Protecting Output Privacy in Stream Mining

Abstract

Privacy preservation in data mining demands protecting both input and output privacy. The former refers to sanitizing the raw data itself before performing mining. The latter refers to preventing the mining output (model/pattern) from malicious pattern-based inference attacks. The preservation of input privacy does not necessarily lead to that of output privacy.

This work studies the problem of protecting output privacy in the context of frequent pattern mining on data streams. After exposing the privacy breaches existing in current stream mining systems, we propose Butterfly*, a light-weighted countermeasure that can effectively eliminate these breaches without explicitly detecting them, meanwhile minimizing the loss of the output accuracy. We further optimize the basic scheme by taking account of two types of semantic constraints, aiming at maximally preserving utility-related semantics while maintaining the hard privacy and accuracy guarantee. We conduct extensive experiments over real-life datasets to show the effectiveness and efficiency of our approach.*

1 Introduction

Recent years have witnessed increasing concerns about individual privacy in numerous data mining and management applications. Individuals were usually unwilling to provide their personal information if they knew that the privacy of the data could be compromised. A plethora of work has been done on preserving the *input privacy* for static data [3, 7, 11, 16, 17], which assumes untrusted mining service providers and enforces privacy regulations by sanitizing the raw data before sending it to the service providers. The mining algorithms are performed over the sanitized data. This scenario is shown as the first four steps of Fig. 1.

However, surprisingly limited attention has been given to preserving *output privacy* in data mining: the published mining output can be leveraged to infer properties possessed only by a unique or a small number of individuals, even though the models/patterns may be built over sanitized data. This can be explained by the fact that input-privacy preserving techniques are designed to make the constructed models/patterns as close as possible to, if not identical to that built over raw data, in order to guarantee the utility of the result. This no-outcome-change property is considered as a pillar of privacy preserving data mining [5]. As long as the significant statistical information of the raw data is preserved, there exists the risk of disclosure of private information. Therefore, the preservation of input privacy does not necessarily lead to that of output privacy.

Example 1. Consider a nursing-care records database that records the observed symptoms of the patients in a hospital. By mining such a database, one can discover valuable information about syndromes characterizing particular diseases. However, the released mining output can also be leveraged to uncover some combinations of symptoms that are so special that only rare people match them (we will show how to achieve this in the following sections), which qualifies as severe threats to individuals' privacy.

Assume that Alice knows that Bob has certain symptoms a, b but not c ($\overline{c}$), and by analyzing the mining output, she finds that only one person in the hospital matches the specific combination of $\{a, b, \overline{c}\}$, and only one has all $\{a, b, \overline{c}, d\}$. She can then conclude that the one is Bob, who also has the symptom d. Further more, by studying other medical database, she may learn that the combination of $\{a, b, d\}$ is linked to a rare disease with high chance.

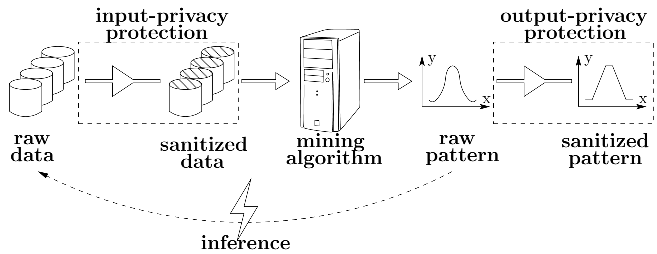

Fig. 1: Illustration of privacy protection in data mining applications.

This output privacy issue is even severer in stream mining: (i) The mining results need to be published in a continuous and in-time manner. Not only a single-time release may contain privacy breaches, but also multiple releases can potentially be exploited in combination, given the overlap of the corresponding input data. Taking the sliding window model as an example, in addition to the leakage in the output of a single window (*intra-window breach*), the output of multiple overlapping window can be combined to infer sensitive information (*inter-window breach*), even though each window itself contains no privacy breach per se. (ii) The techniques developed for protecting privacy of static data are mostly based on global analysis of the data. However the characteristics of stream usually evolve over time, making these approaches not suitable, due to the limited processing time and memory in stream systems. Therefore, one needs to consider protecting output privacy in stream mining as a unique problem.

