) be a numeric attribute with a linear ordering \(<\). Let \( f : \delta(A) \rightarrow \delta'(A) \) be a function. Then \( f \) is monotone (resp., anti-monotone) if for every \( x, y \in \delta(A), x < y \) implies \( f(x) < f(y) \) (resp., \( f(x) > f(y) \)). For a (anti-)monotone function \( f \), the inverse \( f^{-1} \) is always well-defined. For the \( m \) attributes \( A_1, \ldots, A_m \), let there be \( m \) (anti-)monotone functions \( f_1, \ldots, f_m \).

Given a tuple \( \langle t, c \rangle \) (\( c \) being the class label) in \( D \), the tuple is transformed from \( \langle t.A_1, \ldots, t.A_m, c \rangle \) to \( \langle f_1(t.A_1), \ldots, f_m(t.A_m), c \rangle \). We use \( \vec{f} \) to denote the vector of the \( m \) transformations. Let \( D' = \{ \langle \vec{f}(t), c \rangle \mid \langle t, c \rangle \in D \} \) be the data set consisting of these transformed tuples. We refer to a tuple \( \langle t.A, c \rangle \) as an \( A \)-projected tuple, i.e., it retains the \( A \)-value and the class label. Whenever there is no confusion, we refer to an \( A \)-projected tuple as simply a tuple.

### 2.3.2 Domain, Subspace Association and Pattern Disclosure

Input privacy [9] refers to the protection of \( D \) and is measured using various metrics [2, 5], domain disclosure being the popular one.

**Definition 2.1.** Let \( f : \delta(A) \rightarrow \delta'(A) \) be a transformation. A domain crack function \( g : \delta'(A) \rightarrow \delta(A) \) represents the guess the hacker makes on each transformed value. For a value \( v' \in \delta'(A) \) that appears in \( D' \), a guess is a crack if the guess falls within a radius \( \rho \) from the actual value, i.e., \( |g(v') - f^{-1}(v')| \leq \rho \). The domain disclosure risk is the fraction of the number of cracks to the number of distinct values of \( A' \) appearing in \( D' \).

However, for many applications, the data custodian might care more about the association between domain values rather than domain disclosure. For example, for an insurance application, the company cares more about protecting Bob of age 45 earning 50K, rather than the individual values of age or salary. We refer to this as subspace association disclosure.

**Definition 2.2.** Let \( S \subseteq \{A_1, \ldots, A_m\} \) be a subset of attributes of the input training data. For simplicity, let \( S = \{A_1, \ldots, A_s\} \). A subspace crack function \( \vec{g} : (\delta'(A_1) \times \cdots \times \delta'(A_s)) \rightarrow (\delta(A_1) \times \cdots \times \delta(A_s)) \) represents the guess the hacker makes on each transformed \( S \)-tuple. A guess is a (\( S \)-tuple) crack if \( \bigwedge_{i=1}^s |\vec{g}(v'_i) - \vec{f}_i^{-1}(v'_i)| \leq \rho_i \) (radius for \( A_i \)). The subspace association disclosure risk is the fraction of the number of cracks to the number of \( S \)-tuples in \( D' \).
2.3. Preliminaries

In output privacy, the focus is on protecting the data mining outcome from being disclosed. For decision trees, the paths of the tree are to be protected. Most studies adopting the random perturbation paradigm only focus on input privacy. The mining outcome (e.g., association rules, decision trees) are not encoded. Since the hacker only has access to the perturbed data, one may argue that in a twisted sense, the exact identity of the pattern is protected inasmuch as perturbation changes mining outcome. But then the custodian suffers the same fate: she cannot fully recover the exact pattern! In contrast, for (anti-)monotone transformations, the true classifier is revealed to the data custodian (given the no-outcome-change guarantee to be shown in Section 2.4), whereas the hacker has to guess the true identities of the paths of the decision tree, out of many possibilities.

Definition 2.3. Let a path in the decision tree $T'$ be of the form: $\land_{i=1}^{h} A_i \theta_i v_i'$. A path crack function $\vec{g} : (\delta'(A_1) \times \ldots \times \delta'(A_h)) \to (\delta(A_1) \times \ldots \times \delta(A_h))$ represents the guess the hacker makes on a transformed path. A guess is a \textbf{(path) crack} if $\land_{i=1}^{h} |\vec{g}_i(v_i') - f_{A_i}^{-1}(v_i')| \leq \rho_i$. The \textbf{pattern disclosure risk} is the fraction of the number of cracks to the number of paths in $T'$.

2.3.3 Attack Models Possibly with Prior Knowledge

Sources of prior knowledge may be from published statistics (e.g., the minimum age being 17, the median salary being 35K), from samples of similar data (e.g., a rival company having data similar to $D$), or the top $k$ modal classes (e.g., the mode of employee age being 34). Alternatively, the hacker may know that a given attribute follows a certain distribution (e.g., Zipf, Gaussian), even though the parameters may not be known exactly. While it is impossible to enumerate the many forms of prior knowledge, we abstract all these situations in the form of a single notion of knowledge points.

Definition 2.4. Let $v'$ be a value of attribute $A$ in $D'$. Let the hacker guess that $v' \in \delta'(A)$ corresponds to $v \in \delta(A)$. We say that $(v, v')$ is a \textbf{knowledge point} if $|v - f_{A}^{-1}(v')| \leq \rho$.

As encapsulations of various forms of prior knowledge, knowledge points are parameterized by how close they are to their true values. Following Definition 2.1, the radius $\rho$ defined above is the same as the radius used for a crack. Moreover, the hacker can use these knowledge points to form the basis of a curve fitting attack.

Definition 2.5 (Curve Fitting Attack). Let $(x_1, y_1), \ldots, (x_m, y_m)$ be $m$ knowledge points. Apply a curve fitting method to fit the $m$ points into a crack function $g$. 
2.4. The No-Outcome-Change Guarantee

We consider three curve fitting methods: (i) a linear regression line that minimizes the residuals of the \( m \) knowledge points; (ii) a polyline that connects the \( m \) knowledge points; and (iii) a spline that fits the \( m \) points. Clearly, curve fitting can also be adversely affected by wrong knowledge points, i.e., points that the hacker thought were accurate but that turn out to be way off. In our empirical evaluation, we call a knowledge point \((v, v')\) to be good if it satisfies Definition 2.4; but call it bad if \(|v - f^{-1}_A(v')| > 5\rho\). In Section 2.6, we will examine the impact of the knowledge points and various types of curve fitting attacks.

Another attack model is sorting. Consider the age attribute again. Even though the defined range of age is within \([0, 100]\), for a practical training data set with say 10,000 tuples on employees, it is almost guaranteed that every age in the interval \([20, 65]\) occurs in \(D\) at least once. Then the hacker takes the transformed values of age and sorts them in increasing order. Knowing the nature of a domain such as age, the hacker then maps the transformed values in increasing order to consecutive values starting with the (guessed) minimum value all the way to the (guessed) maximum. In the rest of this chapter, we show how to safeguard against these attack models, and simultaneously provide the no-outcome-change guarantee.

2.4 The No-Outcome-Change Guarantee

In this section, we show that the decision tree constructed using \(D\) is identical to the tree constructed using \(D'\) when either the gini index or entropy is used to select the split-points. These two selection criteria are the most widely used [8].

**Definition 2.6.** Let \(\langle t_1, \ldots, t_n\rangle\) be the ordered sequence of \(A\)-projected tuples in \(D\) such that \(t_i.A \leq t_j.A\) whenever \(i < j\). Equal values are in some canonical order. The class string \(\sigma_{A,D}\) for attribute \(A\) in data set \(D\) is defined as the string obtained by concatenating the class labels in the ordered sequence of tuples. Whenever the data set \(D\) is obvious, we denote the class string simply as \(\sigma_A\).

For the data set \(D\) in Figure 2.1(a), consider \(A\) being the age attribute. Sorting the tuples on age gives the class string \(\sigma_{age} = HHHLHL\), where \(H\) and \(L\) denote the class label high and low respectively. For the attribute salary, the class string \(\sigma_{salary} = HHHLHL\). Let us compare the corresponding class strings in \(D'\) shown in Figure 2.1(b). Observe that the class string for each attribute is unchanged.
2.4. The No-Outcome-Change Guarantee

**Proposition 2.1.** Let a monotone function be applied to transform attribute $A$ from $D$ to $D'$. Then the class string is preserved, i.e., $\sigma_{A,D'} = \sigma_{A,D}$. Similarly, for an anti-monotone transformation, $\sigma_{A,D'} = (\sigma_{A,D})^R$, where $\sigma^R$ denotes the reverse of string $\sigma$.

**Definition 2.7.** Given a class string $\sigma$ for attribute $A$ of a data set, we decompose $\sigma$ into multiple non-overlapping substrings $r_1, \ldots, r_m$ such that:

(i) $\sigma = r_1 \circ \ldots \circ r_m$;
(ii) for all $1 \leq i \leq m$, the substring $r_i$ consists of a single class label; and
(iii) for all $1 \leq i \leq (m-1)$, the substrings $r_i$ and $r_{i+1}$ are of different class labels. Each piece $r_i$ is a label run.

For the class string $\sigma_{\text{age}} = HHHLHL$ in Figure 2.1, there are four label runs $HHH, L, H$ and $L$. From Proposition 2.1, monotone functions preserve all the label runs of each attribute. Based on the definitions of the gini index and entropy, we have the following proposition.

**Proposition 2.2.** For each attribute, the split-point that optimizes either the gini index or entropy must not occur within a label run; it must be at the end points between two successive label runs.

Consider the first label run $HHH$ in age in Figure 2.1, corresponding to the age values 17, 20 and 23 in $D$. The above proposition says that the best split point for age cannot occur at age = 20. The candidate locations of the best split point for age must correspond to the end points of successive label runs – in this case 23, 32 and 43. Note that the same proposition applies to the transformed data $D'$. From Figure 2.1(b), the candidate locations of the best split point for age' are again the end points of successive label runs – in this case 31, 39, and 49. As formalized below, the candidate locations of the best split point are identical in $D$ and $D'$.

**Theorem 2.1.** The winning attribute and the split point location in $D'$ – relative to the sequential ordering of the label runs – are identical to that in $D$.

Proof Sketch: The theorem follows from the fact that in $D'$, the label runs are preserved as in $D$. Consequently, all the candidate split point locations are preserved as well. Finally, even though the data values change from $D$ to $D'$, the gini index and entropy calculations remain unchanged as they are based on relative frequencies. Hence, the winning attribute and the split point locations are preserved.

Note the difference between split points and split point locations. E.g., consider $\sigma_{\text{age}} = HHHLHL$ in Figure 2.1. The split point in Figure 2.1(a) is
age = (23+32)/2 and is located between the first and second label runs. In $D'$, as shown in Figure 2.1(b), the actual value of the split point is changed to $age' = (31+39)/2 = 35$, providing output privacy. However, the split point is still at the same location with respect to the sequential ordering of the label runs.

Let $T'$ be the decision tree obtained by mining the transformed database $D'$. Construct another decision tree $S$ as follows. Start from the root of $T'$ going top-down: for every node $x'$ in $T'$ of the form $A\theta v'$ (where $\theta$ is a comparison operator and $v'$ is a value in $\delta'(A)$), create a corresponding node $x$ in $S$ of the form $A\theta f^{-1}_A(v')$, where $f_A : \delta(A) \rightarrow \delta'(A)$ is the data transformation used for $A$. We have the following result from Theorem 2.1

**Corollary 2.1.** Let $D, D', T', S$ be as above. Let $T$ be the decision tree obtained by mining $D$ directly. Then: $S = T$.

### 2.5 A Piecewise Framework

To defend against the various attack models, the central idea of the proposed solution framework is to introduce breakpoints to break up the domain into multiple pieces, each of which is encoded by a different transformation function. Figure 2.2 gives a skeleton algorithm for implementing the piecewise framework. It consists of three main phases: (i) choosing breakpoints, (ii) choosing a transformation for each piece, and (iii) setting them up to satisfy a global constraint. We elaborate on all these aspects below.

**Algorithm PieceTransform**

**Input:** a sorted sequence of $A$– projected tuples

1. Call Procedure ChooseBP or ChooseMaxMP to decompose $A$ into $w$ pieces $r_1, \ldots, r_w$
2. For each piece $r_i$, call ChooseFunction to choose $f_i$
3. For $1 \leq i \leq w$, initialize $f_i$ such that they satisfy the global-monotone invariant, and apply $f_i$ to $\delta_i(A)$

Figure 2.2: A Piecewise Framework

#### 2.5.1 ChooseBP: Choosing Breakpoint Locations Randomly

It is the primary objective that when breakpoints are added, the no-outcome-change guarantee is preserved. To do so, let us return to the concept of label
2.5. A Piecewise Framework

runs in $D$: $HHHHHL LLLH HHHH$

values in $D$: 1 2 15 15 27 28 29 29 42 43 44

values in $D'$: 20 17 16 16 2 4 5 5 5 5 12 13 14 16 17 20

sorted values in $D'$: 2 4 5 5 5 12 13 14 16 17 20

runs in $D'$: L L L L H H H H H H H

Figure 2.3: Failing the Global-Monotone Invariant

runs introduced in Definition 2.7. Figure 2.3 shows a simple example when there are three label runs: $r_1 \equiv HHHH, r_2 \equiv LLLL, r_3 \equiv HHHHH$. Suppose a breakpoint is introduced exactly between two successive label runs to create three pieces, each of which uses its own (anti-)monotone transformation (i.e., third row of Figure 2.3). In the transformed data, however, the label runs are different (i.e., fifth row), and create a new class string, thus violating Theorem 2.1.

The key problem for the situation shown in Figure 2.3 is that the three pieces, after their individual transformations, no longer follow the original ordering among themselves. Specifically, as shown in the first row of Figure 2.3, the three label runs are in the ordering of $r_1 \equiv HHHH$, followed by $r_2 \equiv LLLL$, then by $r_3 \equiv HHHHH$. However, after the transformations as shown in row 4, the runs $r_2$ and $r_3$ have moved ahead of $r_1$, thereby causing the class string to change. The solution to this problem is to ensure that the pieces satisfy a global constraint to preserve their relative ordering after transformation.

**Definition 2.8.** Let the original domain be broken up into $w$ pieces $\delta_1(A), \ldots, \delta_w(A)$ with $w$ transformation functions $f_1, f_2, \ldots, f_w$. This set of transformations is said to satisfy the global-monotone invariant iff for all $1 \leq i < j \leq w, \forall v \in \delta_i(A), \forall u \in \delta_j(A)$, it is necessary that $f_i(v) < f_j(u)$. Similarly, the set is said to satisfy the global-anti-monotone invariant if the latter inequality is changed to $f_i(v) > f_j(u)$.

Figure 2.4 shows how the global-monotone invariant is satisfied. Because the largest transformed value of $r_1$ is 20, the invariant requires that all the transformed values in $r_2$ be strictly greater than 20, which is the case in Figure 2.4. Notice that this invariant is satisfied even though an anti-monotone function has been applied to $r_1$, whereas a monotone function has been applied to $r_2$. Similarly, all the transformed values in $r_3$ are greater
2.5. A Piecewise Framework

![Table with values]

Figure 2.4: Satisfying the Global-Monotone Invariant

than 25, irrespective of whether a monotone or an anti-monotone function is applied to \( r_3 \). In this way, as shown in row 5 of Figure 2.4, the label runs in \( D' \) are identical to those in \( D \).

While both Figure 2.3 and 2.4 consider putting breakpoints right at the boundaries between label runs, this is only a simplification for illustration purposes. In fact, any value can be a breakpoint location, as long as the global-monotone invariant is obeyed. This is essentially the nature of Procedure ChooseBP shown in Figure 2.5. For an attribute \( A \), ChooseBP decomposes the domain of \( A \) into \( w \) pieces for some \( w > 0 \), i.e., \( \delta(A) = \delta_1(A) \cup \ldots \cup \delta_w(A) \) and \( \delta_i(A) \cap \delta_j(A) = \emptyset, i \neq j \). The \( w \) breakpoints are randomly selected from the set of distinct values of \( A \). Even though ChooseBP is simple, its privacy protection power arises from the fact that the number \( w \) and the exact \( w \) locations are not known to the hacker. Specifically, if the cardinality of \( CBP \) is \( N \), then there are \( O(2^N) \) combinations for the hacker to ponder over. Section 2.6 will assess the effectiveness of ChooseBP empirically.

**Procedure ChooseBP**

**Input:** \( D_{A,C} \), the number \( w \) of breakpoints to be returned

1. Set \( CBP \) to be \( \{ t.A | t.A \in D_{A,C} \} \), the set of \( A \)-values
2. Randomly pick \( w \) values from \( CBP \) as the breakpoints
3. Return \( BP /* the set of selected breakpoints */ \)

Figure 2.5: A Skeleton for ChooseBP

2.5.2 ChooseMaxMP: Exploiting Monochromatic Pieces

**Definition 2.9.** A value \( v \in \delta(A) \) in \( D \) is monochromatic if all the tuples with \( v \) as the \( A \) attribute value agree on the label, i.e., \( \forall t_1, t_2 \in D \) such that
2.5. A Piecewise Framework

\( t_1.A = t_2.A = v \) and \( t_1.C \neq t_2.C \), where \( C \) is the class label. Furthermore, if all the tuples in a piece \( r \) contain monochromatic values and the same label, then \( r \) is called a monochromatic piece.

In Figure 2.4, 29 is a non-monochromatic value; every other value, including 15, is monochromatic. The piece \( r_1 \) is a monochromatic piece. The key benefit offered by a monochromatic piece is that there is no requirement to apply either a monotone or an anti-monotone function. In Figure 2.4, the values in \( r_1 \): 1, 2, 15 are transformed anti-monotonically to 20, 17 and 16 respectively. Note that the label runs do not change if the transformation \( f \) is any bijective function. For example, 1, 2 and 15 can be transformed so that \( f(1) < f(15) < f(2) \) (e.g., see row 3 of Figure 2.7). When such a bijective function can be used, a sorting attack is blocked. Furthermore, the space of bijective functions strictly contains the space of monotone or anti-monotone functions. Thus, while ChooseBP uses randomized breakpoints to confuse the hacker, the use of a bijective function here creates an even more serious combinatorial problem for the hacker. Specifically, if \( N \) is the total number of values in all the monochromatic pieces, there are \( O(N!) \) combinations for the hacker to ponder over.

To maximize the benefit of monochromatic pieces, Procedure ChooseMaxMP, shown in Figure 2.6, finds all the monochromatic values and grows them into monochromatic pieces of the largest size. Given a sorted sequence of \( A \)-values, this task can be achieved by a simple scan from the smallest to the largest value.

To illustrate the procedure, consider Figure 2.7, which has the same original label runs as in Figures 2.3 and 2.4. Let us begin from the smallest value 1. Because the value 1 is monochromatic, 1 is added to \( BP \) in line (11). The subsequent monochromatic values 2 and 15 are skipped according to line (16) effectively growing \( r_1 \). The next value is 27. While it is monochromatic, the label is different. Thus, a new monochromatic piece begins in line (13) and 27 is added to \( BP \). The next value 28 remains in the same monochromatic piece containing 27. The next value 29 is non-monochromatic. This marks the end of the previous monochromatic piece and starts a non-monochromatic piece in line (4); 29 is added to \( BP \). To complete the example, the next and last breakpoint is 42. In sum, ChooseMaxMP creates 4 pieces: \( r_1 \equiv (1, H)(2, H)(15, H)(15, H) \), \( r_2 \equiv (27, L)(28, L) \), \( r_3 \equiv (29, L)(29, L)(29, H)(29, H) \) and \( r_4 \equiv (42, H)(43, H)(44, H) \).

Because \( r_1, r_2 \) and \( r_3 \) are monochromatic, any bijective function can be used. Notice that the way that breakpoints are selected by ChooseMaxMP maximizes the number of values for which a bijective function can be applied.
2.5. A Piecewise Framework

Procedure ChooseMaxMP

Input: $D_{A,C}$ sorted on attribute $A$, the desired number $w$ of breakpoints to be returned

1. Set $BP$ to be empty, flag $MP$ to $false$, $curLabel$ to null
2. For each value $v$ starting from the smallest {
3. If $v$ is not monochromatic {
4. If $MP$ is true { /* end of a monochromatic piece */
5. Add $v$ to $BP$ /* a new non-monochromatic piece */
6. Set $MP$ to $false$, and $curLabel$ to null
7. } /* else simply skip to the next value */
8. }
9. else { /* $v$ is monochromatic */
10. If $MP$ is $false$ { /* end of a non-mono. piece */
11. Add $v$ to $BP$ /* a new monochromatic piece */
12. Set $MP$ to $true$ and $curLabel$ to $v.C$.
13. } else if $curLabel \neq v.C$ { /* different label */
14. Add $v$ to $BP$ /* a different monochromatic piece */
15. Set $curLabel$ to $v.C$
16. } /* else continues the current monochromatic piece */
17. } } /* end for-loop */
18. If the size of $BP$ is $h$ and is less than $w$,
19. Randomly pick $(w - h)$ breakpoints, if any, from the set of
20. non-monochromatic values as in ChooseBP
21. Return $BP$

Figure 2.6: A Skeleton for ChooseMaxMP

<table>
<thead>
<tr>
<th>runs in $D$:</th>
<th>$H$</th>
<th>$H$</th>
<th>$H$</th>
<th>$H$</th>
<th>$L$</th>
<th>$L$</th>
<th>$L$</th>
<th>$H$</th>
<th>$H$</th>
<th>$H$</th>
<th>$H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>values in $D$:</td>
<td>1</td>
<td>2</td>
<td>15</td>
<td>15</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>42</td>
<td>43</td>
</tr>
<tr>
<td>values in $D'$:</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>19</td>
<td>18</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>35</td>
<td>31</td>
</tr>
<tr>
<td>pieces obtained:</td>
<td>$r_1$</td>
<td>$r_2$</td>
<td>$r_3$</td>
<td>$r_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sorted values in $D'$:</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>18</td>
<td>19</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>31</td>
<td>33</td>
</tr>
<tr>
<td>runs in $D'$:</td>
<td>$H$</td>
<td>$H$</td>
<td>$H$</td>
<td>$H$</td>
<td>$L$</td>
<td>$L$</td>
<td>$L$</td>
<td>$H$</td>
<td>$H$</td>
<td>$H$</td>
<td>$H$</td>
</tr>
</tbody>
</table>

Figure 2.7: Maximizing Monochromatic Pieces
Row 5 in Figure 2.7 shows the transformed values. Notice that the global-monotone invariant is maintained between successive pieces.