A straightforward solution to preserving output privacy is to detect and eliminate all potential breaches, i.e., the *detecting-then-removing* strategy adopted in inference control of statistical databases and census data from 1970's. However, the results are usually negative in tone [8] for on-line stream mining systems: First, the detection of breaches usually requires complicated computations over entire dataset and the bookkeeping of voluminous history output; Second, even at such high cost, the operations of removing the found breaches, e.g., suppression, addition [2], usually result in significant decrease of the utility of the output.

In this work, we study the problem of protecting output privacy in the context of frequent pattern mining over streams. Concretely, analogous to sanitizing raw data from leaking sensitive information, we propose the concept of "sanitized pattern", and argue that by intelligently modifying the "raw pattern" outputted by mining algorithms, one can significantly reduce the risk of malicious inferences, while maximally preserving the utility of the raw patterns. This scenario is shown as the last step in Fig. 1.

Specifically, we present Butterfly, a light-weighted countermeasure against malicious inferences based on value perturbation. It possesses the following desirable features: (i) No need of explicit detection of privacy breaches; (ii) No need of bookkeeping history output; (iii) Flexible control over the balance of multiple utility metrics and privacy guarantee.

Our Contributions (i) We articulate the problem of protecting output privacy in stream mining, and expose the privacy breaches existing in current stream mining systems; (ii) We propose a generic framework of protecting output privacy: On the first tier, it counters malicious inferences by amplifying the uncertainty of sensitive information; On the second tier, for the given privacy requirement, it maximally preserves output utility; (iii) We provide both theoretical analysis and experimental evaluation to validate our approach in terms of privacy guarantee, output utility and algorithm efficiency.

2 Related Work

In this section, We outline the related work along three most relevant areas.

Disclosure Control in Statistical Database Extensive research has been done in statistical databases to provide statistical information without compromising sensitive information regarding individuals [4, 18], by the techniques of query restriction or data perturbation. Compared with simple statistical information, e.g., min, max, avg, etc, the mining output (model/pattern) is usually more complex, leading to more complicated requirement for output utility, which makes it hard to directly apply these techniques in our scenario.

Input Privacy Preservation The work of [1, 3] paved the way for the rapidly expanding field of privacy preserving data mining. While a plethora of techniques have been developed, including data perturbation [1, 3, 7, 11], k-anonymity [16, 17] and secure multi-party computation [14, 21], these techniques focus on protecting input privacy for static datasets, where the design goal is to provide sufficient privacy guarantee while minimizing the information loss in the mining output. A recent work [15] also addresses the problem of preserving input privacy for streaming data, by on-line analysis of correlation structure of multivariate streams.

Output Privacy Preservation Compared with the wealth of techniques developed for preserving input privacy, protecting mining output privacy has not received the attention it deserves. The work [13] proposes an empirical testing scheme for evaluating if the constructed classifier violates the privacy constraint. It is shown in [2] that the association rules can be exploited to infer information about individual transactions. The work [22] proposes a scheme to block the inference of sensitive patterns satisfying user-specified templates by suppressing certain raw transactions. A recent work [5] tries to consider input and output privacy in a unified framework, however it is not clear that it can prevent malicious inference over the mining output. To the best of our knowledge, none has addressed the problem of protecting output privacy in stream mining applications.

3 Problem Definition

In this section, we introduce the concept of frequent pattern mining over data streams, and the output privacy issue arising in this context.

3.1 Frequent Pattern Mining

Consider a finite set of items $\mathcal{I} = \{i_1, i_2, \ldots, i_M\}$. An itemset I is a subset of $\mathcal{I}$, i.e., $I \subseteq \mathcal{I}$. A database $\mathcal{D}$ consists of a set of records, each corresponding to a non-empty itemset. The support of an itemset I w.r.t. $\mathcal{D}$, denoted by $T_{\mathcal{D}}(I)$, is the number of records in $\mathcal{D}$, which contain I as a subset. Frequent pattern mining aims at finding all itemsets with support exceeding a user-defined threshold C, called the minimum support.

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}	r_{11}	r_{12}	$\ldots$
a	a	a	a	a	a	a		a	a			
b		b	b		b	b	b	b		b		$\ldots$
c	c		c	c	c	c	c	c	c	c	c	
d		d	d				d	d		d	d	

$\mathcal{D}_s(12, 8)$

$\mathcal{D}_s(11, 8)$

Fig. 2: Illustration of data stream and sliding window model.

We model a data stream $\mathcal{D}_s$ as a sequence of records, $\{r_1, \ldots, r_N\}$, where N is the current size of the stream, and grows as time goes by. The sliding window model is introduced to deal with the potential of N going to infinity. Concretely, at each N, one considers only the window of most recent H records, $r_{N-H+1}, \ldots, r_N$, denoted by $\mathcal{D}_s(N, H)$, where H is the size of the window. The problem of frequent stream pattern mining is to find in each $\mathcal{D}_s(N, H)$, all itemsets with support w.r.t. $\mathcal{D}_s(N, H)$ exceeding the minimum support C.

One can further generalize the concept of itemset by introducing the negation of an item: Let $\overline{i}$ denote the negation of the item i. A record is said to contain $\overline{i}$ if it does not contain i. Following, we will use the term *pattern* to denote a set of items or negation of items, e.g., $a\overline{bc}$. We use the notation $\overline{I}$ to represent the negation of an itemset I, i.e., $\overline{I} = \{\overline{i} | i \in I\}$.

Analogously, one can define the support of a pattern p w.r.t. a database $\mathcal{D}$: We say a record satisfies p if it contains all the items and negations of items in p. The support of p w.r.t. $\mathcal{D}$ is the number of records in $\mathcal{D}$ that satisfy p.

Example 2. Consider a data stream with current size N = 12, window size H = 8, shown in Fig. 2, where $a \sim h$ and $r_1 \sim r_{12}$ represent the set of items and records respectively. The pattern $\overline{a}bc$ has support 2 w.r.t. $\mathcal{D}_s(12, 8)$, because only records r_8 and r_{11} match it.

3.2 Mining Output Privacy

The output privacy refers to the requirement that the output (model/pattern) of a mining process does not disclose any sensitive information regarding a unique individual record or a very small number of records.

Examples of such sensitive information in the context of frequent pattern mining are usually in the form of patterns with low support. Recall Example 1 of nursing-care records in Section 1, clearly the disclosure of such patterns through output inference can lead to uncovering sensitive information regarding few individuals.

Therefore we introduce the concept of *vulnerable pattern*. Intuitively, vulnerable patterns are those with very low support w.r.t. the given database, i.e., only few individual records match them. To quantitatively measure this, we introduce a threshold K ($K \ll C$), called the vulnerable support. We have the following classification of patterns based on their support values.

Definition 1. *Pattern Classification. Given a database $\mathcal{D}$, let $\mathcal{P}$ be the set of patterns appearing in $\mathcal{D}$, then all $p \in \mathcal{P}$ can be classified into three disjoint classes:*

$$\begin{cases} \textit{Frequent Pattern}: & \mathcal{P}_f = \{p | T_{\mathcal{D}}(p) \geq C\} \\ \textit{Hard Vulnerable Pattern}: & \mathcal{P}_{hv} = \{p | 0 < T_{\mathcal{D}}(p) \leq K\} \\ \textit{Soft Vulnerable Pattern}: & \mathcal{P}_{sv} = \{p | K < T_{\mathcal{D}}(p) < C\} \end{cases}$$

The frequent patterns ($\mathcal{P}_f$) expose the significant statistics of the underlying data, and are often the candidate in the mining process. Actually the frequent itemsets found by the mining process are a subset of $\mathcal{P}_f$. The hard vulnerable patterns ($\mathcal{P}_{hv}$) represent the properties possessed by only a very small number of individuals, so it is unacceptable that they are disclosed or inferred from the mining output. The soft vulnerable patterns ($\mathcal{P}_{sv}$) neither demonstrate statistical significance, nor violate the privacy of individual records. Therefore $\mathcal{P}_{sv}$ is not contained in mining output, but it is usually tolerable that they are learned from the output.

The problem of protecting output privacy in stream frequent pattern mining can therefore be stated as follows:

For each sliding window $\mathcal{D}_s(N, H)$, output privacy protection prevents the disclosure or inference of any hard vulnerable patterns w.r.t. $\mathcal{D}_s(N, H)$ from the mining output.