Notice that there is no uncertainty in the positions of the monochromatic pieces as their sizes are maximized by ChooseMaxMP. In other words, the hacker knows exactly what the monochromatic and non-monochromatic pieces are in $D'$. However, knowing the pieces in $D'$ does not help the hacker at all in cracking the corresponding values in $D$. For example, in Figure 2.7, knowing rows 5 and 6 do not help the hacker in cracking row 2.

In the simple example in Figure 2.7, three important cases have not been illustrated. First, if the number of monochromatic pieces is less than $w$, ChooseMaxMP still operates by essentially reverting back to ChooseBP in selecting the remaining breakpoints randomly from among the non-monochromatic values. Experimental results later will show that even in this rare case, the proposed framework offers adequate domain disclosure protection. And whenever there are monochromatic pieces, the framework exploits them to give enhanced protection.

Second, if there are not enough non-monochromatic values to choose from, ChooseMaxMP returns all the non-monochromatic values, together with the starting values of the monochromatic pieces. For real data, this is a very good situation for the data custodian as it implies that the total number of values in the monochromatic pieces dominates.

Finally, in our simple example, the shortest piece is of length 2. We include short pieces in our example for simplicity. In practice, ChooseMaxMP may impose a minimum width threshold (e.g., width $\geq 5$).

### 2.5.3 Selecting Transformations Randomly

After breakpoints are selected by using ChooseBP or ChooseMaxMP, the next step is to choose a transformation for each piece from a family of functions. Figure 2.8 shows the procedure of how to randomly select a function for a piece $r$. Let $F_{bi}$ denote a set of bijective functions including non-monotone ones, e.g., any permutation functions. This set $F_{bi}$ is applicable to monochromatic pieces (step 3 and step 4). For non-monochromatic pieces, we are restricted to the family of (anti-)monotone functions, denoted by $F_{mono}$. This set can contain polynomials of degree $\geq k$, logarithmic functions, parabolic functions and so on. Note that $F_{mono}$ is closed under composition. That is to say, for any monotone functions $f, g \in F_{mono}$, their composition $f \circ g$ is also monotone and is in $F_{mono}$. For example, given $f(x) = x^{1/2}$ and $g(x) = \log(x)$, $f \circ g(x) = (\log x)^{1/2}$ is monotone. Thus, again, the size of $F_{mono}$ can be arbitrarily large. As in the case for
2.5. A Piecewise Framework

A monochromatic piece, a randomization step is used to select the transformation for the non-monochromatic piece. We choose a function for \( r \) in the following way. If the transformations for the whole domain is global-monotone, then we randomly choose a monotone function for \( r \); otherwise, we randomly choose an anti-monotone function for \( r \) (step 5 and step 6). ChooseFunction guarantees the global monotone/anti-monotone constraint is satisfied.

**Procedure ChooseFunction**

**Input:** \( r \): a sorted sequence of values from \( A \) with class labels

1. If \( r \) is monochromatic, choose \( f \) randomly from \( F_{bi} \).
2. Else /* a piece with multiple labels */ {
3. If the set of transformations is global-monotone
4. Select a monotone function \( f \) randomly from \( F_{mono} \)
5. Else if the set of transformations is global-anti-monotone
6. Select an anti-monotone function \( f \) randomly from \( F_{mono} \)
}
7. Return \( f \) /* a function to transform the values in \( r \) */

Figure 2.8: A Skeleton for Randomly Choosing Transformations Piecewise

Agrawal et al. propose an order preserving encryption approach in [3]. They map the original values into another domain while preserving the order of the values. Since the orders of the mapped values are the same to the orders of the original values, the label runs are preserved. Therefore, the order preserving encryption approach also preserves the decision tree. However, our piecewise transformation framework has two advantages over the order preserving encryption regarding the privacy protection. First, monochromatic pieces could provide extra protection on privacy because we could use any function in \( F_{bi} \) to transform these pieces. Second, even when a domain does not contain any monochromatic piece, the piecewise framework can still provide more protection. We could regard the order preserving encryption as a kind of monotone function and include it in \( F_{mono} \). Unlike the order preserving encryption approach, by selecting different transformations, which could include a mix of the order preserving encryption and other functions in \( F_{mono} \), for different pieces, we could provide enhanced protection on privacy.
2.5.4 Defense Against Sorting Attacks

Next we turn to sorting attacks. As specified in Procedure ChooseFunction, if a piece $r$ is monochromatic, $f_r$ is chosen from $\mathcal{F}_{bi}$, and a sorting attack is blocked. What about a non-monochromatic piece? Note that a non-monochromatic piece $r$ with a monotone transformation can already have its own inherent protection against a sorting attack – provided that there are enough “discontinuities” within $r$. A value $v \in \delta_r(A)$ is a discontinuity if there is no tuple $t$ in $r$ with $t.A = v$, where $\delta_r(A)$ denotes the dynamic range $[\min_A, \max_A]$ of $A$ in $r$, and $\min_A, \max_A$ denote the least and greatest $A$-value occurring in $r$. To be more precise, if $\delta_r(A)$ contains some discontinuity values, then the attacker can only crack a value $v' \in \delta'_r(A)$ to within a range, say $R_g = [v_1, v_2] \subseteq \delta_r(A)$. Recall from Definition 2.1 that a guess is a crack if $|g(v') - f^{-1}(v')| \leq \rho$. Let $R_\rho = [v - \rho, v + \rho]$, where $v = f^{-1}(v')$. Then the probability that a value $v'$ is cracked under sorting attack can be redefined as: $P(|g(v') - f^{-1}(v')| \leq \rho) = \frac{|R_g \cap R_\rho|}{|R_g|}$.

Let us consider the entire sorted sequence in $D'$ in Figure 2.7 (i.e., row 5). Consider $v' = 27$. There are 5 values ranked ahead of 27 and 3 values ranked after 27. Thus, given that the original domain is $[1, 44]$, $g(v')$ can range from $R_g = [6, 41]$. As shown, the original value is $f^{-1}(v') = 29$. Let us say that the width of a crack is 2, giving $R_\rho = [27, 31]$. Thus, the probability that $v'$ is cracked is $5/36$. From the above formula, it is clear that the larger the number of discontinuities, the wider the range $R_g$ and the smaller the crack probability.

In sum, the data custodian can follow the “recipe” below to determine if an attribute $A$ is safe for disclosure. If $A$ has many monochromatic pieces, or if the non-monochromatic pieces contain many discontinuities, then $A$ is safe with low crack percentage against a sorting attack. The only situation that is unsafe is when $A$ has few monochromatic values and simultaneously few discontinuities. However, the data custodian must decide whether domain disclosure risk for $A$ is a primary concern or not. As discussed before, perhaps the more important issue for the custodian is the association of the $A$-values with the values of other attributes, not just the $A$-values themselves. Empirical results in Section 2.6 will examine this perspective.

A final point concerns the amount of information the data custodian needs to keep in order to decode the decision tree $T'$ to get $T$. While this is discussed in [4], it suffices to say that the information required is rather minimal (i.e., the locations of breakpoints and the transformations used).
2.6 Empirical Evaluation

2.6.1 Experimental Setup

**Data Sets:** We conducted our experimental evaluation on many benchmark data sets, including the forest covertype, census income and WDBC data sets from the UC Irvine collection. The results reported below are based on the forest covertype data set. These results are representative of those that we do not have space to show.

<table>
<thead>
<tr>
<th>attr.</th>
<th>dynamic range width</th>
<th># of distinct values</th>
<th># of mono. piece</th>
<th>avg Length of mono. piece</th>
<th>total % of mono. values</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>2000</td>
<td>1978</td>
<td>9</td>
<td>163</td>
<td>74.2%</td>
</tr>
<tr>
<td>#2</td>
<td>361</td>
<td>361</td>
<td>0</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>#3</td>
<td>67</td>
<td>67</td>
<td>1</td>
<td>15</td>
<td>22.4%</td>
</tr>
<tr>
<td>#4</td>
<td>1398</td>
<td>551</td>
<td>22</td>
<td>10</td>
<td>40.0%</td>
</tr>
<tr>
<td>#5</td>
<td>775</td>
<td>700</td>
<td>14</td>
<td>24</td>
<td>48.0%</td>
</tr>
<tr>
<td>#6</td>
<td>7118</td>
<td>5785</td>
<td>202</td>
<td>18</td>
<td>62.9%</td>
</tr>
<tr>
<td>#7</td>
<td>255</td>
<td>207</td>
<td>2</td>
<td>41</td>
<td>39.6%</td>
</tr>
<tr>
<td>#8</td>
<td>255</td>
<td>185</td>
<td>8</td>
<td>6</td>
<td>25.9%</td>
</tr>
<tr>
<td>#9</td>
<td>255</td>
<td>255</td>
<td>3</td>
<td>8</td>
<td>9.4%</td>
</tr>
<tr>
<td>#10</td>
<td>7174</td>
<td>5827</td>
<td>229</td>
<td>17</td>
<td>66.8%</td>
</tr>
</tbody>
</table>

Table 2.1: Statistics of Attributes

The forest covertype data set consists of 581,012 tuples and 10 numeric attributes (and other non-numeric attributes). The table in Table 2.1 gives the key statistics of the 10 attributes. There are attributes with many discontinuities; the number of discontinuities is the difference between the second column and the third column of the table. But attributes 2, 3 and 9 have no discontinuity. There are also attributes with several monochromatic pieces. The last column gives the percentage of values that are contained in the monochromatic pieces. For attribute 1, the percentage is the highest at 74%. But for attribute 2, the percentage is 0% as there is no monochromatic piece. This attribute represents the worst case, as it also does not contain any discontinuity.

**Transformations:** $\mathcal{F}_\text{bi}$ consists of transformations for monochromatic pieces. We used a random permutation function. $\mathcal{F}_\text{mono}$ is for non-monochromatic pieces. We used linear and higher order polynomials, log, and log$^2$ (denoted $\text{sqrt}(\log)$).
2.6. Empirical Evaluation

**Prior Knowledge:** Curve-fitting attacks – linear regression, spline, polyline – require some number of knowledge points (denoted as KP below) to work with. We simulated the **ignorant hacker** who has no prior knowledge. The width \( \rho \) for a KP varied from 1%, 2% and 5% of the width of the dynamic range. Given the definition of \( \rho \)'s, we feel that each good KP represents a considerable amount of prior knowledge. Thus, a hacker with 2 or 4 good KPs is referred to as a **knowledgeable** and an **expert** hacker respectively. The locations of KPs are selected randomly. Furthermore, given the randomization involved, each disclosure risk figure reported below is based on the median of 500 random trials. The experiments were implemented in a MATLAB environment. We ran the experiments on an Intel Pentium PC with 3GHz CPU and 2GB RAM.

2.6.2 Domain Disclosure Risk

**Impact of Breakpoints, Monochromatic Pieces and KPs**

Figure 2.9 shows the domain disclosure risks (i.e., the y-axis) for all 10 attributes of the forest covertype data set using polyline as the curve fitting model. For each attribute, the first bar gives the baseline when no breakpoint is used but with an expert hacker (i.e., 4 KPs). The second and third bar correspond to the cases when ChooseBP and ChooseMaxMP are used, with an expert hacker. Note that to make the comparisons between ChooseBP and ChooseMaxMP fair, ChooseBP uses the same number of breakpoints as ChooseMaxMP, which is determined by the number of monochromatic pieces for the attribute. (The minimum number of breakpoints in each case is set to \( w = 20 \).) The last bar corresponds to the case when ChooseMaxMP is used with a knowledgeable hacker.

The difference between the first two bars indicates the reduction of crack percentage with breakpoints. For instance, for attribute 1, the crack percentage drops from over 65% to 30%. Reduction is achieved for every attribute. Even for the “worst case” attribute 2 as shown in Table 2.1, breakpoints manage to keep the crack percentage below 25%. This shows that the use of breakpoints alone is effective, *even when there is no monochromatic piece.*

The difference between the second and third bar in Figure 2.9 is due entirely to the monochromatic pieces. Depending on the percentage of monochromatic values, the reduction can be very significant. For the first attribute, the crack percentage drops from 30% to below 10%. This shows the effectiveness of ChooseMaxMP in exploiting monochromatic pieces, if present.
2.6. Empirical Evaluation

As a reference point, when random perturbations are used, there is a chance that a data value is not changed and the true value is revealed. For example, in [9], many situations examined leave a significant percentage (e.g., 30%) of values unchanged, even with no prior knowledge used.

Figure 2.9: Impact of Breakpoints and Monochromatic Pieces on Domain Disclosure

The third bar of each attribute in Figure 2.9 corresponds to an expert hacker (i.e., 4 KPs). For all 10 attributes, the crack percentage is significantly lower when the hacker has less domain knowledge. The fourth bar of each attribute shows the crack percentage when the hacker is only a knowledgeable hacker (i.e., 2 KPs). In all cases, the crack percentage falls below 15%. And for an ignorant hacker, the crack percentage falls below 5%. Finally, the crack percentage is sensitive to the presence of even a single bad KP; for lack of space, details are omitted.

A combination of Attacks

So far all the results presented here are based on the polyline curve fitting model and the sqrt(log) transformation function. The table below shows the corresponding values for alternative situations. All the figures are based on attribute 10 of the forest covertype data set with ChooseMaxMP and an expert hacker. It is clear that results shown above represents a “worst case” analysis.

A natural extension to the various attacks analyzed so far is to consider when the hacker applies all of these attack models. The central question here is whether the different attack models crack similar or different items. Figure 2.10 shows a Venn diagram of the cracks for linear regression, spline
2.6. Empirical Evaluation

<table>
<thead>
<tr>
<th></th>
<th>polynomial</th>
<th>log</th>
<th>sqrt(log)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>10.39%</td>
<td>11.53%</td>
<td>10.85%</td>
</tr>
<tr>
<td>Spline attack</td>
<td>14.51%</td>
<td>14.8%</td>
<td>15.28%</td>
</tr>
<tr>
<td>Polyline attack</td>
<td>15.55%</td>
<td>18.05%</td>
<td>18.03%</td>
</tr>
</tbody>
</table>

fitting and polyline fitting on attribute 10 using sqrt(log) as the transformation. For instance, 3% of the domains are cracked solely by polyline, whereas an additional 9% are cracked by polyline and spline but not by linear regression.

![Venn Diagram of Cracks: the Combination Attack](image)

Figure 2.10: The Venn Diagram of Cracks: the Combination Attack

The question to ask here is: what the disclosure risk for this combination attack is. One simple answer is to add up all the percentages, which gives rise to 25%. However, for most situations, this is an over-estimation of risk. To illustrate, suppose that a specific item $a$ is identified to be $a$ by polyline, $b$ by spline and $c$ by linear regression. While it is true that one of the three attacks correctly reveals the identity of item $a$, the hacker does not know which one. Thus, he cannot distinguish whether $a$ is really $a$, $b$ or $c$. One way to calculate the percentage of cracks is to use expected values. Assuming the hacker trusts the three crack models equally, this gives an expected crack percentage of 12.5%. Another way is to count only those items as cracks if two or more methods agree on them. From the Venn diagram shown in Figure 2.10, this gives a total combined crack percentage of 16%.

---

3This is similar in effectiveness as the 30% of unchanged values left unprotected by random perturbation [9]. In our case, however, the crack percentage includes the hacker’s prior knowledge. If the hacker has no prior knowledge, the crack percentage is significantly lower.
2.6. Empirical Evaluation

Table 2.2: Sorting Attack: Worst Case

<table>
<thead>
<tr>
<th>attr.</th>
<th># of discontinuities</th>
<th>total % of mono. values</th>
<th>worst case crack % by sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>22</td>
<td>74.2%</td>
<td>26%</td>
</tr>
<tr>
<td>#2</td>
<td>0</td>
<td>0.0%</td>
<td>100%</td>
</tr>
<tr>
<td>#3</td>
<td>0</td>
<td>22.4%</td>
<td>78%</td>
</tr>
<tr>
<td>#4</td>
<td>847</td>
<td>40.0%</td>
<td>4%</td>
</tr>
<tr>
<td>#5</td>
<td>75</td>
<td>48.0%</td>
<td>22%</td>
</tr>
<tr>
<td>#6</td>
<td>1333</td>
<td>62.9%</td>
<td>8%</td>
</tr>
<tr>
<td>#7</td>
<td>48</td>
<td>39.6%</td>
<td>13%</td>
</tr>
<tr>
<td>#8</td>
<td>70</td>
<td>25.9%</td>
<td>11%</td>
</tr>
<tr>
<td>#9</td>
<td>0</td>
<td>9.4%</td>
<td>90%</td>
</tr>
<tr>
<td>#10</td>
<td>1347</td>
<td>66.8%</td>
<td>7%</td>
</tr>
</tbody>
</table>

Sorting Attack: the Worst Case Analysis

Finally, we measure the risk of a sorting attack on all 10 attributes. The table in Table 2.2 gives the worst case results because we assume that the hacker has the knowledge of the true minimum and maximum values of the dynamic range. If these two pieces of prior knowledge is not known, the crack percentage is significantly lower. The second column of the table in Table 2.2 gives the number of discontinuities within the dynamic range (cf: Table 2.1). The third column is directly copied from the last column of Table 2.1. As expected, the three attributes 2, 3 and 9 are the most vulnerable with no discontinuity, and a small percentage of monochromatic values (hardly surprising given that these attribute domains are small and the data set contains over 500,000 rows). However, as long as an attribute has a fair number of discontinuities or monochromatic values, the attribute is safe against a sorting attack even in the worst case.

2.6.3 Subspace Association Risk

Next we turn to subspace association disclosure risk. For the 10 attributes of the forest covertype data set, there are over 1,000 subspaces containing two or more attributes. Figure 2.11 shows the results of a few selected subspaces. All subspaces shown contain either 2 or 3 attributes. For the ones selected, they are divided into two categories.

The first category are attributes for which the curve fitting attack is
2.6. Empirical Evaluation

more serious than the sorting attack. Specifically, we consider the set of attributes \{4, 7, 10\} and their subspaces. The first three bars in Figure 2.11 show their individual domain disclosure risks from Figure 2.9 with an expert hacker. The next four bars show the subspace association risks for all the combinations. For example, the individual risks of attribute 4 and 7 are 16% and 25% respectively. The association disclosure risk of attributes 4 and 7 together, however, drops significantly to 4%. And the association disclosure risk among attributes 4, 7 and 10 drops to 0.2%. In general, as the subspace becomes larger, the disclosure risk drops significantly.

Figure 2.11: Subspace Association Disclosure Risk

The second category are attributes for which the sorting attack is more serious. For a worst case discussion, we return to attribute 2. Even though attribute 2 is 100% cracked by sorting in the worst case, note that the association disclosure risks of attribute 2 with other attributes can be acceptable (i.e., last three bars of Figure 2.11). Thus, for the data custodian to decide whether attribute 2 can be safely released, the data custodian needs to first determine which is the primary concern: attribute 2 alone versus the association of attribute 2 with other attributes. For many situations, it is the association disclosure that is more critical.

It is also interesting to compare the association disclosure risk with and without attribute 2 in Figure 2.11. While the domain disclosure risk of attribute 10 alone is 18%, the association disclosure risk of attributes 2 and 10 together is 15%. This is due to how values in attribute 2 are associated with values in attribute 10. This skew leads to the following observation: \( \text{risk}(A, B) < \text{risk}(A) \times \text{risk}(B) \). This is good news for the data custodian if the primary concern is subspace association risk, rather than domain risk.
2.6.4 Output Privacy: Pattern Disclosure Risk

Finally, to measure pattern disclosure risk, we applied the C4.5 decision tree algorithm on the 10 attributes of the forest covertype data set. The decision tree constructed contains 1707 paths from the root. The maximum length of these paths is 40. The following table shows the frequency of path lengths and the corresponding cracks.

<table>
<thead>
<tr>
<th>Length of Decision Paths</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>&gt;6</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Paths</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>15</td>
<td>24</td>
<td>1648</td>
</tr>
<tr>
<td># of Cracks</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.3: Output Privacy

Among all the 1707 paths from the root, there is only 1 path of length 2 that is cracked. In fact, this result is based on an insider hacker (i.e., 8 KPs) and a 5% width. For a less powerful hacker, or a smaller radius, all the paths are protected showing impeccable output privacy protection.

2.7 Future Work

In ongoing work, we study how to generalize the piecewise framework from decision trees to SVM and other kernel methods. The difference is that the dividing planes can have arbitrary orientations. In next chapter, we show how the no-outcome-change guarantee and the other two pillars of privacy can be supported for SVM using a set of transformations different from the ones analyzed here.
2.8 Bibliography


Chapter 3

Patterns and Privacy Preservation with Prior Knowledge for SVM Classification\(^4\)

3.1 Introduction

In outsourcing data mining, data owners provide their data to the service provider. To preserve privacy, there are at least two prevalent scenarios considered in the literature. In the data collector scenario, individual data owners send the data to a collector who will do the data mining task. The assumption is that the collector might not be trusted. Thus, the primary concern is to protect the privacy of the input data. The dominant approach is random perturbation, i.e., transforming data values by adding random noise in a principled way [3, 12, 19].