Although the output of frequent pattern mining contains only those patterns with their support exceeding C ($C \gg K$), as we will show in next section, an adversary may still be able to infer certain patterns with support below K from the released frequent itemsets and their associated support.

4. Attack Model

For ease of presentation, we will use the following notations: for two itemsets I and J, IJ denotes their union, $J \backslash I$ the difference of J and I, and $|I|$ the size of I.

4.1 Attack Techniques

Lattice Structure As a special case of multi-attribute aggregation, computing the support of $I \subset J$ can be considered as generalization over all the attributes of $J \backslash I$. Therefore one can apply the standard work of computing multi-attribute aggregation, a lattice structure. Without ambiguity, we use the notation $\mathcal{X}_I^J = \{X | I \subseteq X \subseteq J\}$ to represent both the set of itemsets and their corresponding lattice structure. An example of lattice $\mathcal{X}_c^{abc}$ is shown in Fig. 3.

Deriving Pattern Support Consider two itemsets $I \subset J$, if the support of the lattice nodes of $\mathcal{X}_I^J$ is available, one is able to derive the support of the pattern p of the form, $p = I(\overline{J \setminus I})$, according to the inclusion-exclusion principle:

$$T_{\mathcal{D}}(I(\overline{J \setminus I})) = \sum_{X \in \mathcal{X}_I^J} (-1)^{|X \setminus I|} T_{\mathcal{D}}(X)$$

Example 3. As illustrated in Fig. 3, given all the support of $\mathcal{X}_c^{abc}$ w.r.t. $\mathcal{D}_s(12, 8)$, the support of pattern $p = \overline{ab}c$ can be derived as 1.

Estimating Itemset Support Since the support of any pattern is non-negative, according to the inclusion-exclusion principle, if the support of the itemsets $\mathcal{X}_I^J \setminus \{J\}$ is available, one is able to bound the support of J as follows:

$$\begin{cases} T_{\mathcal{D}}(J) \leq \sum_{I \subseteq X \subset J} (-1)^{|J \setminus X|+1} T_{\mathcal{D}}(X) & |J \setminus I| \text{ odd} \\ T_{\mathcal{D}}(J) \geq \sum_{I \subseteq X \subset J} (-1)^{|J \setminus X|+1} T_{\mathcal{D}}(X) & |J \setminus I| \text{ even} \end{cases}$$

Example 4. In Fig. 3, given the support of c, ac and bc w.r.t. $\mathcal{D}_s(12, 8)$, one is able to establish the lower/upper bound for $T_{\mathcal{D}_s(12,8)}(abc)$ as [2,5].

When the bound is tight, i.e., the lower bound equals to the upper bound, one can exactly determine the actual support. This technique is used in [19] to mine non-derivable frequent itemsets. In our context, an adversary can leverage this technique to exploit the privacy breaches existing in mining output.

4.2 Intra-Window Inference Attack

In a stream mining system without output privacy protection, the released frequent itemsets over one specific window may contain the intra-window breaches, which can be exploited by an adversary through the technique of deriving pattern support, as shown in Example 3.

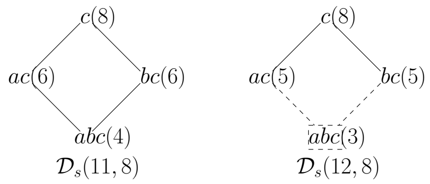

Fig. 3: Illustration of intra-window and inter-window breaches.

Formally, if J is a frequent itemset, then according to the Apriori rule, all $X \subseteq J$ are frequent, which are supposed to be reported with their support, so the information is available to compute the support of pattern $p = I(\overline{J \setminus I})$ for all $I \subset J$. This also implies that the number of possible breaches to be checked is potentially exponential in terms of the number of items.

Even if the information of J is unavailable, i.e., the lattice of $\mathcal{X}_I^J$ is incomplete to infer $p = I(\overline{J \setminus I})$, one could possibly apply the technique of estimating itemset support first to complete some missing "mosaics", then derive vulnerable pattern support. In fact, the itemsets under estimation themselves could be vulnerable.

4.3 Inter-Window Inference Attack

In stream mining, the output of the previous window can be leveraged to infer the vulnerable patterns within the current window, and vice versa, even though no vulnerable patterns can be inferred from the output of each window per se.

Example 5. Consider the two windows $\mathcal{D}_s(11, 8)$ and $\mathcal{D}_s(12, 8)$ shown in Fig. 3. Assuming C = 4, and K = 1, then in $\mathcal{D}_s(11, 8)$, no $\mathcal{P}_{hv}$ exists. In $\mathcal{D}_s(12, 8)$, the itemset abc is unaccessible (shown in dashed box). From the available information of $\mathcal{D}_s(12, 8)$, the best guess about abc is $[2, 5]$, as discussed in Example 4. Clearly, this bound is not tight enough to estimate that the pattern $\overline{ab}c$ is $\mathcal{P}_{hv}$. Thus both windows are currently immune to intra-window inference attack.

However, if one is able to derive that the support of abc decreases by 1 between $\mathcal{D}_s(11, 8)$ and $\mathcal{D}_s(12, 8)$, then based on the information released in $\mathcal{D}_s(11, 8)$, which is $T_{\mathcal{D}_s(11,8)}(abc)$ = 4, the exact value of abc in $\mathcal{D}_s(12, 8)$ can be inferred, and the $\mathcal{P}_{hv}$ $\overline{ab}c$ is uncovered.

The main idea of inter-window inference is to exactly estimate the transition of the support of certain itemsets from the previous window to the current one. We below discuss how to achieve accurate estimation of such transition over two consecutive windows.

Without loss of generality, consider two overlapping windows $W_p = \mathcal{D}_s(N - L, H)$ and $W_c = \mathcal{D}_s(N, H)$ ($L < H$), i.e., W_c is lagging W_p by L records. Assume the adversary is trying to derive the support of pattern $p = I(\overline{J \setminus I})$ in W_c. Let $\mathcal{X}_p$ and $\mathcal{X}_c$ be the subsets of $\mathcal{X}_I^J$ that are released or estimated from the output of W_p and W_c respectively. We assume that $\mathcal{X}_p \cup \mathcal{X}_c = \mathcal{X}_I^J$, i.e., $\mathcal{X}_I^J \setminus \mathcal{X}_c$ = $\mathcal{X}_p \setminus \mathcal{X}_c$, which means the missing part in $\mathcal{X}_c$ is released or can be estimated exactly in $\mathcal{X}_p$. In Fig. 3, $\mathcal{X}_p$ = $\{c, ac, bc, abc\}$, while $\mathcal{X}_c$ = $\{c, ac, bc\}$.

For an itemset X, let Δ_X^+ be the number of records containing X in the window $\mathcal{D}_s(N, L)$, and Δ_X^- the number of records containing X in the window $\mathcal{D}_s(N - H, L)$. Thus the change of the support of X over W_p and W_c can be modeled as inserting Δ_X^+ records and deleting Δ_X^- ones.

Example 6. In our example, with $N = 12$, $H = 8$ and $L = 1$, we have $\mathcal{D}_s(N, L)$ corresponds to the record r_{11} and $\mathcal{D}_s(N - H, L)$ refers to the record r_4. Clearly r_4 contains ac, while r_{11} does not. So $T_{\mathcal{D}_s(12,8)}(ac) = T_{\mathcal{D}_s(11,8)}(ac)$ + Δ_{ac}^+ - Δ_{ac}^- = 5.

One is interested in estimating $T_{W_c}(X^*)$ for $X^* \in \mathcal{X}_p \setminus \mathcal{X}_c$. The bound (min, max) of $T_{W_c}(X^*)$ can be obtained by solving the following integer programming problem:

$$\max(\min) \quad T_{W_p}(X^*) + \Delta^+_{X^*} - \Delta^-_{X^*}$$

satisfying the constraints:

$$\begin{array}{ll} R1: 0 \le \Delta^+_X, \Delta^-_X \le L & \\ R2: \Delta^+_X - \Delta^-_X = T_{W_c}(X) - T_{W_p}(X) & X \in \mathcal{X}_p \cap \mathcal{X}_c \\ R3: \Delta^+_X(\Delta^-_X) \le \sum_{I \subseteq Y \subset X} (-1)^{|X \setminus Y|+1} \Delta^+_Y(\Delta^-_Y) & |X \setminus I| \text{ odd} \\ R4: \Delta^+_X(\Delta^-_X) \ge \sum_{I \subseteq Y \subset X} (-1)^{|X \setminus Y|+1} \Delta^+_Y(\Delta^-_Y) & |X \setminus I| \text{ even} \end{array}$$