The second scenario is the data custodian model. A data custodian is a person or an organization who either owns the data or has the responsibility to protect the privacy information inherent in the data. For example, a medical research group or hospital (i.e., the custodian) may consider outsourcing data mining using patient data. They need to find a way to protect the patient’s private information.

As discussed in [7], there are three “pillars” to privacy preservation. The first pillar, which has received the most attention, is input privacy, which aims to protect the privacy of the input data to the data mining algorithm [25]. The second pillar is output privacy, which aims to protect the true identity of the mining outcome. For example, a disease prediction model learned from patient data should be protected and not be released to the

\(^4\)A version of this chapter will be submitted for publication. Bu, S., Greif, C., Lakshmanan, L. V.S., and Ng, R. T.: Patterns and Privacy Preservation with Prior Knowledge for SVM Classification
public to prevent malicious utilization. The third pillar is the *minimization of outcome change*. Existing studies (e.g., [3, 12, 19]) from the data collector scenario provide input privacy by random perturbation. However, the more the data is perturbed, the more the outcome has been changed (even though input privacy is enhanced).

For the data custodian scenario, several recent studies show situations when it is possible to provide the *No-Outcome-Change* (NOC) guarantee without compromising input and output privacy. In [31], Wong et al. propose an algorithm to give the NOC guarantee for outsourcing of association rule mining. In [7], we propose an algorithm to give the NOC guarantee for building a decision tree, while simultaneously providing input and output privacy. Table 3.1 highlights the differences between the data collector scenario and the data custodian scenario.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Studies</th>
<th>Input Privacy</th>
<th>Output Privacy</th>
<th>NOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Collector</td>
<td>[1, 2, 3, 12], [18, 19], etc.</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Custodian</td>
<td>[8, 9, 21, 31], [7], this thesis, etc.</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>K-Anonymity</td>
<td>[16, 22, 27], [13, 20, 30], etc.</td>
<td>√</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Three Pillars of Privacy Preservation

In this chapter, we address the problem of privacy protection in outsourcing of support vector machines (SVM). Classification is one of the most popular data mining operations, and SVM is one of the most widely used classification methods. Instead of executing the SVM classification by herself, a data custodian might want to outsource the SVM to outside service providers in the following two cases. Firstly, the data custodian might not have data mining expertise. Even though there is an increasing number of off-the-shelf mining software packages, they still call for substantial expertise on the part of the custodian to train the SVM classifier, including selecting a better kernel function, tuning parameters, interpreting the intermediate results during the training process, etc. Thus, a data custodian might want to outsource the SVM classification to a data mining service provider who has the required expertise. Secondly, instead of buying expensive computing facilities, a data custodian might want to submit the heavy data mining computation to the data mining service provider. This may become even more popular with the recent advent of cloud computing. The data custodian needs to protect the private information in both cases.
3.1. Introduction

Chen et al. propose using random rotation to provide the NOC guarantee for outsourcing SVM [8]. In a follow-up study, they consider how to deal with hackers/attacks with background or prior knowledge [9]. Sources of prior knowledge may include published statistics (e.g., the minimum age being 17, the median salary being $35K), from samples of similar data (e.g., a rival company targeting the same market), or from knowledge that a certain attribute follows a specific distribution (e.g., Zipf).

Since it is impossible to consider all possible attack models with prior knowledge in one thesis, we consider two expressive and “realistic” attack models: one based on curve-fitting with correlation, and the other one based on linear algebraic manipulation which we call a global attack. Both attack models are realistic in the sense that they only require a modest amount of prior knowledge; yet they can be very damaging. While these two models are not exhaustive, they are two broad types of attacks. The objective of this chapter, then, is to explore how to provide the NOC guarantee for the SVM classification and be robust against these attack models. Specifically, we make the following contributions:

- We first observe that the random rotation approach proposed by Chen et al. ([8, 9]) is vulnerable to the two attack models. To address this problem, we develop an approach, Principled Orthogonal Transformation (POT), which iteratively and monotonically reduces the correlation by generating orthogonal transformations in a principled way based on the Gram-Schmidt procedure [28]. The generated transformation provides the NOC guarantee for both linear and non-linear SVMs, while providing much stronger input and output privacy protection when compared with the random rotation approach.

- To guard against a "global" attack, which will be formally introduced in Section 3, we propose an algorithm that identifies data points called True Negative Points (TNP), which are guaranteed not to be support vectors. The algorithm then perturbs these true negative points to enhance privacy. It is shown that for a linearly separable data set, the NOC guarantee is preserved. However, for a linearly non-separable data set, as we shall show, the NOC guarantee may be violated but the change is minimized. For the latter case, we present empirical results, using several benchmark data sets, to show that the quality of the SVM classifier is not significantly affected.

Roadmap: The rest of this chapter is structured as follows. Section 3.2 describes the related work in privacy preservation data mining. Section 3.3
introduces the notion of knowledge points and the associated attack models. The POT approach to counter curve fitting attacks is proposed in Section 3.4 and the TNP approach to counter global attacks is studied in Section 3.5. Section 3.6 shows the experiments that validate the effectiveness of the proposed approaches. The way in which POT and the TNP perturbation are meant to be used by a data custodian to outsource her data is discussed at the end of this section. Finally, conclusion and future work are discussed in Section 3.7.

3.2 Related Work

Chen and Liu propose a rotation-based transformation approach for SVM [8]; the transformation gives the NOC guarantee. They propose a unified column privacy metric to select a rotation matrix, which could provide a higher privacy level, by swapping the rows of the rotation matrix. They use Independent Component Analysis (ICA) as the attack model to evaluate the protection level. However, ICA assumes that the hacker has no prior knowledge. In their extended work [9], they continue to study random rotation to transform the original data. They investigated attack models, such as distance-inference attacks, in which a hacker might have enough data points to crack the transformation matrix. Their approach amounts to adding noise to the transformed data; but this sacrifices classification accuracy.

Liu et al. study random orthogonal transformations from an attacker’s view for k-means and k-NN clustering [21]. However, like random rotation, random orthogonal transformation is vulnerable to attacks with prior knowledge.

In [7], we study how to provide the NOC guarantee for building decision tree classifiers. Decision trees are based on recursive partitioning, while SVMs are matrix-based. Thus, the study here has no overlap with the approach proposed in [7]. In [31], Wong et al. propose an algorithm to give the NOC guarantee for outsourcing of association rule mining. They share with us the goal of providing the NOC guarantee.

Random perturbation is a popular approach to transform the original data by adding noise [1, 2, 3, 12, 18, 19]. As discussed earlier, this approach is designed for the data collector scenario and mainly focus on input privacy (Table 3.1).

In [24], Oliveira and Zaiane study privacy preserving clustering problem and apply affine transformations on the data set. However, different attributes are subjected to different transformations. The geometric prop-
3.3 Knowledge Points and Attack Models

In this chapter, given a data set $D$ of $d$ attributes containing $n$ tuples, each tuple $x_i$ is represented as a point in $d$-dimensional space, i.e., $x_i = \{x_1, ..., x_d\}^T$, which is a column vector in $d$-dimensional space. Therefore $D$ can be represented as a $d \times n$ matrix. We can apply a transformation $\tau$ on $D$. The transformed data set is $D' = \tau(D)$. For a point $x$ in $D$, the corresponding transformed point is $x' = \tau(x)$.

3.3.1 Knowledge Points

There could be many sources of prior knowledge (e.g., published statistics, samples of similar data, etc.). We abstract all of them into a uniform notion of knowledge points.
3.3 Knowledge Points and Attack Models

Both Chen et al. [9] and Liu et al. [21] define prior knowledge as pairs of original points and the corresponding transformed points. However, in real life, a hacker might only know approximate values but not the exact original values. In this chapter, we propose a more realistic and practical notion of knowledge points. The key idea is that if the hacker’s guessed value for a given transformed value falls within a certain radius of the exact original value, we regard that pair as a knowledge point. Definition 3.1 formalizes this notion.

Definition 3.1. (Domain Knowledge Point) Given a transformed value $u'$ in attribute $i$ of $D'$, suppose a hacker believes that the corresponding original value of $u'$ is $w$ from her prior knowledge. We say that $(w, u')$ is a domain knowledge point if $|w - u| \leq \delta_i$, where $u$ is the true original value of $u'$ and $\delta_i$ is a given parameter for attribute $i$.

Here, $\delta_i$ measures how close the guessed original value $w$ is to the real value $u$ and is defined as a fraction of the domain range of attribute $i$. Thus, for a knowledge point as defined above, the guessed original value is within a radius of $\delta_i$ from the true original value.

A hacker could gain prior knowledge from various sources of published statistics, samples of similar data, or knowledge that a certain attribute follows a specific distribution. For example, if a hacker has a similar data set from a rival company and knows that for the Age attribute, the maximum is 60 and the minimum is 18. When she gets the transformed data set, and finds the maximum of the transformed value to be 125 and the minimum to be 6, she can simply make a guess that 125 is transformed from 60 and 6 is from 18. Even if the true original value for 125 is 58 and the true original value for 6 is 19, the guessed values are very close. Meanwhile, if a hacker knows the mean, median, or other statistics of the original domain, she could simply map those values to the corresponding transformed values to form more knowledge points. Section 3.6.2 shows that even this kind of simple statistic value mapping between original domain and transformed domain could breach considerable privacy if the original values and transformed values are highly correlated.

3.3.2 Attack Models with Prior Knowledge

In Definition 3.1, $\delta_i$ measures how close the guessed original value $w$ is to the real value $u$. We next define a crack, using the same radius $\delta_i$.

Definition 3.2. (Crack) Given a data set $D$ and the corresponding transformed data $D' = \tau(D)$, for each original value $u$ in $D$, we have a trans-
formed value $u'$ in $D'$. A domain crack function $g : D' \rightarrow D$ represents a guess made by a hacker. For a value $u'$ in $D'$, a guess is a crack if the guess falls within a radius $\delta_i$ from the actual value, i.e., $|g(u') - u| \leq \delta_i$. The domain disclosure risk is the fraction of the number of cracks over the number of values in $D'$.

We consider two possible attack models in this chapter. The first one is called curve fitting attack.

**Definition 3.3.** (Curve Fitting Attack) Suppose a hacker has $k$ domain knowledge points, i.e., $(w_1, u'_1), ..., (w_k, u'_k)$, for an attribute. The hacker can apply a curve fitting approach to fit these $k$ knowledge points into a crack function $g$, which is used by the hacker to crack the other transformed values in the same attribute.

The second attack model is global attack and is defined for matrix transformations. Recall that a $d$ dimensional data set $D$ containing $n$ tuples is represented as a $d \times n$ matrix. Each column is a tuple (data point) and each row is an attribute of the data set. Suppose $D$ is transformed by a $d \times d$ matrix $Q$, the transformed data set $D' = Q \times D$. For the custodian to decode the support vectors, she needs to use an invertible matrix. We have $D = Q^{-1} \times D'$. Global attacks refer to the situation when a hacker could make a guess about the transformation matrix $Q$ with her prior knowledge and further to crack the original data set $D$. Before we go to details of the definition of global attack, let us study an example first.

**Example 3.1.** Given a three dimensional data set containing six data points

$$D = \begin{pmatrix} 13 & 5 & 6 & 40 & 22 & 33 \\ 18 & 28 & 27 & 14 & 15 & 30 \\ 36 & 3 & 31 & 27 & 34 & 2 \end{pmatrix}$$

and a transformation matrix

$$Q = \begin{pmatrix} 0.78 & 0.62 & 0.08 \\ 0.29 & -0.47 & 0.84 \\ -0.55 & 0.63 & 0.54 \end{pmatrix},$$

the transformed data set $D' = Q \times D$. Suppose a hacker has some prior knowledge about the first three values in the first attribute of $D$ and guesses $\vec{w} = (w_1, w_2, w_3) = (13.5, 4.8, 6.5)$, which are very close to the true values $D_{(1,1:3)} = (13, 5, 6)$. Let us use $D_k$ to represent the first three data points
3.3. Knowledge Points and Attack Models

containing \( D_{(1,1:3)} \) in \( D \) and use \( D'_k \) to represent the corresponding three transformed data points in \( D' \).

Since the hacker has the transformed data set \( D' = Q \times D \), she can make a guess on the first row of the inverse matrix of \( Q \), i.e., \((Q^{-1})_{(1,:)}\), by using \( D' \) and \( \tilde{w} \) in the following way.

Let us use \( \hat{Q} \) to represent the matrix guessed by the hacker. She believes that \( D' = \hat{Q} \times D \) and \( \hat{Q}^{-1} \times D' = D \). Therefore, she reasons that \((\hat{Q}^{-1}) \times D'_k = D_k \). This means \((\hat{Q}^{-1})_{(1,:)} \times D'_k = D_{(1,1:3)} \). She does not know the original values of \( D_{(1,1:3)} \). Instead, she has guessed values \( \tilde{w} \) and believes that \( \tilde{w} \) is close to \( D_{(1,1:3)} \). Thus, she guesses \((\hat{Q}^{-1})_{(1,:)} \times D'_k = \tilde{w} \), which contains three linear equations of the three elements in \((\hat{Q}^{-1})_{(1,:)}\). By solving these three linear equations, the hacker can get the first row of matrix \((\hat{Q}^{-1})\), i.e., \((\hat{Q}^{-1})_{(1,:)} = \tilde{w} \times (D'_k)^{-1} = (0.76, 0.31, -0.53) \). Since \( Q \) is an orthogonal matrix, \( Q^{-1} \) equals to the transpose of \( Q \), i.e., \( Q^T \). Therefore, the first row of the inverse of \( Q \) is \((Q^{-1})_{(1,:)} = (0.78, 0.29, -0.55) \). We can see that the values of the guessed row \((\hat{Q}^{-1})_{(1,:)}\) are close to the actual values in the first row of the inverse of \( Q \), i.e., \((Q^{-1})_{(1,:)}\).

Once the hacker has \((\hat{Q}^{-1})_{(1,:)}\), she can use it to crack the first attribute of \( D \) by the equation \( \hat{D}_{(1,:)} = (\hat{Q}^{-1})_{(1,:)} \times D' \) and guesses that \( \hat{D}_{(1,:)} = (13.5, 4.8, 6.5, 39.6, 22.2, 32.1) \). Here, the first three values are the hacker’s knowledge, and the remaining three values are close to the actual values \( D_{(1,4:6)} = (40, 22, 33) \) in the original data set.

From the above example, we have the definition for global attack as follows:

**Definition 3.4.** (Global Attack). Suppose a hacker has \( d \) domain knowledge points in the \( i^{th} \) attribute of \( D \), i.e., \((w_1, u'_1), \ldots, (w_d, u'_d)\). Let \( \tilde{w} = (w_1, w_2, \ldots, w_d) \), where \( w_i \) is a guessed value. \( D'_k \) contains the tuples containing the \( d \) transformed values \((u'_1, \ldots, u'_d)\). She can guess out the \( i^{th} \) row \((\hat{Q}^{-1})_{(i,:)}\) of matrix \((\hat{Q}^{-1})\) by solving \((\hat{Q}^{-1})_{(i,:)} \times D'_k = \tilde{w} \), i.e., \((\hat{Q}^{-1})_{(i,:)} = \tilde{w} \times (D'_k)^{-1}\). \((\hat{Q}^{-1})_{(i,:)}\) is used to crack other values in the \( i^{th} \) attribute of \( D \) by the hacker.

Remark: Our definition of global attack is less strict, and thus riskier, than the matrix crack (e.g., distance inference attack) discussed in [9, 21]. In their work, a matrix attack requires \( d \) \( d \)-dimensional knowledge points. A \( d \)-dimensional knowledge point is an exact mapping between a transformed data point and the corresponding original data point across all attributes. They do not consider the situation when a hacker could use \( d \) 1-dimensional
knowledge points (i.e., domain knowledge points) in an attribute to make a
guess on the transformation matrix and further to crack the original data
set.

3.4 POT: Principled Orthogonal Transformation

3.4.1 Preliminaries

SVM Overview

Given a data set $D$ containing $n$ data points, each point $x_i$ is associated
with a class label $y_i \in \{-1, 1\}$. SVM uses the function

$$f(x) = \sum_{i}^{n} \alpha_i y_i K(x, x_i) + b$$

to form the classifiers from the training data set, where $\alpha_i$ and $b$ are pa-
rameters tuned by the training process and $K$ is a kernel function [10].

Geometrically, SVM classifiers minimize errors on the training set and sepa-
rates the rest of the elements with maximal margin in the feature space [5].

Specifically, for linear SVM, if the two classes of the data set $D$ are linearly
separable, SVM classification finds the optimal hyperplane that separates
those two classes and maximizes the margin between the two classes. If the
two classes are linearly non-separable, the optimal hyperplane should mini-
mize the classification errors while maximizing the margin between the two
classes.

SVM-Preserving Transformations

A data custodian sends the transformed data to the data mining service
provider, who conducts data mining tasks on the transformed data. After
the data custodian gets back the mining outcome (support vectors), the trans-
formation used allows the data custodian to easily decode the support
vectors. We have the following definition for the NOC guarantee for SVM.

Definition 3.5. (SVM-preserving Transformation) Given a data set $D$,
a transformation $\tau$, and the transformed data set $D' = \tau(D)$, let $V = \{v_1, \ldots, v_m\}$ be the support vectors in $D$. The transformation $\tau$ is called
an SVM-preserving transformation iff $V' = \tau(V) = \{\tau(v_1), \ldots, \tau(v_m)\}$ are
also the support vectors in $D'$.
Rotation is a transformation that transforms $D$ by applying a rotation matrix $M$, i.e., $D' = M \times D$. Rotation has been shown to preserve SVM in [8].

As our primary goal is to transform the data while having the NOC guarantee, we are interested in a class of transformations broader than random rotation. The POT method introduced in Section 3.4.3 transforms the original data set $D$ by an orthogonal matrix $Q$: $D' = Q \times D$. A transformation matrix $Q$ is orthogonal if $Q^T Q = QQ^T = I$, where $I$ is the identity matrix.

The proof in [8] mainly relies on the fact that a rotation matrix $M$ satisfies the property $M^T M = MM^T = I$. An orthogonal matrix $Q$, which is more general, also has the property $Q^T Q = QQ^T = I$. Therefore, orthogonal transformations are also SVM-preserving and give the NOC guarantee.

SVM classification has broad applications and can be applied on both categorical and numerical data. However, in this thesis, we mainly focus on numerical attributes.

### 3.4.2 Limitations of Random Transformations

In order to evaluate the privacy protection level of random transformations, i.e., random rotation [8] and random orthogonal transformation [21], we have conducted a set of experiments on benchmark data sets from the UC Irvine collection [29]. The details are shown in Section 3.6. Empirical studies show that random transformations are vulnerable to curve fitting attack even when a hacker has a small number of knowledge points. In the worst case, a hacker can crack more than half of the values in an attribute if the hacker only has 4 domain knowledge points in that attribute. For those “easy-to-crack” attributes, we found that the correlations between the original values and the corresponding transformed values are very high. The correlation coefficient $\rho$ between two random variables $X$ and $Y$ is defined as

$$\rho = \frac{\text{cov}(X, Y)}{\text{std}(X) \text{std}(Y)}$$

(3.1)

where $\text{cov}(X, Y)$ denotes the covariance between $X$ and $Y$ [17].

Intuitively, if the correlation is high, the fitting error is small when we fit a linear line between the original values and the transformed values. Therefore, when a hacker applies the curve fitting attack with her knowledge points to crack the transformed values, the percentage of the values to be cracked will be high too. In order to break down the strong linear correlations between original values and transformed values, we propose principled
orthogonal transformation (POT), which can enhance the protection of both input and output privacy.

3.4.3 Reducing Correlations by POT

Our technical objective here is to construct an orthogonal transformation while reducing correlation simultaneously. Figure 3.1 gives the outline of Algorithm 3.1 (POT-GenMatrix) which does exactly that. In this subsection, we give a general overview of the algorithm, using an example for illustration. Technical details for each step are given in the next subsection.

Example 3.2. Let us again use the data set $D$ discussed before, i.e.,

$$
D = \begin{pmatrix}
13 & 5 & 6 & 40 & 22 & 33 \\
18 & 28 & 27 & 14 & 15 & 30 \\
36 & 3 & 31 & 27 & 34 & 2
\end{pmatrix}
$$

Step 1 of the algorithm picks a random $d \times d$ matrix $A$, say, $A = \begin{pmatrix} 20 & 6 & 1 \\ 16 & 1 & 2 \\ 2 & 5 & 12 \end{pmatrix}$, which is orthogonalized to $Q$ by the Gram-Schmidt process. The details of orthogonalization are explained in Section 3.4.4. Data set $D$ is then transformed to $D' = Q^T \times D$. We use $Q^T$ to transform $D$ because we want to use the $j^{th}$ column of $Q$ to transform the $j^{th}$ attribute of $D$. Step 3 computes the correlations between respective rows of $D$ and $D'$. These correlations are sorted in descending order.