R1 comes from the fact that W_p differs from W_c by L records. When transiting from W_p to W_c, the records containing X that are deleted or added could not exceed L. R2 is amount to saying that the support change ($\Delta^+_X - \Delta^-_X$) for those itemsets $X \in \mathcal{X}_c \cap \mathcal{X}_p$ is known. R3 and R4 are the application of estimating itemset support for itemsets in the windows $\mathcal{D}_s(N, L)$ and $\mathcal{D}_s(N-H, L)$.

Intuitively, the inference process runs as follows: Starting from the change of $X \in \mathcal{X}_n \cap \mathcal{X}_p$ (R2), by using rules R1, R3 and R4, one tries to estimate $\Delta^+_X(\Delta^-_X)$ for $X \in \mathcal{X}_p \setminus \mathcal{X}_n$. It is noted that when the interval between W_p and W_c, L is small enough, the estimation can be fairly tight.

Example 7. In our running example, one can first observe the following facts based on R1 and R2:

$$\begin{array}{lll} \Delta^+_{ac} - \Delta^-_{ac} = -1, 0 \le \Delta^+_{ac}, \Delta^-_{ac} \le 1 & \Rightarrow & \Delta^+_{ac} = 0, \Delta^-_{ac} = 1 \\ \Delta^+_{bc} - \Delta^-_{bc} = -1, 0 \le \Delta^+_{bc}, \Delta^-_{bc} \le 1 & \Rightarrow & \Delta^+_{bc} = 0, \Delta^-_{bc} = 1 \end{array}$$

Take ac as an instance, its change over W_p and W_c is Δ^+_{ac} - Δ^-_{ac} = -1, and both Δ^+_{ac} and Δ^-_{ac} are bounded by 0 and 1, therefore the only possibility is Δ^+_{ac} = 0 and Δ^-_{ac} = 1.

By applying R3 and R4, one has the following facts:

$$\begin{array}{lll} \Delta^+_{abc} \le \Delta^+_{ac} = 0 & \Rightarrow & \Delta^+_{abc} = 0 \\ \Delta^-_{abc} \ge \Delta^-_{ac} + \Delta^-_{bc} - \Delta^-_{c} = 1 & \Rightarrow & \Delta^-_{abc} = 1 \end{array}$$

Take abc as an instance, one knows that Δ^+_{abc} should be no greater than Δ^+_{ac} = 0, therefore Δ^+_{abc} = 0. Meanwhile Δ^-_{abc} has an equal upper and lower bound as 1. Thus the estimation of abc over W_c, $T_{W_c}(abc)$ = $T_{W_p}(abc)$ + Δ^+_{abc} - Δ^-_{abc} = 4 + 0 - 1 = 3, and the $\mathcal{P}_{hv}$ $\overline{abc}$ is uncovered.

The computation cost of inter-window inferences is dominated by the cost of solving the constrained integer optimization problems. There exist several off-the-shelf fast techniques, which make this inference attack feasible with moderate computing power.

4.4 Design Consideration of Solutions

A naïve approach to blocking intra-/inter-window inferences is a two-staged scheme: In the first stage, one detects all intra-window breaches and eliminates them using the techniques developed for static datasets, e.g., adding fake records, or suppressing vulnerable records [2], etc; In the second stage, by referring to released output of previous windows, one further blocks the inter-window inferences. However, such a detecting-then-removing scheme is haunted by two main problems: (i) The computation cost is impractical for high-rate streaming data applications, as it requires not only detecting all potential breaches, but also keeping track of the history output; (ii) The technique of modifying raw data twists the output, resulting in significant decrease in the utility of the mining output.