We orthogonalize $A$ to get $Q$, and we compute the correlation of each attribute between the original values in $D$ and the transformed values in $D'$. For attribute #1, Step 5.2 checks whether $\rho^1 > \rho^2_0$ (Equation 3.5). If $\rho^1 = 0.95$, now $\rho^1 = 0.95$, then the loop in Step 5.2 continues. It generates another vector $\bar{v} = (0, 0, -14)^T$ by solving an appropriate inequality. A new vector $A_{(:,1)} = A_{(:,1)} + \bar{v}$ is then computed in Step 5.2.3 and $A_{(:,1)} = (20, 16, -12)^T$. Based on this new $A_{(:,1)}$, the new correlation $\rho^1$ becomes $0.37$ (Equation 3.5). At this point, the algorithm has found a vector $A_{(:,1)}$ that can produce a smaller correlation than the threshold. New orthogonal vector $Q_{(:,1)}$ is computed in Step 5.4, using Equation 3.2.
3.4. POT: Principled Orthogonal Transformation

Algorithm 3.1: POT-GenMatrix

Input: $D$: d-dimensional training data set
       $\rho_0$: correlation coefficient threshold

1. Randomly generate $A$ and compute $Q$ by Equation 3.2 (the Gram-Schmidt process);
2. Get $D' = Q^T \times D$;
3. For each $i : 1, ..., d$, compute $\rho_i$ between $D_{(i,:)}$ and $D'_{(i,:)}$;
4. Sort $|\rho_i|$ in descending order and store this order in $\text{idx}$;
   Reorder attributes of $D$ by this order, i.e., $D = D_{(\text{idx},:)}$;
5. For $j = 1$ to $d-1$
   5.1 Compute $\rho_j$ by Equation 3.5;
   5.2 While $\rho_j^2 > \rho_0^2$
      5.2.1 Get a $\vec{v}$ by solving Inequality 3.6;
      5.2.2 Set $A_{(:,j)} = A_{(:,j)} + \vec{v}$;
      5.3 Compute $\rho_j$ by Equation 3.5;
   5.4 Compute $Q_{(:,j)}$ by Equation 3.2;
   }$
6. Compute $Q_{(:,d)}$ by Equation 3.2;
7. $Q_{(\text{idx},:)} = Q$;
8. Return $Q$;

Figure 3.1: Generating Orthogonal Matrix $Q$

For attribute #2, given the orthogonal vector $Q_{(:,1)}$ for the first attribute and $A_{(:,2)}$, Step 5.1 recomputes $\rho_2 = -0.82$, and Step 5.2 finds that $\rho_2^2$ is bigger than $\rho_0^2$. Similar to attribute #1, Step 5.2.1 generates a vector $\vec{v} = (0, 0, -16)^T$, and Step 5.2.3 updates $A_{(:,2)} = A_{(:,2)} + \vec{v} = (6, 1, -11)^T$. New correlation $\rho_2$ is computed in Step 5.3 and $\rho_2 = 0.22$. We can see the adjustment of $A_{(:,2)}$ does not change $\rho_1$ and $Q_{(:,1)}$, as will be shown in Lemma 3.1.

Finally, for attribute #3, the orthogonal vector $Q_{(:,3)}$ is computed in Step 6 and $\rho_3 = 0.48$. In Step 7, the newly generated orthogonal vectors $Q(:,j)$ are reordered into $Q$ based on $\text{idx}$, which is generated in Step 4. □
3.4.4 Technical Details and Equations

Orthogonalization

Orthogonalization is done by the usual Gram-Schmidt process on a linearly independent set of vectors $A$, i.e., $A = \{A(:,1), ..., A(:,d)\}$ [28]. Specifically, let us use $Q_{j-1}$ to represent the first $j-1$ generated vectors in $Q$ when we are going to compute the $j^{th}$ vector $Q(:,j)$, i.e., $Q_{j-1} = \{Q(:,1), ..., Q(:,j-1)\}$. The Gram-Schmidt process to compute $Q(:,j)$ can be expressed as [28, pp.57]:

$$T(:,j) = (I - Q_{j-1}Q_{j-1}^T) \times A(:,j), \quad Q(:,j) = \frac{T(:,j)}{||T(:,j)||} \quad (3.2)$$

To simplify the above equation, let $P_j$ denote $(I - Q_{j-1}Q_{j-1}^T)$. Thus, $T(:,j) = P_j \times A(:,j)$.

The Gram-Schmidt process requires that vectors in $A$ be linearly independent, which is not an issue for Algorithm 3.1 ($POT$-$GenMatrix$): if the vectors in $A$ are generated randomly (Step 1), the probability to have linearly dependent vectors in $A$ is negligible. For an example, if the elements of $A$ are uniformly selected from the integers between 1 and 100, this probability is $\frac{1}{100^{d-1}}$, which is very small.

Let us now consider the correlations between the original data and the transformed data. The transpose of $Q$ (i.e., $Q^T$) is also an orthogonal matrix. We use $Q^T$ to transform $D$ to $D'$, that is $D' = Q^T \times D$. In this way, the $j^{th}$ attribute of $D$ is transformed by the $j^{th}$ column of $Q$. Therefore, values in the $j^{th}$ attribute of $D'$ can be computed as $D'_{(i,j)} = (Q(:,j))^T \times D = Q_{(1,j)}D_{(1,:)} + ... + Q_{(d,j)}D_{(d,:)}$. The covariance between the $j^{th}$ attribute of $D$ and $D'$ and the standard deviation of $D'_{(j,:)}$ can now be expressed as:

$$cov(D_{(j,:)}, D'_{(j,:)}) = cov(D_{(j,:)}, (Q(:,j))^T \times D)$$
$$= (Q(:,j))^T cov(D_{(j,:)}, D)$$
$$= (Q(:,j))^T \times C_{(,:)} \quad (3.3)$$

$$std^2(D'_{(j,:)}) = var(D'_{(j,:)}) = (Q(:,j))^T \times C \times Q(:,j) \quad (3.4)$$

where the matrix $C$ is the covariance matrix of the original $D$, i.e., $C_{(i,j)} = cov(D_{(i,:)}, D_{(j,:)}), 1 \leq i, j \leq d$.

Based on Equations 3.2, 3.3 and 3.4, we can now rewrite the square of correlation coefficient $\rho_j$ as:

$$\rho_j^2 = \frac{cov(D_{(j,:)}, D'_{(j,:)} \times \text{std}(D_{(j,:)} \times \text{std}(D_{(j,:)}))}{\text{std}(D_{(j,:))}^2 \times \text{std}(D'_{(j,:)}))}$$

where $\rho_j$ is the correlation coefficient for the $j^{th}$ attribute.
3.4. POT: Principled Orthogonal Transformation

\[
\rho_j^2 = \frac{\text{cov}^2(D_{(j,:)}, D'_{(j,:)})}{\text{std}^2(D_{(j,:)}) \text{std}^2(D'_{(j,:)})} = \frac{(A_{(j,:)})^T \times S_1 \times A_{(j,:)}}{(A_{(j,:)})^T \times S_2 \times A_{(j,:)}},
\]

(3.5)

where \(S_1 = P_j^T \times C_{(j,:)} \times (C_{(j,:)})^T \times P_j\) and \(S_2 = \text{std}^2(D_{(j,:)}) \times P_j^T \times C \times P_j\).

Lemma 3.1. For \(1 \leq i < j\), \(Q_{(i,:)}\) and \(\rho_i\) are independent of \(A_{(j,:)}\).

Proof Sketch: For \(1 \leq i < j\), \(Q_{(i,:)}\) is computed from \(A_{(i,:)}, Q_{(i,1)}, ..., Q_{(i,i-1)}\) (Equation 3.2). \(\rho_i\) is computed from \(A_{(i,:)}, Q_{(i,1)}, ..., Q_{(i,i-1)}\) and covariance matrix \(C\) (Equation 3.5). Therefore, both \(Q_{(i,:)}\) and \(\rho_i\) are independent of \(A_{(j,:)}, A_{(j,:)}\) does not change any \(Q_{(i,:)}\) and \(\rho_i\), for \(1 \leq i < j\). \(\square\)

The above lemma states that the computation of the \(j\)th column of \(A\) in Step 5.2 of Algorithm 3.1 does not change the adjusted correlations of the previous columns.

**Correlation Reduction**

So far, we have explained how the orthogonalization can be done column-by-column incrementally. Below we explain how the correlations can be reduced.

Suppose we have a vector \(A_{(j,:)}\) and the corresponding coefficient based on \(A_{(j,:)}\) is \(\rho_j\). We are looking for a new vector \(\tilde{A}_{(j,:)}/(\tilde{A}_{(j,:)}) = A_{(j,:)}/(A_{(j,:)} + \tilde{v})\) that can produce a smaller correlation, which means \(\tilde{\rho}_j^2 = \frac{(\tilde{A}_{(j,:)})^T \times S_1 \times \tilde{A}_{(j,:)}}{(\tilde{A}_{(j,:)})^T \times S_2 \times \tilde{A}_{(j,:)}} < \rho_j^2 = \frac{(A_{(j,:)})^T \times S_1 \times A_{(j,:)}}{(A_{(j,:)})^T \times S_2 \times A_{(j,:)}}\) (Equation 3.5). \((\tilde{A}_{(j,:)})^T \times S_2 \times \tilde{A}_{(j,:)}\) has the same sign as \(\text{std}^2(D_{(j,:)}) \times \text{std}^2(D'_{(j,:)})\) and is a non-negative value. We have:

\[
\frac{A_{(j,:)}^T S_1 A_{(j,:)}}{A_{(j,:)}^T S_2 A_{(j,:)}} + 2S_1 A_{(j,:)} \tilde{v} + \tilde{v}^T S_1 \tilde{v} < \frac{\tilde{A}_{(j,:)}^T S_1 \tilde{A}_{(j,:)}}{\tilde{A}_{(j,:)}^T S_2 \tilde{A}_{(j,:)}} + 2S_2 \tilde{A}_{(j,:)} \tilde{v} + \tilde{v}^T S_2 \tilde{v} \quad (3.6)
\]

Inequality 3.6 is a quadratic function containing \(d\) unknown elements of \(\tilde{v}\). There are standard procedures for solving a quadratic inequality [14]. By solving Inequality 3.6, we can get a vector \(\tilde{v}\) and a new vector \(\tilde{A}_{(j,:)} = A_{(j,:)}/(A_{(j,:)} + \tilde{v})\) that can produce a smaller value of \(\tilde{\rho}_j^2\). Step 5.2 of the algorithm is designed to monotonically reduce \(|\rho_j|\) to be below a given threshold \(\rho_0\).
3.4.5 Properties of the Algorithm

In the example discussed in Section 3.4.3, the correlation $\rho_1$ is monotonically reduced by adjusting $A_{(1:1)} = A_{(1:1)} + \vec{v}$. The reason this works is that $\vec{v}$ comes from a solution of Inequality 3.6, which ensures that the new vector $A_{(1:1)}$ can produce smaller correlation $|\rho_1|$. From this property of monotonicity, we have the following theorem:

**Theorem 3.1.** Given a threshold $\rho_0$, for attributes $j : 1, \ldots, d - 1$, algorithm POT-GenMatrix always finds a vector $A_{(:,j)}$ that produces correlation $\rho_j$ s.t. $\rho_j^2 \leq \rho_0^2$.

Proof: Firstly, algorithm POT-GenMatrix monotonically reduces $\rho_j^2$ because $\vec{v}$ comes from a solution of Inequality 3.6. Secondly, we need to show that there is always a solution $\vec{v}$ for Inequality 3.6 if $\rho_j^2 > \rho_0^2$ (Step 5.2.2). Inequality 3.6 is not solvable only when $\rho_j^2$ has reached the minimum value, which is actually 0. The reason is as follows:

We have $\rho_j = \frac{\text{cov}(D_{(:,j)}, D'_{(:,j)})}{\text{std}(D_{(:,j)}) \cdot \text{std}(D'_{(:,j)})}$ (Equation 3.5) and $\text{cov}(D_{(:,j)}, D'_{(:,j)}) = (Q_{(:,j)})^T \cdot C_{(:,j)}$ (Equation 3.3), where $C_{(:,j)}$ is the $j^{th}$ column of covariance matrix $C$ and is a non-zero vector.

In $d$ dimensional space there must be a non-zero vector $q$ that is orthogonal to $C_{(:,j)}$ and $Q_{(:,i)}$, $1 \leq i < j$, which means $q^T \cdot C_{(:,j)} = 0$. Let $Q_{(:,j)} = q$, then $\text{cov}(D_{(:,j)}, D'_{(:,j)}) = 0$. The denominator of $\rho_j$ is $\text{std}(D_{(:,j)}) \cdot \text{std}(D'_{(:,j)})$, which is not 0 because $Q_{(:,j)}$ is a non-zero vector. Therefore, $\rho_j^2$ equals 0.

In conclusion, POT-GenMatrix can always iteratively find a new vector $A_{(:,j)}$ to monotonically reduce $\rho_j^2$ until $\rho_j^2 \leq \rho_0^2$.

Note that Step 5.2 of Algorithm 3.1 (Figure 3.1) loops until $\rho_j^2 \leq \rho_0^2$. Theorem 3.1 guarantees that the correlation reduces monotonically until $\rho_j^2 \leq \rho_0^2$. In Section 3.6.3, experimental results show that the number of loops is small (around 10), but it might converge slowly for some data sets. If the computational cost of the procedure is high, a data custodian could set a maximum number of loops for this step. In this case, however, the correlation may be higher than the threshold.

In algorithm POT-GenMatrix, Step 6 computes the $d^{th}$ vector $Q_{(:,d)}$ directly from the preceding $d - 1$ vectors $Q_{(:,1)}, \ldots, Q_{(:,d-1)}$. Because $Q$ is an orthogonal matrix, the last vector $Q_{(:,d)}$ is orthogonal to all the preceding $d - 1$ vectors. Therefore, in a $d$ dimensional space, if the preceding $d - 1$ vectors

---

5A rigorous convergence analysis of this algorithm may be rather challenging, and requires further assumptions on various characteristics of the data, such as its degree of variation.
3.5 Providing Extra Protection for Linear SVM: True Negative Point (TNP)

vectors of $Q$ have been fixed, the last vector $Q_{(:,d)}$ will be determined by those vectors. It is not able to change the last vector $Q_{(:,d)}$ by adjusting the vector $A_{(:,d)}$. This is why we reorder $D$ in descending order of $|\rho_j|$ in Step 3 of algorithm $POT$-GenMatrix. The attributes that have high correlation coefficients with the transformed attributes are processed early.

The complexity of algorithm $GenOrthMatrix$ contains two parts. First, the cost to transform $D$ and to compute the correlations in Steps 2-4 of Algorithm 3.1 is $O(nd^2)$, where $n$ is the number of points and $d$ is the number of attributes. Second, the cost to generate $A$ and $Q$ in Steps 1, 5, and 6 is $O(c \times d \times d^m)$, where $c$ is the iteration times for Step 5.2. $O(d^m), 2 < m \leq 3$ is the cost of matrix multiplication. Fast matrix multiplication is a well studied problem. There are several practical optimizations that have been developed for large $d$ [26]. In summary, the complexity of algorithm $GenOrthMatrix$ is $O(nd^2 + cd^{m+1})$. Experimental results in Section 3.6 show that POT can significantly reduce the crack percentage by reducing the correlation between the original values and the transformed values with low computational overhead.

3.5 Providing Extra Protection for Linear SVM: True Negative Point (TNP)

Section 3.4 shows that the POT approach can break the correlations between the original data and the transformed data. Experimental results shown later will demonstrate that the POT approach is effective in reducing the crack percentage against curve fitting attacks for both linear and non-linear SVM. However, if a hacker has enough knowledge points to conduct a global attack (Definition 3.4), POT might not be sufficient to protect the data. The following lemma shows the surprising result that the crack percentage is independent of transformations if a hacker has $d$ knowledge points in an attribute for a $d$ dimensional data set and she conducts a global attack.

**Lemma 3.2.** The crack percentage of global attack is independent of $Q$.

Proof Sketch: Given a data set $D$ and the transformed data $D' = Q \times D$, suppose a hacker has $d$ domain knowledge points in the $i^{th}$ attribute, i.e., $(w_1, u_1'), \ldots, (w_d, u_d')$. From Definition 3.4, the hacker can guess out a row of matrix $\bar{Q}^{-1}$, i.e., $(\bar{Q}^{-1})_{(i,:)} = \bar{w} \times D_k^{i-1}$, where $\bar{w} = (w_1 \ w_2 \ \ldots \ w_d)$ and the matrix $D_k'$ contains the $d$ tuples $D_{(:,j_k)}'$, $1 \leq k \leq d$ that are associated with the transformed values $u_k'$ ($j_k$ is the index of the transformed value $u_k'$). Therefore, $D_k'$ is a $d \times d$ matrix and $D_k' = (D'_{(:,j_k)})_{1 \leq k \leq d}$. Because
3.5. Providing Extra Protection for Linear SVM: True Negative Point (TNP)

\(D' = Q \times D\), we have \(D'_k = Q \times D_k\). Let us use \(D''\) to represent the data guessed by the hacker, and we have:

\[
D''_{(i,:)} = (\hat{Q}^{-1})_{(i,:)} \times D' = \bar{w} \times D'^{-1} \times D' = \bar{w} \times (Q \times D_k)^{-1} \times Q \times D = \bar{w} \times D^{-1} \times D
\]  

(3.7)

The transformation matrix \(Q\) does not appear in the above expression of \(D''\). Cracks indicate how close \(D''\) is to \(D\) (Definition 3.2). Therefore, the crack percentage of global attack is independent of \(Q\).

Lemma 3.2 shows that we are not able to reduce the crack percentage by adjusting the transformation matrix \(Q\) whenever the hacker has \(d\) domain knowledge points in an attribute and conducts a global attack. Notice that Lemma 3.2 also covers the matrix attack [9, 21], which is a special case of the global attack. In order to counter global attacks, we propose the true negative point (TNP) approach to enhance protection.

Recall that a hacker needs \(d\) domain knowledge points to execute a global attack (Definition 3.4). When \(d\) becomes larger (i.e., as dimensionality increases), it becomes harder for a hacker to have enough knowledge points [9] and the likelihood for a hacker to have ‘poor’ knowledge points, which are defined below, becomes higher too.

**Definition 3.6.** (Poor Knowledge Point) A knowledge point is poor means that the distance between the guessed value \(w\) and the actual value \(u\) is greater than the defined radius \(\delta_i\) (Definition 3.1), i.e., \(|w - u| > \delta_i\).

Experimental results (Section 3.6.2) show that even a single ‘poor’ knowledge point could significantly bring down a hacker’s crack ability. Thus, the TNP approach is primarily needed for low dimensional data sets when global attacks are possible.

The POT approach proposed in Section 3.4 is designed for both linear and non-linear SVM. However, for the TNP approach, we only focus on linear SVM in this chapter. We will extend the TNP approach against global attacks for non-linear SVM in our future work.

The general idea of the TNP perturbation is as follows. Support vectors are the points that determine the optimal separating hyperplane and form the classifier [10]. We call all the other points true negative points (i.e., TNPs). The NOC guarantee requires the support vectors be preserved. For all the TNPs, we can arbitrarily perturb them as long as they do not change the optimal hyperplane and hence the support vectors. To exploit this, we need a method for identifying TNPs. Below we propose a method to find a
3.5. Providing Extra Protection for Linear SVM: True Negative Point (TNP)

subset of TNPs and design a way to perturb them while providing the NOC guarantee. In Section 3.6, experimental results will show that even a small subset of TNPs can help significantly.

3.5.1 Linearly Separable Data Sets

The optimal hyperplane separates the two classes and maximizes the margin between the two classes for a linearly separable data set [10]. Geometrically, suppose we move the optimal hyperplane parallel to itself in the space until it hits any data point. Then the hitting points are the support vectors. The hyperplane passing through a support vector and parallel to the optimal hyperplane also separates the two classes. We have the following necessary condition for support vectors:

**Proposition 3.1.** A necessary condition for a point $p$ to be a support vector is that there is a hyperplane $h$ such that $h$ passes through point $p$ and separates the two classes.

This proposition helps us to identify a specific subset of TNPs. Let us illustrate the idea for 2-dimensional space. For any given two points $p, q$ in 2-dimensional space, we denote the line connecting $p, q$ as $f_{pq}$, which partitions the space into two half-spaces.

**Lemma 3.3.** (TNPs in 2-dimensional space) For any two points $p_1, p_2$ from class I and a point $w$ from II, let $m$ be the center of line segment $p_1p_2$. A class I point $q$ is a true negative point if the following conditions are satisfied:

1. $q$ and $w$ are from different half-spaces generated by $f_{p_1p_2}$;
2. $q$ and $m$ are from the same half-space generated by $f_{p_1,w}$; and
3. $q$ and $m$ are from the same half-space generated by $f_{p_2,w}$.

Proof Sketch: Figure 3.2 shows a set of points from Class I and Class II. Points $p_1, p_2$ are from class I (marked as crosses) and $w$ is from class II (marked as circles). $m$ is the center of line segment $p_1p_2$. Note that only points in the area $\Omega$ can satisfy all of these conditions; this is the area “below” $f_{p_1p_2}$ and bounded by $f_{p_1,w}$ and $f_{p_2,w}$ in Figure 3.2. For example, only points $p_3$ and $p_4$ satisfy these conditions.

For any points $q$ in the area $\Omega$, it is impossible to draw a line through $q$ that can separate both $p_1$ and $p_2$ from $w$, which means that we cannot find a line passing through $q$ that can separate class I and class II. From
Proposition 3.1, we know that $q$ cannot be a support vector and $q$ is a true negative point.

For $d$-dimensional data sets ($d > 2$), Lemma 3.3 is still applicable. The only difference is that we need to select at least $d$ points $p_i$ instead of 2. Based on Lemma 3.3, we propose Algorithm 3.2 (PerturbTNP) to find TNPs and perturb them. The basic idea is to form an “umbrella” as in Figure 3.2(a) to find TNPs. Hereafter, the points $p_i$ are called base points as they form the base of the umbrella, and point $w$ is called the top point.

In algorithm PerturbTNP, the input parameter $t_r$ is the ratio of TNPs to points in $D$ and indicates the minimum number of true negative points to be chosen. $k \geq d$ is a positive integer which specifies how many base points $p$ are to be selected. $maxCnt$ specifies the maximum number of loops of Step 2.

Step 1 initializes TNP set $T$. Setp 2 to Step 8 find TNPs iteratively. $S$ is a temporary variable to record the TNPs found in current loop and is initialized in Step 3. Step 4 selects a top point $w$ and Step 5 selects $k$ base points $p$. All base points $p$ are from the opposite class of $w$. Step 6 forms an umbrella shape as what we did in Figure 3.2(a). Each point $s$ that satisfies Lemma 3.3 is a TNP and is found in Step 7. Step 7.1 randomly perturbs $s$ to $s'$ in the area $\Omega$. Point $s$ is deleted from $D$. Point $s'$ is put into set $S$ and is finally put into TNP set $T$ in Step 8. The returned data set $D$ contains all left over points and the perturbed TNPs in $T$. Based on Lemma 3.3, we have the following theorem for algorithm PerturbTNP:

**Theorem 3.2.** (The NOC Guarantee) PerturbTNP preserves support vec-
3.5. Providing Extra Protection for Linear SVM: True Negative Point (TNP)

Algorithm 3.2: PerturbTNP

Input: $D$: data set; 
$t_r$: ratio of TNPs to points in $D$; 
$k$: a positive integer; 
$maxCnt$: a positive integer;

1. $T = Empty; cnt = 0$;
2. While $(\frac{|TNP|}{|D|} < t_r)$ and $(cnt \leq maxCnt)$;
3. Set $S = Empty$;
4. Select a top point $w$ from $D$;
5. Select $k$ base points $p = \{p_i|1 \leq i \leq k\}$ from the opposite class of $w$;
6. Build an umbrella shape as shown in Figure 3.2(a) based on points $w \cup p$;
7. For each point $s$ that satisfies Lemma 3.3
   
   \begin{itemize}
   \item Randomly move $s$ to $s'$ in area $\Omega$(Figure 3.2(a));
   \item Delete $s$ from $D$;
   \item Put $s'$ to $S$;
   \end{itemize}
8. $T = T \cup S$
   $cnt +=$;
9. Return $D = D \cup T$;

Figure 3.3: A Skeleton for the TNP Perturbation Approach

tors.

Proof Sketch: All points $s$ found in Step 7 of PerturbTNP are true negative points (Theorem 3.3). Deleting $s$ from $D$ does not change support vectors of SVM. Because $s'$ is a point generated by moving $s$ in the area $\Omega$, $s'$ also satisfies Lemma 3.3 and is also a true negative point in the perturbed data set. Therefore, adding $s'$ to $D$ also does not change support vectors of SVM. That is, PerturbTNP preserves support vectors.

The loop in Step 2 stops when either the requested percentage $t_r$ of TNPs are found or there maximum number of iterations is exceeded. Parameter $t_r$ is expected to be small (e.g., 10%, 30%). Experimental results given in Section 3.6 will show that only a small number of iterations are required for
the benchmark data sets and even with a small value of $t_r$ a high degree of protection is achieved.

In algorithm $PerturbTNP$, Step 5 selects $k$ base points. Intuitively, we can randomly select these base points. However, the number of TNPs captured in each loop is small for random selection. In order to improve the efficiency and convergence rate of algorithm $PerturbTNP$, especially for very large data sets, we propose another approach based on Skyline queries.

**Skyline Point:** A skyline point is a point that is not dominated by any other point [6]. Given two points $p$ and $q$, $p$ is said to dominate $q$ if $\forall 1 \leq i \leq d, p(i) \leq q(i)$. This means $p$ has smaller value than $q$ in each dimension. Figure 3.2(b) shows an example of skyline point. Points $a$, $b$ and $c$ all are skyline points. If we select $w$ as the top point, we can use $a$, $c$ and $w$ to form an umbrella shape like the one formed by $p_1$, $p_2$, $w$ in Figure 3.2(a).

In algorithm $PerturbTNP$, we can use the skyline operator to select $k$ base points (Step 5). Instead of finding all skyline points, we only need $k$ skyline points. The advantage of umbrella based on the skyline points is that the umbrella might be wider than random selection, which means more true negative points can be captured in Step 7 in each round.

**Complexity:** The complexity of $PerturbTNP$ is the combination of three parts, i.e., finding skyline points (Step 5), building umbrella (Step 6), and checking containment (Step 7). The complexity to find $k$ skyline points is $O(nk)$. Building umbrella is $O(k^{d/2})$. Containment checking is $O(nf)$ where $f$ is the number of faces of the umbrella which is exponential in $k$. For $c_s$ rounds selection, the total complexity is $O(c_s \times (nk + k^{d/2} + nf))$.

The most significant part of the complexity of $PerturbTNP$ if $k^{d/2}$. Let us discuss $d$ and $k$ respectively. First, the dimensionality $d$ is small because the TNP approach is purposely designed to counter global attacks, and recall that global attacks are mainly effective for lower dimensional data sets (Section 3.3.2 and paper [9]). Second, the number of skyline points $k$ if much less than the number of data points $n$. Bennett et al. find support vectors geometrically in [5]. Their approach needs to build convex hull over the whole data set and the complexity is $O(n^{d/2})$. In contrast, $PerturbTNP$ finds a subset of TNPs instead of support vectors. The number of base points (i.e., $k$) is significantly smaller than the number of points in $D$ (i.e., $n$). For a data set that contains hundreds of thousands of data points, we only need to set $k$ less than 100. Therefore, $PerturbTNP$ is practical and scalable even though there is a exponential part $k^{d/2}$ in its complexity.

In practice, we can use the algorithm $PerturbTNP$ to find and perturb TNPs first and then use POT to transform the perturbed data with orthogonal transformation. Because both the TNP perturbation and POT preserve
support vectors, we have the following corollary:

**Corollary 3.1.** Given a linearly separable data set $D$, the combination of the TNP perturbation and POT provides the NOC guarantee.

### 3.5.2 Linearly Non-Separable Data Sets

For linearly non-separable data sets, the two classes of data points intersect. No hyperplane can separate the two classes. Therefore, Proposition 3.1 does not hold for linearly non-separable data sets. Algorithm 3.2 (PerturbTNP) cannot be applied on such data sets directly. Since it is hard to derive purely geometric definition for support vectors for linearly non-separable data sets [10], we propose a heuristic to find negative points, which might include true negative and false negative points.

From the original data set, our heuristic tries to extract a linearly separable subset, on which the algorithm PerturbTNP can then be applied. We define a *shifting factor* to control the selection of this subset. We explain the details via the following example.

Figure 3.4 shows an example of a linearly non-separable data set. Given two classes I and II, we first find the centre of each class (e.g., $A$ is the center of class I and $B$ is the centre of class II). By connecting $A$ and $B$, we can get a line $AB$. $L$ is perpendicular to line $AB$ and passes through the centre of line segment $AB$. To create a linearly separable subset, we can first remove all class I points below $L$ and class II points above $L$. However, we would need to increase the “gap” between the two classes. This is achieved by moving $L$ parallel towards $A$ or $B$ and getting $L_1$ and $L_2$ respectively. The shifting factor, defined below, controls how far $L$ moves.
Definition 3.7. (Shifting Factor $sf$) Let $L_0$ be parallel to $L$ and be the tangent to Class I. Let $d$ be the distance between $L$ and $L_0$ and $d_1$ be the distance between $L_1$ and $L$. We define the shifting factor $sf = \frac{d_1}{d}$.

Instead of using the distance $d_1$, we use the shifting factor $sf$ to control the position of $L_1$ and $L_2$ to extract linearly separable subsets. We use the same shifting factor $sf$ for both classes. By deleting all Class I points below $L_1$ and all Class II points above $L_2$ in Figure 3.4, we get a linearly separable subset $D_s$ of $D$ (with no points in between $L_1$ and $L_2$). Algorithm \texttt{PerturbTNP} is then applied to $D_s$ directly. Once the identified TNPs from $D_s$ have been perturbed, we add back all the points deleted by the above heuristic and then apply \texttt{POT} to the resultant data set. Even though the \textit{NOC} guarantee is not provided, the experimental results below will show the effectiveness of this heuristic in minimizing the outcome change.

3.6 Experiments

We run our experiments on data sets from the UCI collection [29], including Forest CoverType, Boston Housing, Credit and WDBC (i.e., Wisconsin Breast Cancer (Diagnostic)). In this chapter, we deal with numerical attributes for all data sets. For WDBC, we use the first 10 numerical attributes that are the mean of the real-valued feature for the cell nucleus. Table 3.2 shows basic information about each data set.

<table>
<thead>
<tr>
<th># of Numeric Attrs</th>
<th>WDBC</th>
<th>Housing</th>
<th>Credit</th>
<th>CoverType</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Points</td>
<td>569</td>
<td>509</td>
<td>690</td>
<td>581012</td>
</tr>
</tbody>
</table>

Table 3.2: Data Sets

All these data sets are linearly non-separable. In our experiments, we use the ratio of the number of data points located between line $L_3$ and $L_4$ (Figure 3.4) to the number of data points in a data set to measure the degree of overlap between the two classes. Lines $L_3$ and $L_4$ are parallel to line $L$ and are tangent to Class II and Class I respectively. We observed that all data sets, except WDBC, are heavily overlapped (i.e., the ratio discussed above is greater than 0.9).

We evaluate the \texttt{POT} approach and the TNP perturbation with the curve fitting attack and the global attack. The radius $\delta$ (Definition 3.1) of the knowledge point is 2% of the 1% trimmed range of each domain, i.e., the range without the 0.5% biggest values and the 0.5% smallest values.
3.6. Experiments

In our experiments, we used linear regression, spline fitting and polyline fitting as curve fitting attacks. All of them showed similar results. In this section, we mainly show results for linear regression. We varied the radius $\delta$ for knowledge points, and observed that the trend of the results were the same as for $\delta$ is 2% of the trimmed range.

The experiments were implemented in MATLAB. We ran the experiments on an Intel Pentium PC with 3GHz CPU and 2GB RAM.

3.6.1 ICA Attack

Independent component analysis (ICA) [15] is considered as the attack model to reconstruct the original data from the transformed data while a hacker has no prior knowledge in [8]. The standard deviation of the difference between the reconstructed data and the original data is used to measure the protection level. Figure 3.5(a) shows the results for the ICA attack for the WDBC data set. In the figure, there are two groups of bars and each group contains four bars. The first group of bars show the minimum standard deviation of the difference between the reconstructed data and the original data for all attributes. The second group of bars show the average standard deviation of the difference. For each group of bars, the first bar shows the result for rotation and the other three bars show the results for POT with correlation threshold as 0.6, 0.3, 0.1 respectively. The results show that there is no significant difference between the rotation and POT regarding the protection against the ICA attack. The POT approach actually yield a little high difference than the random rotation approach, which means the POT approach provides more protection against the ICA attack.

![ICA Attack](image)

(a) WDBC  
(b) Housing

Figure 3.5: ICA Attack
3.6. Experiments

attack than the approach proposed in [8]. Figure 3.5(b) shows the results for the Housing data set and we can see results similar to those in Figure 3.5(a).

3.6.2 Knowledge Points

In this sub-section, we show a simple way for a hacker to gain knowledge points defined in Definition 3.1 and present the experimental results showing that how a hacker could use those knowledge points to crack the random transformations.

Figure 3.6 shows experimental results about the crack ability if a hacker maps statistics (such as the maximum, the minimum, the median, and the quantiles) of the transformed domain to the original domain. The data sets are transformed by random rotation transformations.

![Figure 3.6: Sources of Knowledge Points (Kps)](a) WDBC (b) Credit

We can see from Figure 3.6(a) that the effectiveness of knowledge points (w.r.t. crack percentage) varies across the attributes. For attribute #4 of WDBC, even if the hacker only maps the maximum and the minimum of the transformed domain to the original domain, she can still crack about 30% of the values. If she also has the knowledge of the quantiles of the original domain, the crack percentage can go up close to 38% for attribute #4. Similar results for the Credit data set are shown in Figure 3.6(b).

---

*Experimental results show that random rotations [8] and random orthogonal transformations [21] have the similar behaviour on the crack analysis. Due to space requirements, we only show experiments for random rotations in the following parts of Section 3.6 when we talk about random transformations.*
3.6 Experiments

3.6.3 Effectiveness of POT Against Curve Fitting Attacks

Figure 3.7(a) shows the crack percentage of the attribute that has the highest crack percentage in each data set. We set the number of knowledge points to 4. The threshold of correlation coefficient is set to 0.6. The correlations for different transformations are shown in Table 3.3. We can see that the crack percentage is very high for random rotations. The reason is that the correlation ρ is high (e.g., the crack percentage for WDBC is about 60% and the correlation is 0.99). POT can significantly reduce both the correlation and the crack percentage. All of the crack percentages are less than 10% for POT.

Figure 3.7: Privacy Analysis of POT vs. Rotation

Figure 3.7(b) shows the crack percentage for each attribute of WDBC and Figure 3.7(c) shows the crack percentage for each attribute of Cover-
3.6. Experiments

<table>
<thead>
<tr>
<th></th>
<th>WDBC</th>
<th>Housing</th>
<th>Credit</th>
<th>CoverType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Rotation</td>
<td>0.99</td>
<td>0.74</td>
<td>0.98</td>
<td>0.71</td>
</tr>
<tr>
<td>POT</td>
<td>0.36</td>
<td>0.47</td>
<td>0.32</td>
<td>0.11</td>
</tr>
</tbody>
</table>

Table 3.3: Correlations

Type. Generally, we can see that POT provides significantly more protection than rotation. For all attributes with high crack percentage in rotation transformations (e.g., attributes #1, #3, #4 in WDBC and #6 and #10 in CoverType), the crack percentage for POT drops significantly. For other attributes, both the correlation coefficient and the crack percentage are already very small even for random rotations, which means there is not much room for improvement.

Figure 3.7(d) shows the result for attribute #10 of CoverType with different correlation thresholds. Generally, the smaller the correlation threshold $\rho_0$, the smaller the crack percentage. Notice that when $\rho_0$ is below 0.6, the resulting drop in crack percentage becomes insignificant. We also get similar results for other data sets.

Impact of Knowledge Points

Figure 3.8 shows the crack percentage for attribute #4 of WDBC against the number of knowledge points the hacker has. The graph shows that if we only transform the data with random rotations, the crack percentage increases rapidly with the increasing of the number of knowledge points. However, POT makes it harder for the hacker to guess out more data even when she has more knowledge points.

![Figure 3.8: Impact of the Number of Knowledge Points](image)

Figure 3.8 also confirms the effectiveness of the combination of POT
and the TNP perturbation. The privacy protection provided by POT will not be hurt by combining the TNP perturbation. Actually, POT and the TNP perturbation together could provide more protection than POT only to counter curve fitting attacks. Recall that the TNP perturbation is designed to counter global attacks. Experimental results about how the TNP perturbation counters global attacks will be shown in the next sub-section.

### Impact of Radius \( \delta \)

We introduce a radius \( \delta \) in the definition of knowledge points (Definition 3.1). Figure 3.9 shows the crack percentage w.r.t. different radius \( \delta \) for attribute \#4 of WDBC with 4 knowledge points. All above results are based on the radius \( \delta \) to be 2\% of the domain range. In Figure 3.9, we vary the radius from 2\% to 10\% of the domain range. For each radius, we always have lower crack percentage (higher protection level) for the POT approach than the random rotation approach.

![Figure 3.9: WDBC : Radius \( \delta \)](image)

### Runtime

Table 3.4 shows the experimental results of the cost of \( POT\text{-}GenMatrix \) on the large data set (i.e., CoverType) in terms of running time and the number of iterations. From empirical studies, even for very small correlation threshold (e.g., \( \rho = 0 \)), the running time of \( POT\text{-}GenMatrix \) is less than 2 seconds. The total number of iterations of Step 5 is 120. For any single attribute, Step 5 loops less than 20 times.

From the crack analysis in Figure 3.7 and the cost of algorithm \( POT\text{-}GenMatrix \) shown in Table 3.4, we can conclude that \( POT\text{-}GenMatrix \) significantly enhances the protection levels on the original data for a very small cost.
3.6 Experiments

<table>
<thead>
<tr>
<th>$\rho_0$</th>
<th>0.9</th>
<th>0.6</th>
<th>0.3</th>
<th>0.1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>running time (secs)</td>
<td>1.7</td>
<td>1.75</td>
<td>1.78</td>
<td>1.84</td>
<td>1.9</td>
</tr>
<tr>
<td>total number of iterations</td>
<td>2</td>
<td>4</td>
<td>15</td>
<td>36</td>
<td>120</td>
</tr>
</tbody>
</table>

Table 3.4: Cost of POT-GenMatrix

3.6.4 Effectiveness of TNP Against Global Attacks

Global Attacks

Figure 3.10 shows the results of a global attack on WDBC transformed using POT with the TNP approach incorporated for various values of the TNP ratio $t_r$. We set shifting factor $sf$ to be 0.1 and the percentage of TNPs (i.e., $t_r$) to be 0%, 10% and 20%. We can see that the crack percentage drops quickly for all the attributes with TNPs. Even for $t_r = 10\%$, there is a substantial reduction in crack percentage. However, the difference of the crack percentage between $t_r = 10\%$ and $t_r = 20\%$ is not much. Therefore, for Algorithm 3.2 ($PerturbTNP$), we do not need large values of the parameter $t_r$ for reducing the crack percentage significantly.

![Figure 3.10: Global Attack on WDBC](image)

Table 3.5 shows the crack percentage under global attack for all the data sets. Based on the results shown in Table 3.5, we can see the TNP perturbation is very effective for countering global attacks relative to random rotations. The more the true negative points, the smaller the crack percentage. In all data sets, the inclusion of TNPs yields significant reduction in the crack percentage.

For Credit, the crack percentage is still 24% even when there are 20% TNPs. The reason is that Credit is a heavily overlapped data set, i.e., the
3.6. Experiments

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Random Rotation</th>
<th>POT without TNP</th>
<th>POT with TNP $t_r = 10%$</th>
<th>POT with TNP $t_r = 20%$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDBC</td>
<td>39%</td>
<td>39%</td>
<td>7.2%</td>
<td>7.1%</td>
</tr>
<tr>
<td>Housing</td>
<td>34%</td>
<td>34%</td>
<td>7.9%</td>
<td>6.5%</td>
</tr>
<tr>
<td>Credit</td>
<td>48%</td>
<td>48%</td>
<td>35%</td>
<td>24%</td>
</tr>
<tr>
<td>CoverType</td>
<td>36%</td>
<td>36%</td>
<td>17%</td>
<td>11%</td>
</tr>
</tbody>
</table>

Table 3.5: Global Attacks on All Data Sets

overlapped part of the two classes is big in Figure 3.4. For $sf = 0.1$, there are only about 200 points in $D_s$. Therefore, 20% TNPs means we only have 40 of them, which is too few with respect to the whole Credit data set containing 690 data points. In our experiments, we observed that if $sf$ is 0 and $t_r$ is 40%, we can get 150 TNPs and the crack percentage drops to 12%.

We also evaluate the cost of algorithm $PerturbTNP$. For small data sets (e.g., WDBC), it takes about 10 seconds to find $t_r = 20\%$ TNPs. For a large data set like CoverType, if the shifting factor $sf = 0.1$ and $t_r = 10\%$, it takes about 200 seconds to find TNPs. Given the effectiveness of the TNP perturbation against global attacks, we feel that the time taken is more than justified.

Impact of Poor Knowledge Points on Global Attacks

Recall that A hacker needs $d$ domain knowledge points to conduct a global attack on the transformation matrix (Definition 3.4). However, there is no guarantee on the quality of the hacker’s knowledge points. If a knowledge point is poor (Definition 3.6), it can dramatically limit the crack power of the hacker. Figure 3.11 shows the experimental results of the impact of poor knowledge points for WDBC.

In the experiments, we chose poor knowledge points at a distance between 2\% and 10\% of the trimmed range from the actual values. In the real word, a poor knowledge point might be far more away from the actual value. Figure 3.11 shows that even a single poor knowledge point can make the crack percentage drop almost half on all attributes. The crack percentage further drops to around 10\% if there are two poor knowledge points. And further, if 20\% TNPs are used to perturb the data, the crack percentage drops to around 5\%.

From experimental results we can know that even a single or two poor knowledge points can dramatically bring down a hacker’s crack ability. Meanwhile, the higher the dimensionality of the data sets, the more chance that
3.6. Experiments

a hacker might get poor knowledge points. Therefore, global attacks and
the TNP approach are mainly useful for low dimensional data. For high
dimensional data, POT by itself can be very effective.

3.6.5 Minimizing the Outcome Change Against Global
Attacks

The other part of experiments aims at evaluating the outcome change caused
by the TNP perturbation in linearly non-separable data sets (Recall that the
POT approach provides the NOC guarantee for both linear and non-linear
SVMs, and on linearly separable data sets the TNP approach provides the
NOC guarantee). After the data custodian gets the transformed support
vectors from the mining service provider, she decodes the transformed sup-
port vectors to original values and builds a classifier, which is called the
classifier after the TNP perturbation. The classifier directly built on the
original data set is called the classifier before the TNP perturbation.

For each data set, we measure the classification accuracy ($r_{ac}$) before and
after the TNP perturbation. We also measure the disagreement between the
classification results before and after the TNP perturbation. Disagreement
($r_{dg}$) is the ratio of the number of points that have been classified into
different classes by the classifiers built before and after the TNP perturbation.
Recall that $t_r$ denotes the TNP ratio and $sf$ denotes the shifting factor
(Definition 3.7).

Table 3.6 shows classification accuracy and disagreement before and after
the TNP perturbation. From previous studies ([4, 35]), it is computationally
prohibitive to train SVM on CoverType. Therefore, we use data sets WDBC,
Housing and Credit for this set of experiments.
3.6. Experiments

The first part of Table 3.6 shows the accuracy $r_{ac}$ of the classifiers before the TNP perturbation. The second part of Table 3.6 shows the accuracy and disagreement of the classifiers after the TNP perturbation. The shifting factor $sf$ is set to 0.1 as for experiments shown in Table 3.5.