Alternative to this reactive approach, we propose to use a proactive approach to deal with these two types of attacks in a uniform way. Our approach is motivated by two key observations: (i) In many mining applications, users do not expect the exact support of frequent itemsets. Rather they care more about the semantic relationships in terms of their support, e.g., the ranking or the ratio of their support values. Thus it is tolerable to trade some precision of the support of frequent itemsets for increasing the output privacy guarantee, provided that such desired output utility is maintained; (ii) Both intra- and inter-window inferences are based on the inclusion-exclusion principle, which involves multiple frequent itemsets. If we introduce some trivial uncertainty into each frequent itemset, the resulting inferred pattern can have considerable uncertainty, due to the accumulative property of uncertainty.

Based on these two observations, we propose Butterfly, a light-weighted output privacy preservation scheme based on random perturbation.

5 Output Privacy Protection

Following we first present some background of random perturbation, then show its feasibility in preserving output privacy in stream mining by a sound theoretical proof.

5.1 Mining Output Perturbation

Data perturbation refers to the process of modifying confidential data while preserving its utility for intended applications [4]. This is arguably the most important technique used so far for protecting original input data.

In our scheme however, we employ perturbation to inject uncertainty into the mining output. The perturbation over output pattern is significantly different from that over input data. In input perturbation, the data utility is defined on the overall statistical characteristics of the dataset. The distorted data is fed as input into the following mining algorithms. There is usually no utility constraints for individual data value. While in output perturbation, the perturbed results are directly presented to the end-user, and the data utility is defined over each individual value.

Specifically, there are two types of utility constraints for the perturbed results: (i) Each reported value should have enough accuracy, i.e., the perturbed value should not deviate from the actual value too far; (ii) The semantic relationships among the results should be preserved to the maximum extent, e.g., the order or the ratio of the support values of frequent itemsets. There is non-trivial tradeoff among these utility metrics. To our best knowledge however, no previous work has considered such multiple tradeoff in mining output perturbation.

Concretely, we consider the following two perturbation methods, which have their roots at statistics literatures [9]: (i) *Value distortion* perturbs the support by adding a random value drawn from some distribution, e.g., a uniform distribution over the interval $[l, u]$; (ii) *Value bucketization* partitions the range of support value into a set of disjoint, mutually exclusive intervals. Instead of reporting the support of an frequent itemset, one returns the interval which the support belongs to.

Both these techniques can be applied to perturbing the mining output. However, value bucketization leads to poor output utility compared with value distortion, since all support values in an interval are modified to the same value, and semantic constraints, e.g., order or ratio, can not be enforced in this model. We therefore, focus on the technique of value distortion. Moreover, in order to guarantee the precision of the output frequent itemset, we are more interested in probabilistic distribution with bounded interval. Therefore, in the rest of the paper, we instantiate this with a discrete uniform distribution over integers, though our discussion is general enough to be applicable for other distributions.

5.2 Basic Butterfly Approach

On releasing the mining output of a stream window, one perturbs the support of each frequent itemset X, $T(X)$[1] by adding a random variable r_X drawn from a discrete uniform distribution over integers within an interval $[l_X, u_X]$. The sanitized support value $T'(X)$ = $T(X)$ + r_X is therefore a random

[1]In the presentation below, without ambiguity, we will omit the referred database $\mathcal{D}$ in the notations.

variable, which can be specified by its bias $\beta(X)$ and variance $\sigma^2(X)$. Intuitively, the bias indicates the difference of the expected value $E[T'(X)]$ and the actual value $T(X)$, and the variance represents the average deviation of $T'(X)$ from $E[T'(X)]$.

While this operation is simple, the setting of $\beta(X)$ and $\sigma^2(X)$ is non-trivial, in order to achieve both sufficient privacy protection and utility guarantee, which is the focus of our following discussion. At this moment, just note that compared with $T(X)$, r_X is non-significant, i.e., $|r_X| \ll T(X)$.

Given the basic characteristics of the perturbation, we further define the metrics to measure the precision for output frequent itemset, and the privacy guarantee for vulnerable patterns.

5.2.1 Precision Measure

Under the perturbation, the precision loss of each frequent itemset X can be measured by the *mean square error* (mse) of the perturbed support $T'(X)$: $\mathsf{mse}(X) = E[(T'(X) - T(X))^2] = \sigma^2(X) + \beta^2(X)$.