For the WDBC data set, even with 20% TNPs, the accuracy ($r_{ac}$) is 91.2%, which is still very close to the original classification accuracy 91.9% with 1% disagreement ($r_{dg}$). There are more disagreements for Housing and Credit data sets than WDBC. The reason is that the two classes are heavily overlapped in both Housing and Credit data sets. However, even for the Credit data set, the accuracy changes from 82.1% to 80.5% with 10% TNPs. The outcome is not significantly affected.

The overlap part of a data set should be small for linear SVM to be effective [5], which our TNP perturbation approach is designed for. Using non-linear SVM on heavily overlapped data sets might call for new perturbation techniques for the TNP. This is an interesting direction for future work.

### TNP v.s. Random Perturbation

Even though the TNP approach could provide the NOC guarantee for linear SVM on linearly separable data sets, in the case of linearly non-separable data sets, it changes the optimal classifiers. Recall that random perturbation typically changes the optimal classifiers. We compare the TNP approach and random perturbation with respect to classification accuracy for linearly non-separable data sets and show the experimental results in Figure 3.12.

The x-axis shows the standard deviation of the added noise, which varies from 10% to 30% of the domain range. The y-axis shows the classification accuracy. The top solid line shows the accuracy of the classifier on the original data without adding noise. The two dashed lines show the classification accuracy for the TNP approach. The dot line with the cross marker shows the result for the random perturbation approach. We can see that the TNP approach always has significantly higher classification accuracy than...
the random perturbation approach.

### 3.6.6 Intended Usage of POT and TNP

One of the motivations to outsource data mining in the data custodian model is to leverage not only the computational resources but also the data mining expertise offered by the service provider. Given the POT approach and the TNP perturbation techniques developed in this chapter, a natural question to ask is how they are meant to be used by a data custodian. In particular, while the custodian does not need to possess sophisticated expertise related to SVM, are these transformations likely to demand some other expertise from the custodian and if so how complicated can that get? In this section, we address this question.

Recall that both POT and TNP require certain parameters to be set for their application (e.g., correlation coefficient $\rho$ for POT and ratio $t_r$ for the TNP perturbation)$^7$. Clearly, choosing these values without any care might result in a transformation whose privacy protection could be poor. Does the custodian have to do any complicated analysis of the data in order to choose the “right” values $\rho$ and $t_r$?

We conduct a comprehensive experimental analysis of both POT and TNP and discuss our findings in the preceding sub-sections. Based on that, we recommend that the custodian can choose the default values of $\rho = 0.6$ and $t_r = 10\%$. Our empirical analysis shows that with this setting, the crack percentage will be less than 10\% for all the data sets we tested. Should the

---

$^7$Other parameters for the algorithm PerturbTNP are internal parameters and they will not affect the privacy protection.
3.7. Conclusion and Future Work

In this chapter, we study privacy preservation in outsourcing SVM classification. Our goal is to protect input and output privacy while minimizing the outcome change. We propose two approaches, viz., POT and TNP, that accomplish this goal.

We first show that POT provides the no-outcome-change (NOC) guarantee for both linear and non-linear SVM. Thus, the data mining service provider could train the SVM classifier on the transformed data (including kernel selection, parameter tuning, etc.), which saves the data custodian the need to acquire substantial expertise. The algorithm POT-\textit{GenMatrix} monotonically reduces the correlation between the original values and the transformed values to a given threshold. Empirical studies show that POT significantly reduces the crack percentage by reducing the correlation at small computational cost.

We show the surprising result that if a hacker has $d$ domain knowledge points for an attribute, the crack percentage achievable by global attacks is independent of the transformation.

To counter global attacks, we develop the TNP approach. The TNP approach (i.e., the algorithm \textit{PerturbTNP}) finds and perturbs (a subset of) true negative points. We prove that the TNP approach provides the NOC guarantee for linear SVM on linearly separable data set. For linearly non-separable data set, we propose a heuristic to extract a linearly separable subset, to which \textit{PerturbTNP} can be applied. A shifting factor $sf$ is defined to control the size of the linearly separable subset that is extracted from the original data set. Experimental results show that the TNP approach works very well to counter global attacks and significantly reduces the crack percentage without compromising too much classification accuracy.

In summary, we propose two approaches, i.e., POT and TNP, to provide the three pillars of privacy preservation in outsourcing of SVM classification. POT provides the NOC guarantee for both linear and non-linear SVM. The TNP approach is designed for linear SVM when a global attack is possi-
3.7. Conclusion and Future Work

ble, which is typically true for low dimensional data. Extending the TNP approach to deal with non-linear kernel functions is an interesting open problem.
3.8 Bibliography


3.8. Bibliography


Chapter 4

Extension to POT

4.1 Introduction

Privacy preservation is a key issue in the data-mining-as-a-service model [6, 11]. In this model, there are two scenarios, i.e., the data collector scenario and the data custodian scenario [3]. In the data collector scenario, individual data owners submit their data to a data collector to conduct data mining tasks [2]. In the other scenario, a data custodian, who has the responsibility to protect the data, outsources data mining tasks to an outside data miner [3, 18].

For both scenarios, a perturbation or transformation approach needs to be applied on the original data to protect the inherent private information. Random perturbation is the dominant approach to protect the privacy in the data collector scenario [1, 2, 8]. Random noise drawn from a given distribution is added to the original data. In order to provide more protection, the approach needs more noise, and at the same time there will be more change on the data mining outcome [2]. In the data custodian scenario, several studies show that it is possible to preserve the mined patterns in outsourcing of association rule mining [18], kNN computation [20], etc.

There are three pillars to privacy preservation, i.e., the input privacy, the output privacy, and minimizing the outcome change [3]. The input privacy aims to protect the private information inherent in the original data. The output privacy is to disguise the mined patterns to avoid malicious utilization. While protecting the privacy, we also want to derive meaningful data mining results, which means to minimize the data mining outcome change after transformation. After we apply a transformation on the original data, if we could still get the same data mining patterns as the patterns mined from the original data, we say that this transformation provides the no-outcome-change (NOC) guarantee.

\footnote{A version of this chapter will be submitted for publication: Bu, S., Lakshmanan, L. V.S., and Ng, R. T.: Enhancing Privacy Preservation with Prior Knowledge for Classification}
In this chapter, we focus our study on privacy preservation for classification, especially for the SVM classification and linear classification models, which include linear regression, logistic regression, etc. The orthogonal transformation has been shown to provide the NOC guarantee for the SVM classification and a principled orthogonal transformation (POT) has been proposed to generate orthogonal transformations in chapter three. In this chapter, we extend POT to other linear classification models and make the following contributions.

Firstly, we investigate the POT approach to enhance the protection on privacy in the worst cases. Generally, the POT approach could provide high level protection by reducing the correlation between the original values and the transformed values. However, it is possible that a hacker could still guess out a significant number of values based on her knowledge. The reason is that even when the correlation between the original values and the transformed values in a domain is low there might exist subsets of the original values that still have high correlation with the corresponding transformed values. We propose a heuristic to further break down the correlations between the original values and the transformed values among those subsets.

Secondly, we extend the POT approach to linear classification models. We show that the orthogonal transformation also provides the NOC guarantee for linear classification models including linear regression, ridge regression and logistic regression.

Thirdly, we propose to combine the POT approach and the random perturbation approach to transform the original data. After we use random perturbation to add noise on the original data, we use POT to generate an orthogonal transformation to transform the perturbed data. For a given desired level of privacy protection, the combination of POT and random perturbation needs less noise than random perturbation alone and thus leads to a reduction in the outcome change.

Finally, we conduct a comprehensive set of experiments to evaluate the proposed approaches.

4.2 Related Works

Private information might be breached if the original data is not protected while data owners submit their data to an outside data mining service provider [6, 17]. In order to protect privacy, researchers have proposed different approaches to transform the data. Several studies show that it is possible to generate a transformation that could preserve both the privacy
and the data mining patterns. Wong et al. propose an approach based on one-to-n mapping for secure outsourcing of association rule in [18]. Molly et al. show that the one-to-n mapping can be reduced to a one-to-one mapping and can be broke under the attacks with known frequency knowledge in [13]. Wong et al. propose an audit approach for frequent item mining in [19], and an approach that preserves the orders of the distance between data points for kNN computation in [20]. We propose piece-wise transformation for decision tree classification in [3]. All these methods provide the NOC guarantee while preserving privacy. However, these methods are not suitable for the SVM classification and linear classifications.

Chen et al. use random rotation to transform the original data for the SVM classification [4, 5]. Random rotation is vulnerable to curve fitting attacks while a hacker has prior knowledge. The POT approach proposed in chapter three generates an orthogonal transformation to protect privacy in outsourcing of the SVM classification. POT is effective to counter attacks with prior knowledge.

Random perturbation is a widely studied approach to perturb the original data by adding random noise [1, 2, 7, 8, 10, 12, 21]. Data owners only submit the perturbed data to an outside data miner. The original data is protected by the added noise. The more noise added, the more protection on the private information, meanwhile the more change on the data mining outcome. Prior knowledge is not counted in the privacy analysis for random perturbation in previous works. In this chapter, we evaluate the random perturbation approach against attack models with prior knowledge and apply POT on random perturbation to enhance protection and minimize the outcome change.

4.3 Preliminaries

4.3.1 Attack Models with Prior Knowledge

We introduce attack models with prior knowledge in this section. As discussed in the previous chapters, there are various sources for a hacker to gain prior knowledge about the original data. We abstract all these forms of the prior knowledge to an abstract notion of knowledge points as defined below.

Definition 4.1. (Domain Knowledge Point) Given a value \( u \) in attribute \( i \) of the original data set \( D \) and the corresponding value \( u' \) in the transformed data set \( D' \), a hacker believes that \( u' \) is transformed from \( w \). We say that
4.3. Preliminaries

\((w, u')\) is a domain knowledge point if \(|w - u| \leq \delta_i\), where \(\delta_i\) is a given parameter for attribute \(i\).

The parameter \(\delta_i\) defines the distance between the value \(w\) and the real value \(u\). In other words, the hacker’s knowledge (i.e., the value \(w\)) is within a radius of \(\delta_i\) from the true original value. This radius, \(\delta_i\), is defined as a fraction of the domain range of attribute \(i\). For example, we can define \(\delta_i\) to be two percent of the domain range. A hacker could make a guess on the original data based on her prior knowledge. We define a crack using the same radius \(\delta_i\) as follows:

**Definition 4.2.** (Crack) For a value \(u'\) in the transformed data set \(D'\), if the value guessed by a hacker’s crack function \(g\) falls in a radius \(\delta_i\) from the actual value \(u\), i.e., \(|g(u') - u| \leq \delta_i\), we call the guess \(g(u')\) a crack.

The domain disclosure risk of attribute \(i\) is the fraction of the number of cracks over the number of values in attribute \(i\) of \(D\).

In this chapter, we consider the curve fitting attack as the attack model with prior knowledge.

**Definition 4.3.** (Curve Fitting Attack) Suppose a hacker has \(k\) domain knowledge points, i.e., \((w_1, u'_1), \ldots, (w_k, u'_k)\), for an attribute. The hacker can fit these knowledge points into a crack function \(g\) by a curve fitting approach to crack other transformed values in the same attribute.

4.3.2 POT : Principled Orthogonal Transformation

As shown in the previous chapter, the POT approach could enhance the protection level on private information. POT is a method based on the Gram-Schmidt procedure ([15]) to iteratively and monotonically reduce the correlation between the original values and the corresponding transformed values.

In the Gram-Schmidt procedure, a \(d \times d\) orthogonal matrix \(Q\) is computed from a linearly independent set of vectors \(A = \{A(:,1), \ldots, A(:,d)\}\) [15]. Let us use \(Q_j - 1\) to represent the first \(j - 1\) vectors in \(Q\), i.e., \(Q_{j-1} = \{Q(:,1), \ldots, Q(:,j-1)\}\). Let us use \(P_j\) to denote \((I - Q_{j-1}Q_{j-1}^T)\). The method to compute the \(j^{th}\) vector of \(Q\) is:

\[
T(:,j) = (I - Q_{j-1}Q_{j-1}^T) \times A(:,i), \quad Q(:,j) = \frac{T(:,j)}{||T(:,j)||}
\]  

(4.1)
4.4 Optimizing POT : POT for Linear Sub-groups

Given a data set \( D \) and its \( j \)th attribute \( D_{(j,:)} \), the transformed data set \( D' = Q^T \times D \) and the transformed \( j \)th attribute is \( D'_{(j,:)} = (Q_{(j,:)})^T \times D_{(j,:)} \). The correlation between the original attribute \( D_{(j,:)} \) and the transformed attribute \( D'_{(j,:)} \) is:

\[
\rho_j^2 = \frac{\text{cov}^2(D_{(j,:)}, D'_{(j,:)})}{\text{std}^2(D_{(j,:)}) \times \text{std}^2(D'_{(j,:)})} = (A_{(j,:)}^T \times S_1 \times A_{(j,:)} \times S_2 \times A_{(j,:)}^T)
\]  

where

\[
S_1 = P_j^T \times C_{(j,:)} \times (C_{(j,:)})^T \times P_j
\]

\[
S_2 = \text{std}^2(D_{(j,:)}) \times P_j^T \times C \times P_j
\]

The matrix \( C \) is the covariance matrix of the original data \( D \), i.e., \( C_{(i,j)} = \text{cov}(D_{(i,:)}, D_{(j,:)}), 1 \leq i, j \leq d \).

The POT approach looks for a new vector \( \tilde{A}_{(j,:)} = A_{(j,:)} + \tilde{v} \) that can produce a smaller correlation by solving the following inequality, which has been shown in chapter three:

\[
\frac{A_{(j,:)}^T S_1 A_{(j,:)} + 2S_1 A_{(j,:)} \tilde{v} + \tilde{v}^T S_1 \tilde{v}}{A_{(j,:)}^T S_2 A_{(j,:)} + 2S_2 A_{(j,:)} \tilde{v} + \tilde{v}^T S_2 \tilde{v}} < \rho_j^2
\]

\[
2S_1 A_{(j,:)} \tilde{v} + \tilde{v}^T S_1 \tilde{v} < 2\rho_j^2 S_2 A_{(j,:)} \tilde{v} + \rho_j^2 \tilde{v}^T S_2 \tilde{v}
\]  

4.4 Optimizing POT : POT for Linear Sub-groups

In chapter three, experimental results have shown that POT is effective to counter curve fitting attacks with prior knowledge, which means the average crack percentage has been reduced substantially to a low level. However, we found that the crack percentage might be still high in particular settings of knowledge points. The reason is that there exist subsets of the values that still have high correlation with the corresponding transformed values even when the correlation of the whole domain is low. Let us study an example first.

\(^9\)The vector generated by the Gram-Schmidt procedure is a column vector. In order to use a generated column vector to transform data, we use \( Q^T \) as the transformation matrix. Notice that \( Q^T \) is also an orthogonal matrix.
4.4. Optimizing POT: POT for Linear Sub-groups

4.4.1 A Motivating Example

In Figure 4.1 (a) and (b), the x-axis shows the original values and the y-axis shows the transformed values. Each point in the figure is a pair of an original value and the corresponding transformed value. Thus the distribution of the points in the space shows the relationship between the original values and the transformed values. The two transformations shown in Figure 4.1(a) and (b) yield the same correlations ($|\rho| = 0.6$) between the original values and the corresponding transformed values.

(a) Without Linear Sub-groups  
(b) With Linear Sub-groups

Figure 4.1: Examples of Transformations

From Figure 4.1(a) we can see that all the points are scattered in the space. However, in Figure 4.1(b), we find two linear sub-groups. The correlation coefficient for one sub-group (marked as ‘dot’) is 0.92 and the correlation coefficient for the other sub-group (marked as ‘circle’) is -0.91. Both of the two sub-groups are highly correlated. If a hacker’s knowledge points all fall in the ‘dot’ group, or all fall in the ‘circle’ group, she might be able to crack a significant number of values. On the contrast, this situation will not happen for the transformation in Figure 4.1(a).

4.4.2 A Heuristic: POT for Linear Sub-Groups

We do not want to use an orthogonal transformation that will yield highly correlated linear sub-groups like the one shown in Figure 4.1(b). In order to generate transformations like the one shown Figure 4.1(a), we propose an approach to break down the correlations for linear sub-groups. We use the linear grouping algorithm (the Lga method [14]) to check whether a
4.4. Optimizing POT : POT for Linear Sub-groups

A linear sub-group exists. The Lga method assigns values into linear groups by minimizing the squared orthogonal residuals within each group [14]. The Lga method needs to know the number of groups (i.e., clusters). The GAP statistics is used to estimate the number of clusters by checking the decrease of the ‘within clusters dispersion’ while increasing the number of clusters $K$ [16]. The proposed procedure is described as follows:

- **POT** : generating orthogonal transformation.
  The POT approach generates a vector, which is a column of the generated orthogonal transformation, for each attribute respectively. Once a vector has been generated for an attribute, we could form a two dimensional data set containing the original values and the transformed values for this attribute.

- **GAP** : estimating the number of clusters.
  We use the GAP statistics to estimate the number of clusters in the formed two dimensional data set for this attribute.

- **Lga** : finding linear groups.
  If there is more than one cluster found by GAP, we apply the Lga method to form linear sub-groups.

- **Breaking down correlations for linear sub-groups:**
  We check the correlation for each sub-group. If there exists a sub-group that has higher correlation than the predefined threshold, then this sub-group does not satisfy the privacy requirement. We propose a heuristic to break down the high correlations in linear sub-groups to further enhance the protection on the original data.

Let us assume that there are $K$ linear sub-groups found by Lga for the $j^{th}$ attribute after transformation. For the $l^{th}$ (1 $\leq l \leq K$) linear sub-group, let us use $idx_l$ to represent the indices of the values that the $l^{th}$ sub-group contains in the $j^{th}$ attribute. For example, suppose an attribute contains 10 values. After transformation, we find two linear sub-groups. The first sub-group contains the first four values and the second sub-group contains the left over six values. Then the indices for the first group is $idx_1 = \{1, 2, 3, 4\}$ and the indices for the second group is $idx_2 = \{5, 6, 7, 8, 9, 10\}$.

The original values of the $l^{th}$ linear sub-group in the $j^{th}$ attribute can be represented by $D_{(j, idx_l)}$. The correlation matrix for this sub-group can
be calculated by

\[
C^l = \text{Cov}(D_{(i, idx_l)}, D_{(j, idx_l)}), 1 \leq i, j \leq d.
\]

We have similar expressions to Equation 4.3 and Equation 4.4 for the \(l^{th}\) linear sub-group and they are:

\[
S^l_1 = P_j^T \times C^l_{(i,j)} \times (C^l_{(i,j)})^T \times P_j
\]

\[
S^l_2 = \text{std}^2(D_{(j, idx_l)}) \times P_j^T \times C^l \times P_j
\]

The POT approach generates a vector by repeatedly solving Inequality 4.5 until the correlation is less than the given threshold. Similar to Inequality 4.5 for the whole attribute, we could have an inequality for the \(l^{th}\) linear sub-group as follows:

\[
2S^l_1 A_{(i,j)} \vec{v} + \vec{v}^T S^l_1 \vec{v} < 2\rho_j^2 S^l_2 A_{(i,j)} \vec{v} + \rho_j^2 \vec{v}^T S^l_2 \vec{v}
\]

We have the above inequality for each linear sub-group. Therefore, we will have \(K\) inequalities for all the \(K\) linear sub-groups. The question is how many inequalities and which inequalities we need to solve.

For a domain containing \(n\) values and a sub-group in this domain containing \(n_l\) values, if a hacker could crack \(c\%\) of those \(n_l\) values in this sub-group, then this sub-group contributes \(\frac{c\%}{n} \times n\) to the total cracks for the whole attribute. For example, given a domain with 1000 data values and a linear sub-group containing 10 values, if all the 10 values in this sub-group have been cracked (i.e., \(c\% = 100\%\)), its contribution to the crack percentage for the whole attribute is \(\frac{10}{1000} \times 100\% = 1\%\). We only need to select the sub-groups that have significant contribution to the attribute crack, which is determined by the crack percentage \(c\%\) and the size of a linear sub-group \(n_l\). Therefore, we could select sub-groups by two predefined thresholds in the following way:

- **Correlation Threshold**: the crack percentage \(c\%\) is positive correlated to the correlation between the original values and the transformed values. Therefore, we only care about the linear sub-groups that have higher correlation than the given threshold between the original values and the transformed values. We call those linear sub-groups *vulnerable sub-groups*.

- **Size Threshold**: the size of a vulnerable sub-group is defined as the number of values it contains. If the size of a vulnerable sub-group is too
4.4. Optimizing POT : POT for Linear Sub-groups

small (i.e., \( n_i \) is small), it might be not necessary to take extra effort to break down the correlation for this sub-group. We only need to pick up the sub-groups that have greater size than a given threshold\(^{10}\).

Among those \( K \) linear sub-groups, suppose there are \( k \) vulnerable sub-groups that have greater sizes than a given threshold, then we have \( k + 1 \) inequalities in total, and they are:

\[
\begin{align*}
2S_1 A_{(\cdot,j)} \bar{v} + \bar{v}^T S_1 \bar{v} &< 2\rho_j^2 S_2 A_{(\cdot,j)} \bar{v} + \rho_j^2 \bar{v}^T S_2 \bar{v} \\
2S_1^1 A_{(\cdot,j)} \bar{v} + \bar{v}^T S_1^1 \bar{v} &< 2\rho_j^2 S_2^1 A_{(\cdot,j)} \bar{v} + \rho_j^2 \bar{v}^T S_2^1 \bar{v} \\
\vdots &\vdots \vdots \\
2S_1^k A_{(\cdot,j)} \bar{v} + \bar{v}^T S_1^k \bar{v} &< 2\rho_j^2 S_2^k A_{(\cdot,j)} \bar{v} + \rho_j^2 \bar{v}^T S_2^k \bar{v}
\end{align*}
\]

(4.9)

The first inequality is for the whole attribute and the other \( k \) inequalities are for those \( k \) vulnerable sub-groups. By solving this set of inequalities, we could get a solution \( \bar{v} \) and a new vector \( \tilde{A}_{(\cdot,j)} \) for attribute \( j \) = \( A_{(\cdot,j)} + \bar{v} \), which yields a lower correlation.

Algorithm POT-LS

We summarize the above procedure in Algorithm POT-LS shown in Figure 4.2. The variable \( cnt \) counts how many times the algorithm has checked vulnerable sub-groups and is set to be zero at the beginning of the algorithm. The vector \( q \) is generated by the POT approach for attribute \( j \). Attribute \( j \) is transformed by \( q^T \times D \) in Step 1. In Step 2, we use the GAP statistics and the Lga method to find linear sub-groups. Step 3 checks whether there exist sub-groups that have greater size than the size threshold \( S \) and higher correlation than the correlation threshold \( \rho_0 \). If such vulnerable sub-group exists, the algorithm will generate a new vector \( q \) for attribute \( j \) from Step 3.0 to Step 3.4.

In Step 3.0, the counter \( cnt \) is increased by 1. In Step 3.1, we select the top \( k \) vulnerable sub-groups by the descending order of correlation and generate Inequality Set 4.9. We introduce the input parameter \( k \), the number of selected vulnerable sub-groups, to evaluate the efficiency of the algorithm. For example, if there are three vulnerable sub-groups satisfy the selection

\(^{10}\)Data owners or custodians can also choose to select all vulnerable sub-groups, regardless of size. However, in this case, they need to pay the additional effort required to solve the inequalities for all sub-groups.
4.4. Optimizing POT : POT for Linear Sub-groups

Algorithm 4.1: POT-LS

Input: $D$: d-dimensional training data set
$q$: a vector generated by the POT method
$\rho_0$: the correlation coefficient threshold
$S$: the size threshold
$k$: the number linear sub-groups to be broke down
$L$: the total number of loops

0 cnt = 0;
1 Get $D_{(j,:)}' = q^T \times D$
2 Use the GAP statistics and the Lga method to find linear sub-groups
3 If there exist groups such that
   1) sizes are greater than $S$ and
   2) correlations are higher than $\rho_0$
   3.0 { cnt = cnt + 1;
   3.1 Select the top $k$ vulnerable subgroups and generate the inequality set 4.9
   3.2 do
      { Get a $\vec{v}$ by solving Inequality Set 4.9;
        Set $A_{(j,:)} = A_{(j,:)} + \vec{v}$;
        Compute $\rho_j$ and $\rho_{lj}, 1 \leq l \leq k$ by Equation 4.2;
      } until $\rho_j \leq \rho_0$ and $\rho_{lj} \leq \rho_0, 1 \leq l \leq k$
   3.3 Compute $q$ by Equation 4.1;
   3.4 If cnt $\leq L$ Goto Step 1
}

Figure 4.2: Algorithm POT-LS : POT for Linear Sub-Groups

condition, i.e., sizes are greater than $S$ and correlations are higher than $\rho_0$, the range of $k$ is from 1 to 3. If $k = 1$, we only select the vulnerable sub-group with the highest correlation. In experiments, we will evaluate the efficiency and the protection level for different values of $k$.

Step 3.2 is a loop to generate a new vector $A_{(j,:)}$. First, we get a vector $\vec{v}$ by solving Inequality Set 4.9 and update the vector $A_{(j,:)}$. The correlation is computed by Equation 4.2. Step 3.2 will repeat until both the correlation for the whole attribute ($\rho_j$) and the correlations for the selected $k$ vulnerable subgroups ($\rho_{lj}, 1 \leq l \leq k$) are less than the threshold $\rho_0$. The new vector $q$ is generated in Step 3.3.
Step 3.4 checks how many times the algorithm has run. \( L \) is an input parameter to define the total number of checks. Each time when we generate a new vector \( q \) in Step 3.2, we could only guarantee that the correlations of the selected \( k \) vulnerable sub-groups have been reduced to be lower than the given threshold \( \rho_0 \). However, there is no guarantee that the new vector \( q \) will not yield other vulnerable sub-groups. Therefore, Step 3.4 of Algorithm 4.1 will go back to Step 1 to check vulnerable sub-groups again. The algorithm will stop when there is no vulnerable sub-group or the algorithm has run \( L \) times.

Let us study an example to show how the algorithm works. We set the correlation threshold to be 0.6 and the size threshold to be 30% of the domain size. Figure 4.3 shows the relationship between the original values and the corresponding transformed values for different transformations. Figure 4.3(a) is the same as Figure 4.1(b), which transforms the original values with an orthogonal matrix generated by POT. The correlation for the whole domain is 0.6, but there are two highly correlated linear sub-groups with correlations as 0.92 and −0.91 respectively. Figure 4.3(b) shows the results for POT-LS when we set the parameter \( k \) to be 2. After generating a new matrix, the algorithm goes back to Step 1 and identifies two new linear sub-groups with correlations as −0.71 and 0.62 respectively. Note that the two groups in Figure 4.3(b) and the two groups Figure 4.3(a) contain different original values and transformed values. Even though the PO-LS breaks down the correlations for the two sub-groups shown in Figure 4.3(a) to be lower than the correlation threshold, POT-LS produces another two new linear sub-groups. The conditions in Step 3 are still true and POT-LS will generate another orthogonal matrix. Figure 4.3(c) shows the results for the new generated transformation. The correlation for the whole attribute is 0.15. There is no more linear sub-group identified and the algorithm will stop.

**Beyond Lga**

In Algorithm POT-LS, we use the GAP statistics to determine the number of linear sub-groups and use the Lga method to assign values into these sub-groups. In Step 3 of Algorithm POT-LS, we check the correlation and the size of each sub-group. If all sub-groups have lower correlation than the given threshold, the algorithm stops. However, even though a sub-group has lower correlation than the threshold \( \rho_0 \), we might be able to find a subset of values in this sub-group that still has higher correlation than \( \rho_0 \). Algorithm 4.2 (Lga-Posterior) in Figure 4.4 shows how to find a vulnerable subset in a
4.4. Optimizing POT : POT for Linear Sub-groups

Figure 4.3: Examples of Transformations : Algorithm POT-LS

linear sub-group.

G is a linear sub-group found by the Lga method in Algorithm POT-LS (Figure 4.2). Same to Algorithm POT-LS, $\rho_0$ is the correlation threshold and $S$ is the size threshold. A new variable $H$ is set to equal to $G$ in Step 1.

Step 2 is a while loop and the conditions are whether the size of $H$ is greater than $S$ and the correlation is lower than $\rho_0$. Step 2 repeatedly deletes the point that is farthest from the linear fitting line. The line is formed in Step 2.1 and the farthest point $p$ is located in Step 2.2. We update the variable $H$ by deleting point $p$ in Step 2.3. The last step checks whether the correlation between the original values and the transformed values for $H$ is higher than $\rho_0$. If $H$ has higher correlation than $\rho_0$, the algorithm returns
4.4. Optimizing POT: POT for Linear Sub-groups

**Algorithm 4.2: Lga-Posterior**

**Input:**
- $G$: a sub-group found by Lga
- $\rho_0$: correlation coefficient threshold
- $S$: the size threshold

**Output:**
- $H$: vulnerable sub-group

1. $H = G$;
2. while size of $H$ is greater than $S$ and the correlation is lower than $\rho_0$
   2.1. { Fit a line for $H$ by linear regression;
   2.2. find the farthest point $p$ from the fitting line;
   2.3. $H = H - p$
   }
3. If correlation $\geq \rho_0$ return $H$
   Else return Empty

Figure 4.4: Lga-Posterior : Identifying More Linear Sub-Groups

$H$. Otherwise, the algorithm returns Empty.

Figure 4.5 shows an example for Lga-Posterior. After transformed by an orthogonal matrix, we show the original values and the transformed values in Figure 4.5(a). The correlation for the whole domain is 0.07 and no linear sub-group is identified by Lga. We then use Lga-Posterior on the data. The algorithm iteratively removes the points that are far away from the

![Figure 4.5: Finding Linear Sub-Groups : Lga-Posterior](image-url)
4.4. Optimizing POT: POT for Linear Sub-groups

regression line and finally identifies a sub-group of values that has correlation as 0.85. The identified sub-group is marked by circles and locates between the two dashed lines in Figure 4.5(b).

The algorithm Lga-Posterior can be inserted after Step 2 of Algorithm POT-LS when Lga has formed sub-groups. Lga-Posterior helps us to further find more vulnerable sub-groups even though the sub-groups found by Lga have lower correlations than the correlation threshold. In the experiments, we will evaluate the effectiveness of Algorithm Lga-Posterior.

4.4.3 Properties of Algorithm POT-LS

The first question is whether there is always a solution for Inequality Set 4.9. In other words, whether Step 3.2 could find a solution to make both $\rho_j$ and $\rho^2_j, 1 \leq l \leq k$ to be less than the threshold $\rho_0$.

Since we select the top $k$ vulnerable sub-groups, the total number of inequalities is $k + 1$ in Inequality Set 4.9. For attribute $j$, there is always a solution for Inequality Set 4.9 if $j + (k + 1) \leq d$. We have the following theorem for Step 3.2 of Algorithm POT-LS.

**Theorem 4.1.** Given a threshold $\rho_0$, for attributes $j : 1, ..., d - 2$, if the number of selected sub-groups $k$ satisfies $j + (k + 1) \leq d$, Step 3.2 of Algorithm POT-LS can always find a vector $A_{(j,:)}$ such that both the correlation for the whole attribute and the correlation for each sub-group are less than the threshold $\rho_0$, i.e., $\rho^2_j \leq \rho_0^2$ and $(\rho^2_l)_j \leq \rho_0^2, 1 \leq l \leq k$.

**Proof:** Firstly, Step 3.2 of Algorithm POT-LS monotonically reduces $\rho^2_j$ and $(\rho^2_l)_j$. The new vector $\vec{v}$ comes from a solution of Inequality Set 4.9, which ensures that the new vector $A_{(j,:)}$ can produce smaller correlations.

Secondly, we need to show that there is always a solution $\vec{v}$ for Inequality Set 4.9 if $\rho^2_j > \rho_0^2$ or $\exists l, 1 \leq l \leq k, (\rho^2_l)_j > \rho_0^2$. Inequality Set 4.9 is not solvable only when $\rho^2_j$ and all $(\rho^2_l)_j$ have reached the minimum values, which are actually 0. The reason is as follows:

The correlation between $D_{(j,:)}$ and $D'_{(j,:)}$ is

$$\rho_j = \frac{\text{cov}(D_{(j,:)}, D'_{(j,:)})}{\text{std}(D_{(j,:)}) \text{std}(D'_{(j,:)})}$$

and the covariance

$$\text{cov}(D_{(j,:)}, D'_{(j,:)}) = \text{cov}(D_{(j,:)}, (Q_{(j,:)}))^{T} \times D$$

$$= (Q_{(j,:)}))^{T} \times C_{(j,:)}$$
where $C_{(::,j)}$ is the $j$th column of covariance matrix $C$ and is a non-zero vector.

Similarly, for each sub-group, we have

$$\rho^l_j = \frac{\text{cov}(D_{(j,:), (j,l)_{idx}}), D'_{(j,:), (j,l)_{idx}})}{\text{std}(D_{(j,:), (j,l)_{idx}}) \times \text{std}(D'_{(j,:), (j,l)_{idx}})}$$

where $C^l_{(:,j)}$ is the $j$th column of covariance matrix $C^l$ and is a non-zero vector.

In $d$ dimensional space, because $j + (k + 1) \leq d$, there must exist a non-zero vector $q$ that is orthogonal to $C_{(:,j)}$, $C^l_{(:,j)}$, $1 \leq l \leq k$ and $Q_{(:,i)}$, $1 \leq i < j$, which means $q^T \times C_{(:,j)} = 0$ and $q^T \times C^l_{(:,j)} = 0$. Let $Q_{(:,j)} = q$, then $\text{cov}(D_{(j,:), (j,:), (j,:)}) = 0$. The denominator of $\rho_j$ is $\text{std}(D_{(j,:), (j,:), (j,:)})$, which is not 0 because $Q_{(:,i)}$ is a non-zero vector. Therefore, $\rho^2_j$ equals 0. For the same reason, $(\rho^l_j)^2$ equals 0.

In conclusion, Step 3.2 of Algorithm POT-LS can always iteratively find a new vector $A_{(:,j)}$ to monotonically reduce $\rho^2_j$ and $(\rho^l_j)^2$ till they reach the minimum value 0.

In Algorithm POT-LS, we use a counter ‘cnt’ to count how many times the algorithm have checked vulnerable sub-groups. Theorem 4.1 can guarantee that the correlation for the whole attribute and the correlations for the top $k$ linear sub-groups are less than the threshold. However, the new generated orthogonal vector $q$ might introduce other vulnerable sub-groups. The algorithm stops after checking vulnerable sub-groups for $L$ times. The value of $L$ is an input parameter. However, experiments show that the algorithm will not repeat too many times (>10) to reach a solution.

### 4.5 POT for Other Classification Methods

The POT approach is proposed for the SVM classification. There are five other widely used linear classification methods, i.e., linear regression, ridge regression, lasso regression, Elastic Net, and logistic regression [9]. In this section, we extend the POT approach to these linear classification methods.

In particular, we examine whether the NOC guarantee is provided by POT for the linear models.

Given a data set $D$ with $n$ data points, each point $x_i$ is associated with a class label $y_i$. The goal of the linear model is to find the optimal parameters
4.5. POT for Other Classification Methods

(b, w) such that the linear regression function \( f(x, w, b) = b + x^T w \) minimizes the error function. We have the following definition for the NOC guarantee provided by POT for linear classification models.

**Definition 4.4.** Given a data set \( D \), an orthogonal matrix \( Q \), and the transformed data set \( D' = Q \times D \), let \((b_0, w_0)\) be the optimal parameters of a linear model for \( D \). We say that the orthogonal transformation provides the NOC guarantee for this linear model if and only if the parameters \((b_0, Q \times w_0)\) are also the optimal parameters for the transformed data set \( D' \).

Based on this definition, we examine each linear classification model as follows:

- **Linear regression:**
  The error function to be minimized is \( J_n(w, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w, b))^2 \), which aims to minimize the errors between the class labels and the predicted class labels. We have the following theorem for linear regression:

**Theorem 4.2.** The POT approach provides the NOC guarantee for linear regression.

Proof: Let us use \((b_0, w_0)\) to represent the optimal parameters of the linear regression, which means the value \( J_n(w_0, b_0) \) is the minimal error for \( D \) and we have:

\[
J_n(w_0, b_0) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (b_0 + x_i^T w_0))
\]

For the transformed data set \( D' = Q \times D \), given a point \( x_i \) in \( D \), the corresponding transformed point is in the form of \( x'_i = Q \times x_i \). The parameters \((b_0, w'_0) = (b_0, Q \times w_0)\) yields the error for \( D' \):

\[
J_n(w'_0, b_0) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (b_0 + (Q \times x_i)^T \times Q \times w_0))
\]

\[
= \frac{1}{n} \sum_{i=1}^{n} (y_i - (b_0 + x_i^T Q \times Q \times w_0))
\]

\[
= \frac{1}{n} \sum_{i=1}^{n} (y_i - (b_0 + x_i^T w_0)) = J_n(w_0, b_0)
\]
4.5. POT for Other Classification Methods

If \((b_0, w'_0)\) are not the optimal parameters, then there exists a pair \((b', w'_1)\) such that \(J_n(w'_1, b') < J_n(w'_0, b)\) for \(D'\). The error for \((b', w'_1)\) is as follows:

\[
J_n(w'_1, b') = \frac{1}{n} \sum_{i=1}^{n} (y_i - (b' + (Q \times x_i)^T \times w'_1))^2
\]

Then for data set \(D\), we can have a pair of parameters \((b', w_1) = (b', Q^T \times w'_1)\) such that the error of classification is:

\[
J_n(w_1, b') = \frac{1}{n} \sum_{i=1}^{n} (y_i - (b' + x_i^T \times w_1))^2 = J_n(w'_1, b')
\]

Then we will have \(J_n(w_1, b') < J_n(w_0, b)\) for dataset \(D\), which is contradictory with the assumption that \((b_0, w_0)\) are the optimal parameters for \(D\). Therefore, \((b_0, Q \times w_0)\) are the optimal parameters for the transformed data set \(D'\).

For the same reason, if \((b_0, Q \times w_0)\) are the optimal parameters for the transformed data set \(D'\), then \((b_0, w_0)\) will be the optimal parameters for the data set \(D\) too.

Thus, the POT approach provides the NOC guarantee for linear regression.

The orthogonal transformation contains rotation and reflection. Geometrically, both rotation and reflection preserve the geometric shape of a data set and preserve the distance between any two points. Therefore, if a line is the optimal classifier in the original space, the corresponding line in the transformed space will be also the optimal classifier for the transformed data set, which is similar to the reason that POT preserves the SVM classification discussed in chapter three.

- Ridge regression:

The optimization function is \(\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 + \lambda \sum_{j=1}^{d} w_j^2\). Different from the linear regression, there is a penalty, \(\lambda \sum_{j=1}^{d} w_j^2\), which
4.5. POT for Other Classification Methods

equals to \( w^T w \). The geometric meaning of \( \sum_{j=1}^d w_j^2 \) is the square of the length of the vector \( w \). The orthogonal transformation will not change the length of a vector. Therefore, we could have the following corollary from Theorem 4.2:

**Corollary 4.1.** The POT approach provides the NOC guarantee for ridge regression.

- **Lasso regression:**
  The optimization function is \( \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i, w))^2 + \lambda \sum_{j=1}^d |w_j| \). The penalty is \( \lambda \sum_{j=1}^d |w_j| \), which means the sum of the absolute projections on the axes of vector \( w \). The orthogonal transformation contains rotation and reflection. The reflection preserves the sum of axis-projections of a vector. However, the rotation will not preserve this sum generally. Therefore, the orthogonal transformation will not preserve this sum. Thus, the orthogonal transformation does not provide the NOC guarantee for lasso regression.

- **Elastic Net:**
  The optimization function is \( \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i, w))^2 + \lambda \sum_{j=1}^d ((1 - \alpha)w_j^2 + \alpha |w_j|) \). The penalty is \( \lambda \sum_{j=1}^d ((1 - \alpha)w_j^2 + \alpha |w_j|) \). If \( \alpha \) equals 0, the Elastic Net is actually ridge regression. If \( \alpha \) equals 1, the Elastic Net is lasso regression. For other values of \( \alpha \), the penalty is the combination of length of the vector \( w \) and the sum of the axis-projections of the vector \( w \). We already know that the orthogonal transformation does not preserve the sum of the axis-projection of the vector \( w \). Therefore, the orthogonal transformation does not provide the NOC guarantee for the Elastic Net.

- **Logistic regression**
  The optimization function is \( f(z) = \frac{1}{1+e^{-z}}, z = \beta_0 + x^T \times \beta \). From Theorem 4.2, we could know that the function \( z = \beta_0 + x^T \times \beta \) is preserved by the orthogonal transformation \( Q \). Therefore, \( f(z) \) is also preserved by the orthogonal transformation. We have the following corollary from Theorem 4.2:

**Corollary 4.2.** The POT approach provides the NOC guarantee for logistic regression.

We summarize the NOC guarantee for the linear classifications in Table 4.1.
4.6. Combination of POT and Random Perturbation

<table>
<thead>
<tr>
<th>Models</th>
<th>Optimization Functions</th>
<th>NOC from POT?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression</td>
<td>$\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2$</td>
<td>Yes</td>
</tr>
<tr>
<td>Ridge Regression</td>
<td>$\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 + \lambda \sum_{j=1}^{d} w_j^2$</td>
<td>Yes</td>
</tr>
<tr>
<td>Lasso Regression</td>
<td>$\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 + \lambda \sum_{j=1}^{d}</td>
<td>w_j</td>
</tr>
<tr>
<td>Elastic Net</td>
<td>$\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, w))^2 + \lambda \sum_{j=1}^{d} ((1 - \alpha)w_j^2 + \alpha</td>
<td>w_j</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>$f(z) = \frac{1}{1 + e^{-z}}$ $z = \beta_0 + x^T \times \beta$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4.1: The NOC Guarantee for Linear Classification

4.6 Combination of POT and Random Perturbation

Random perturbation has been widely studied and is a popular approach for privacy preserving classification [1, 2]. Random perturbation adds random noise to the original data in order to hide the sensitive information. The more noise added, the more protection it provides, and meanwhile the more change it produces on the classification patterns. It will be advantageous to provide an approach that could reduce the outcome change while not sacrificing the privacy. The POT approach preserves the classification patterns while providing high level protection. Therefore, we could apply POT on the perturbed data to provide extra protection. With the added extra protection, we might need less noise to be added on the original data and thus yield less change on the classification patterns. Figure 4.6 shows how to apply POT to random perturbation and shows how to compare the combined approach and random perturbation.

In the random perturbation approach, we add random noise $N$ to the original data set $D$ and generate a new data set $D' = D + N$. Then the classification is applied on the perturbed data $D'$ and the pattern $P_1$ is produced.

In the combination of POT and random perturbation, we apply an orthogonal transformation $Q$ on the perturbed data $D'$ and create a new data set $D'' = Q \times D' = Q \times (D + N)$. The classification is applied on the data.

\[\text{Note that Figure 4.6 is a little different from Figure 1.3. The reason is that Figure 1.3 shows the framework for SVM only and the pattern } P_1 \text{ and the pattern } P_2 \text{ are identical for the SVM classification}\]
4.7 Experiments

In this section, we show experimental results about the POT-LS approach and the effectiveness of the combination of POT and random perturbation. In the experiments, we use various data sets from the UCI collection [22], including Boston Housing, Credit and WDBC (i.e., Wisconsin Breast Cancer (Diagnostic)) and the IBM synthetic data set used in paper [2].

We deal with numerical attributes for all data sets. We show the descriptions of the data sets in Table 4.2.

We evaluate the POT-LS approach and the combination of POT and random perturbation with the curve fitting attack, which is defined in Definition 4.3. The radius $\delta$ (Definition 4.1) of the knowledge point is set to...
4.7. Experiments

We applied different curve fitting attack models, e.g., linear fitting, spline fitting and polyline fitting. Experiments show that all of them produce similar results. In this section, we mainly show the results for linear fitting.

We implemented the experiments in MATLAB environment and ran the experiments on an Intel Pentium PC with 3GHz CPU and 2GB RAM.

### 4.7.1 Optimizing POT: POT-LS

The first part of the experiments is to evaluate the proposed heuristic (i.e., Algorithm POT-LS) that breaks down the correlations among the linear sub-groups and enhances the protection in the worst cases. As discussed in Section 4.4.1, the worst case means that there exist at least one linear sub-group and all of a hacker’s knowledge points falling within this linear sub-group. In the experiments, we set the number of the knowledge points to be four.

In the experiments, the correlation threshold ($\rho_0$) is set to be 0.6. The size threshold ($S$) varies from 10% to 50% of the domain size and all of them show similar results. In this section, we only show the results for $S = 30\%$. Another input parameter to Algorithm POT-LS, $k$, is the number of selected vulnerable sub-groups. In the experiments, in order to evaluate the effectiveness and the efficiency of the algorithm, we vary $k$ from 1 to the total number of vulnerable sub-groups.

Figure 4.7 shows the experimental results for the WDBC data set. We show the results for five attributes, e.g., attributes #1,#3,#4,#8, and #9. No linear sub-group is identified for the other five attributes, even when the size threshold is set to be 10%.

For each figure in Figure 4.7, the x-axis is the parameter $k$, the number of linear sub-groups to be handled. The primary y-axis shows the crack percentage and the second y-axis shows the runtime in seconds.

There are three lines in each figure. The solid line with the triangle marker shows the sub-group improvements with respect to the level of protection. The crack percentage for a sub-group is defined as the fraction of

---

<table>
<thead>
<tr>
<th></th>
<th>WDBC</th>
<th>Housing</th>
<th>Credit</th>
<th>IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Numeric Attrs</td>
<td>10</td>
<td>10</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td># of Points</td>
<td>569</td>
<td>509</td>
<td>690</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 4.2: Data Sets

be 2% of the 1% trimmed range of each domain, i.e., the range without the 0.5% biggest values and the 0.5% smallest values.
4.7. Experiments

![Figure 4.7: POT-LS (in the Worst Cases) : Data Set WDBC](image)

(a) Attr #1  
(b) Attr #3  
(c) Attr #4  
(d) Attr #8  
(e) Attr #9

Figure 4.7: POT-LS (in the Worst Cases) : Data Set WDBC
4.7. Experiments

the number of cracks in this group over the number of values in this group, which is similar to the domain disclosure risk. The dashed line with the dot marker shows the overall attribute improvements. The dotted line with the diamond marker shows the runtime.

Figure 4.7(a,b,c) shows the results for attributes #1,#3,and #4. We can see that the crack percentages for these three attributes could be around 50% for the whole attributes in the worst cases if the data is transformed by POT. The crack percentages for the linear sub-groups (the solid lines) are even higher.

When $k$ is set to be 1, the POT-LS algorithm breaks down the correlation for the most correlated sub-group. We could see from the figure that the crack percentage drops quickly for both the linear sub-groups and the overall attributes. For example, Figure 4.7(c) shows that the sub-group improvement is about 32% (the crack percentage drops from 60% to 28%), which contributes to about 25% overall improvements for the whole attribute. Furthermore, if $k$ is 2, the crack percentage will be about 20%.

For attributes #8 and #9, Figure 4.7(d,e) show that the crack percentage is not as high as attributes #1,#3,#4, but the crack percentage also drops while we introduce $k$ to break down the correlations among the linear sub-groups. In Figure 4.7(d,e), only the results of $k = 1$ are shown because the algorithm only finds one linear sub-group for attribute #8 and attribute #9 respectively.

The runtime is the time needed to solve the inequalities. The POT approach only solves one inequality for the whole attribute. The POT-LS approach solves a set of inequalities, i.e., Inequality Set 4.9. The runtime only shows the extra time needed to solve this set of inequalities. The experimental results show that the total runtime to solve the set of inequalities is less than 0.1 seconds even though the runtime has increased a lot by comparing with the POT approach.

Figure 4.8 shows the experimental results for the Housing data set. This data set also contains ten numerical attributes. POT-LS found two attributes that contains linear sub-groups after applying the orthogonal transformation.

Figure 4.8(b) shows the results for attribute #5. From the figure we can know that the crack percentage in the worst cases is more than 40% in the linear sub-groups and about 35% for the whole attribute. POT-LS reduces the crack percentage to below 20%. When $k = 2$, the crack percentage will

\footnote{We do not show the runtime for the GAP approach and the Lga method. Different linear sub-group methods might have different runtime properties.}
4.7. Experiments

Figure 4.8: POT-LS (in the Worst Cases) : Data Set Housing

be about 10%. Figure 4.8(a) shows the results for attribute #4.

In Figure 4.8(b), when \( k = 2 \), the crack percentage for the linear sub-group is lower than the whole attribute, which is different from Figure 4.7 and Figure 4.8(a). This is because the cracked values counted in the crack percentage for the whole attribute contain other values which are not in the identified linear sub-groups.

Figure 4.9: POT-LS (in the Worst Cases) : Data Set Credit

Figure 4.9 shows the results for the Credit data set. Credit contains four numerical attributes. Attributes #2 and #3 contains linear sub-groups after applying the orthogonal transformation. Figure 4.10 shows the results for the IBM synthetic data set. All of them show similar results to the WDBC
and the Housing data sets.

![Figure 4.10: POT-LS (in the Worst Cases) : IBM Synthetic DataSet](image)

**Evaluation of Lga-Posterior**

So far we have evaluated the algorithm POT-LS and we use the GAP statistics and the Lga method to identify linear sub-groups. If the linear sub-groups found by Lga have lower correlations than the threshold, we propose to use Algorithm 4.2 (Lga-Posterior) to find more vulnerable sub-groups. In this part, we show experimental results about Algorithm Lga-Posterior in Table 4.3.

We compare the Lga method and the heuristic Lga-Posterior in three aspects, i.e., the number of identified linear sub-groups, the sub-group improvements on the crack percentage, and runtime.

For the WDBC data set, we can see Lga-Posterior finds two linear sub-groups and yields 20% sub-group improvements for Attribute #8. In contrast, if we only use the Lga method, only one linear sub-group is found with 12% improvements. Lga-Posterior also finds one vulnerable sub-group for Attribute #7 with 10% sub-group improvements.

For the Housing data set, we can see Lga-Posterior could also find more vulnerable sub-groups and yield more improvements on the crack percentage. Lga-Posterior finds one more linear sub-groups for both attribute #6 and #9. For attribute #10, Lga-Posterior finds two linear sub-groups with 20% improvements on the crack percentage.

Table 4.3 also shows the results for Attribute #3 of the Credit data set and the results for Attribute #5 of the IBM synthetic data. In both cases,
Lga-Posterior could find more vulnerable sub-groups and yield lower crack percentage. Experimental results show that Lga-Posterior could provide more protection in the worst cases.

The last two columns in Table 4.3 show runtimes for Lga and Lga-Posterior. We can see Lga-Posterior needs less than 0.1 seconds to identify more linear sub-groups and yields more improvements on the crack percentage even though Lga-Posterior needs more time to run.

### 4.7.2 Combination Of POT and Random Perturbation

In this section, we evaluate the effectiveness of the combination of POT and random perturbation with regard to the classification accuracy and the protection level.

#### 4.7.2.1 Classification Model : SVM

Figure 4.11- 4.14 show the experimental results of the SVM classification for datasets WDBC, Housing, Credit, and the IBM synthetic data respectively. In each figure, (a) shows the crack percentage and (b) shows the classification accuracy and the disagreements of the support vectors with respect to the corresponding standard deviation.

Figure 4.11 shows the results for WDBC. In Figure 4.11(a), the x-axis shows the standard deviation of the added noise. The noise is randomly drawn from a normal distribution with standard deviation as a given percentage (c%) of the domain range. The experiments vary the standard deviation from 1% to 50% of the domain range. The y-axis shows the crack percentage. The solid line with diamond marker shows the crack percentage for random perturbation. The crack percentage is more than 20% when the standard deviation of the noise is 5% of the domain range. Especially, when
4.7. Experiments

c% is less than 2%, the crack percentage could be about 50%. Only when
c% is more than 20%, the crack percentage drops to about 10%. The dashed
line with the triangle marker shows the crack percentage for the combina-
tion of POT and perturbation. From Figure 4.11(a) we can see that the
dashed line is much flatter than the solid line. Even for c% = 1%, the crack
percentage is as low as 10%.

![Diagram](image-url)

(a) Crack Percentage

(b) Classification Accuracy & SV Disagreements

Figure 4.11: The Combination Approach for SVM: WDBC

Figure 4.11(b) shows the classification accuracy and the support vector
disagreements. As shown in chapter three, since POT preserves the support
vectors and thus preserves the classification patterns, the patterns derived
from the POT transformed data and the patterns mined from the perturbed
data are identical, i.e., $P_1 \equiv P_2$ in Figure 4.6. Therefore, in the experiments,
we only need to measure the discrepancy of the classification patterns mined
before and after the random perturbation.

The solid line shows the classification accuracy. The first value is the
classification accuracy for the original data set without adding noise. The
figure shows that the classification accuracy drops while increasing the stan-
dard deviation of the added noise.

Besides the classification accuracy, the other way is to directly compare
the patterns mined before and after perturbation. For the SVM classi-
cation, it is to compare the found support vectors before and after the
perturbation. Let us use $V_o$ to represent the set of support vectors in
the original data set and $V_p$ to represent the set of support vectors in the
perturbed data set. The disagreement between $V_o$ and $V_p$ is defined as
$disag(V_o, V_p) = 1 - \frac{V_o \cap V_p}{V_o \cup V_p}$. In Figure 4.11(b), the dashed line shows that
4.7. Experiments

disagreements of the support vectors consistently increase while adding more noise.

Finally, let us look at the Figure 4.11(a) and (b) together. Figure 4.11(a) shows that only when the standard deviation of the noise is more than 20% of the domain range, the random perturbation will have the same level of protection (i.e., the crack percentage is less than 10%) as the combination approach. Meanwhile, Figure 4.11(b) shows that the classification accuracy drops a lot from the original classification models and the support vector disagreements are more than 60% when the standard deviation of the noise is more than 20%. In the random perturbation approach, if we would like to derive a classification model with less discrepancy from the original model, we might want to limit the standard deviation of the added noise to be less than 10% or even less than 5%, which provide poor protection (i.e., high crack percentage). However, the combination of POT and random perturbation could significantly improve the protection on the data even when the noise is small. Figure 4.11(a) shows that the crack percentage is about 10% when the noise is 1% and further drops to about 7% when the noise is 5%.

Figure 4.12 shows the results for the Housing data set. Figure 4.13 shows the results for the IBM synthetic data set. Figure 4.14 shows the results for the Credit data set. All of them show similar results to Figure 4.11. We can derive more accurate classification models for SVM while not sacrificing the privacy protection by applying POT on random perturbation.

![Figure 4.12: The Combination Approach for SVM: Housing](image)

(a) Crack Percentage  
(b) Classification Accuracy & SV Disagreements
4.7. Experiments

Figure 4.13: The Combination Approach for SVM: IBM Synthetic Data

Figure 4.14: The Combination Approach for SVM: Credit

4.7.2.2 Linear, Ridge, and Logistic Regression

The WDBC Data Set

Figure 4.15 shows experimental results for linear models, i.e., linear regression, logistic regression and ridge regression, for the WDBC data set. Results for lasso and Elastic Net are shown in Figure 4.17 and Figure 4.18.

Figure 4.15(a) shows the crack percentage, which is the same as Figure 4.11(a); Figure 4.15(b,c,d) show the classification measurement for different classification models. For each figure, the solid line with the triangle marker shows the classification accuracy. The first value shows the classification accuracy on the original data set without adding noise.
4.7. Experiments

Figure 4.15: The Combination Approach for Linear Models: WDBC

The dashed line with the diamond marker shows the angles between the classifier (i.e., the optimal hyperplane) mined from the original data set and the classifier mined from the data set transformed by the combination of POT and random perturbation. Similar to the SVM classification (Figure 4.11), for all these three linear models, the classification accuracy drops quickly and the angles between the two classifiers increase fast while adding more noise (i.e., increasing the standard deviation of the added noise). With POT applied on the perturbed data, we need less noise to achieve high level of protection while not sacrificing classification accuracy.

The Housing Data Set

Figure 4.16 shows the experimental results for the Housing data set. The results show similar trends to Figure 4.15. Experiments for the Credit and the IBM synthetic data sets also show similar results. All the experiments confirm that the combination of POT and random perturbation can produce more accuracy classifiers while not sacrificing privacy by adding less noise.
4.7. Experiments

Figure 4.16: The Combination Approach for Linear Models: Housing

4.7.2.3 Lasso and Elastic Net

The WDBC Data Set

We show experimental results for lasso and Elastic Net separately in Figure 4.17 and Figure 4.18. We do not repeat to show the crack percentage in this part since it is the same as Figure 4.15(a).

In Figure 4.17 and Figure 4.18, (a) shows the classification measurement for random perturbation. Similar to Figure 4.15, the solid line with the triangle marker shows the classification accuracy and the dashed line with the diamond marker shows the angles between the two classifiers. Figure(b) shows the classification measurement for the combination of POT and random perturbation. Figure(c) compares the classification accuracy and Figure(d) compares the angles between the two classifiers for the two approaches. The black line with the cross marker shows the results for random perturbation and the gray line with the dot marker shows the results.
for the combination of POT and random perturbation.

![Figure 4.17: The Combination Approach for Lasso: WDBC](image)

From Figure 4.17(a) and (b), we can see the accuracy drops and the angles increases while adding more noise. Because POT does not preserve the lasso classification, the combination approach yields a little less accurate classifier than the random perturbation in Figure 4.17(c). Similarly, the combination approach also introduce wider angles between the two classifiers than the random perturbation approach in Figure 4.17(d) for most cases. However, the difference of the classification accuracy between the random perturbation and the combination approach is less than 2%. The introduction of POT could significantly improve the protection level (Figure 4.15(a)), we only need a small amount of noise while achieving enough protection with high classification accuracy. For example, for the combination of POT and perturbation, the crack percentage is about 7% (Figure 4.15(a)) and the classification accuracy is about 79% when the standard deviation of the
added noise is 5% of the domain range. In contrast, for random perturbation alone, in order to limit the crack percentage to be around 7%, the standard deviation of the noise needs to about 30% and the classification accuracy is round 70%.

Figure 4.18 shows the results for Elastic Net. The penalty part for

Figure 4.18: The Combination Approach for Elastic Net : WDBC

Elastic Net \( \sum_{j=1}^{d}(1 - \alpha)w_j^2 + \alpha|w_j| \) combines the penalty parts for the ridge regression and the lasso regression. In the experiments, we set \( \alpha \) to be 0.5. We can find similar trends for Elastic Net to lasso regression. For both Elastic Net and lasso, the combination approach provides high level of protection on the private information by adding less noise.

**The Housing Data Set**

For the housing Data Set, Figure 4.19 shows the experimental results of lasso and Figure 4.20 shows the results of Elastic Net.

Both of them show similar trends to the WDBC data set (Figure 4.17.
4.8 Conclusions

In this chapter, we extend the POT approach for other linear classifications, propose a transformation approach by combining POT and random perturbation, and provide a heuristic to enhance the level of privacy protection of POT in the worst cases.

We use POT for privacy preserving linear classifications, including linear regression, ridge regression, lasso regression, Elastic Net, and logistic regres-
4.8. Conclusions

Figure 4.20: The Combination Approach for Linear Models: Housing

We show that POT provides the NOC guarantee for linear, ridge and logistic classification models by preserving the classification error functions.

In order to take advantage of the POT approach in privacy persevering, i.e., providing high level protection and preserving classification patterns, we propose a transformation approach that combines POT and random perturbation. After the original data has been perturbed by adding random noise, we further use POT to generate an orthogonal matrix to transform the perturbed data. Experimental results confirm that the combination approach needs less noise to reach high level of privacy protection while minimizing the outcome change.

In order to provide more protection on privacy, we investigate the privacy protection properties of the POT approach in the worst cases. We find that there might exist linear sub-groups that still have high correlations between the original values and the corresponding transformed values in an attribute.
4.8. Conclusions

after the orthogonal transformation. Furthermore, those high correlated linear sub-groups are vulnerable to curve fitting attacks and undermine the protection ability of the POT approach. We propose a heuristic to be applied on the POT approach to break down the correlations for vulnerable sub-groups. We present empirical results to show that the proposed heuristic could generate a more robust orthogonal transformation to significantly improve the protection on the private information.
4.9 Bibliography


4.9. Bibliography


Chapter 5

Conclusions and Future Works

5.1 Conclusions

In this thesis, we study the problem of privacy preservation in the data-mining-as-a-service model [8]. Before the data is submitted to a service provider, transformation approaches should be applied on the original data to protect the private information while minimizing the change on the mining outcome [1, 2, 11, 12, 13]. We focus our study in the context of classification and propose transformation approaches to counter attack models with prior knowledge [3, 6]. The contributions of this thesis can be concluded as follows.

- We propose a piecewise transformation approach for privacy preservation in outsourcing of the decision tree classification.

  We show that a substantially rich class of functions – (anti-)monotone functions – can provide the no-outcome-change guarantee. Since it might not be safe enough to transform a whole domain with only one (anti-)monotone function, we further propose a piecewise transformation based on two methods to enhance the protection. The first method is to break a domain into pieces and apply different (anti-)monotone functions on different pieces. The second method is to identify the monochromatic pieces in which all values are associated with the same class label. We show that we could use an arbitrary function to transform a monochromatic piece. We prove that the proposed piecewise transformation still provides the NOC guarantee if the functions applied on pieces satisfy the global (anti-)monotone invariant.

  We empirically evaluate the effectiveness of the proposed framework using an extensive battery of tests and provided a comprehensive analysis of the experimental results. Our results show that with the introduction of breakpoints and monochromatic pieces, the piecewise framework is effective in reducing the disclosure risk.
5.1. Conclusions

- We propose two approaches, i.e., the principled orthogonal transformation (POT) and the true negative points (TNP) perturbation, for privacy preservation in outsourcing of the SVM classification.

We first propose to use the POT approach to generate an orthogonal matrix to transform the original data. We prove that the POT approach provides the NOC guarantee for both linear and non-linear SVM and show that the POT approach could break down the correlation between the original values and the corresponding transformed values. We present empirical results to show that POT could effectively counter curve fitting attacks with prior knowledge and significantly reduce the crack percentage by reducing the correlation. Also, the smaller the correlation, the smaller the crack percentage.

Another attack model with prior knowledge is the global attack, which aims to crack the encrypted values by making a guess on the transformation matrix. The global attack is similar to the matrix attack defined in [4, 5, 9]. We prove that the crack ability of the global attacks is independent of the transformation matrix. In order to counter global attacks, we propose the TNP approach, which finds and perturbs true negative points. We show that the TNP approach provides the NOC guarantee for linear SVM on linearly separable data set. Experimental results show that the TNP approach could effectively counter global attacks while minimizing the outcome change.

- In the last part of this thesis, we extend the POT approach to linear classification models and show that POT also provides the NOC guarantee for linear models including linear regression, ridge regression and logistic regression. We propose a combination approach that applies POT on random perturbation in order to minimize the outcome change without sacrificing privacy. In order to further improve the protection on privacy in the worst cases, we propose a heuristic (Algorithm POT-LS) to break down the correlations between the original values and the corresponding transformed values of subsets.

We conduct a comprehensive set of experiments to evaluate both the proposed heuristic POT-LS and the combination approach. The experimental results show that the proposed heuristic, i.e., POT-LS, could substantially reduce the crack percentage and thus enhance the protection on the original data. Moreover, the results also show that the combination of POT and random perturbation could reach high level of privacy protection by adding less noise, which means the outcome
5.2 Future Works

Besides the problems studied and the approaches proposed in this thesis, there are several open problems in privacy preservation in outsourcing of data mining, and we would like to further investigate the following problems in future:

- Optimizing the POT approach:
  We propose the heuristic POT-LS to further improve the privacy protection by breaking down the correlations between the original values and the corresponding transformed values of subsets. However, as we have discussed in chapter four, POT-LS cannot guarantee that the new generated vector will not produce new highly correlated subsets of values. Although experimental results show that the proposed heuristic is effective to reduce the crack percentage, it will be helpful to design an approach that could generate an orthogonal matrix that will not produce any highly correlated subsets of values.

- Beyond linear SVM on linear separable data sets:
  The TNP approach proposed in this thesis is to counter the global attack. However, the TNP approach only provides the NOC guarantee for linear SVM on linearly separable data sets. For linearly non-separable data sets, there is no guarantee on preserving the mining outcome. Meanwhile, it is still an open problem to design a transformation approach against global attacks with the NOC guarantee for SVM with non-linear kernels.

- Privacy preservation in clouds:
  More and more computing tasks will be conducted in cloud computing environment [7, 10]. Even though the proposed approaches could be used in cloud computing, it might be necessary to design specific transformation schema for submitting data to clouds. For example, a data owner might want to use clouds to both store data and run data mining tasks. If the data is updated frequently, the piecewise transformation framework and the TNP approach might not work. It is interesting to propose new transformations for data updating while providing both privacy and pattern preservation in cloud computing.
5.3 Bibliography


128