Intuitively, $\mathsf{mse}(X)$ indicates the average deviation of the perturbed support $T'(X)$ w.r.t. the actual value $T(X)$. A smaller mse implies higher precision of the frequent itemset. Also it is clear that the precision loss should depend on the actual support. A mse of 5 for frequent itmeset I with $T(I)$ = 100 may indicate sufficient accuracy, while the same mse for itemset J with $T(J)$ = 5 may render the output of little value. Therefore, we have the following precision metric:

Definition 2. *Precision Degradation (*pred*): for each frequent itemset X, its precision degradation* $\mathsf{pred}(X)$ *is defined as the relative mean squared error of $T'(X)$:*

$$\mathsf{pred}(X) = \frac{\sigma^2(X) + \beta^2(X)}{T^2(X)}$$

5.2.2 Privacy Measure

Distorting the original support of frequent itemsets is only a part of the story, it is necessary to ensure that the distortion could not be filtered out. Therefore one needs to consider the power of the adversary in estimating the support of vulnerable patterns through the protection.

Without loss of generality, suppose that the adversary desires to estimate the support of pattern p of the form $I(\overline{J \setminus I})$, and has full access to the sanitized support $T'(X)$ of all $X \in \mathcal{X}_I^J$. The privacy protection should be measured by the error of the adversary's estimation of the support of p, denoted as $T''(p)$. Let us discuss this estimation from an adversary's perspective. Along the discussion, we will show how various prior knowledge the adversary possesses may impact the estimation.

From the adversary's view, of each $X \in \mathcal{X}_I^J$, its actual support $T(X) = T'(X) - r_X$, is a variable with a discrete uniform distribution over the interval $[l'_X, u'_X]$, where $l'_X = T'(X) - u_X$, $u'_X = T'(X) - l_X$ (since $|r_X| \ll T(X)$, this is a bounded distribution over positive integers), and has variance of $\sigma^2(X)$.

Prior Knowledge 1. *The support values of different frequent itemsets are related by a set of inequalities, derived from the inclusion-exclusion principle.*

For the given frequent itemset-interval pairs, the adversary may attempt to apply these inequalities to tighten the intervals, therefore obtaining better estimation of the support. A key question she needs to answer is: for the modified interval(s), does there exist a database $\mathcal{D}$ that satisfies the constraints of these itemset-interval pairs? It is called the itemset frequency satisfiability problem (FREQSAT), which however as shown in [6] is equivalent to the probabilistic satisfiability problem (pSAT), i.e., **NP**-complete. This indicates that the inequality relationships among itemsets can hardly be leveraged to improve the estimation of a single itemset, and one can approximately consider frequent itemsets as independent in measuring the adversary's power.

The actual support of p, $T(p)$ from the adversary's view, is also a random variable. The mse of the estimation $T''(p)$ is defined as $mse(p) = E[(T(p) - T''(p))^2]$ w.r.t. $T(p)$. One has the following theorem:

Theorem 5.1. *Given the distribution $f(x)$ of a random variable x, the mean square error of an estimation e of x, $\mathsf{mse}(e) = \int_{-\infty}^{\infty}(x-e)^2 f(x)dx$ reaches its minimum value $Var[x]$, when $e = E[x]$.*

In our scenario, $\mathsf{mse}(p)$ is minimized when $T''(p) = E[T(p)]$, this is the best guess the adversary can achieve (note that the optimality is defined in the term of average estimation error, not the semantics, e.g., $E[T(p)]$ could possibly be negative). In this worst case (best case for the adversary), the $\mathsf{mse}(p)$ equals to the variance of $T(p)$, which implies the lower bound of the estimation error.

Considering the analysis regarding the prior knowledge 1, and the fact that $T(p) = \sum_{X \in \mathcal{X}_I^J} (-1)^{|X \setminus I|} T(X)$ (a linear combination), the variance of $T(p)$ can be approximated by the sum of the variance of all involved $T(X)$, therefore $\mathsf{mse}(p) = \sum_{X \in \mathcal{X}_I^J} \sigma^2(X)$. Intuitively, a larger $\mathsf{mse}(p)$ indicates a more significant error in estimating $T(p)$ by the adversary, and better privacy protection.

It is also noted that the privacy guarantee should depend on the actual value $T(p)$: if $T(p)$ is close to 0, trivial variance makes it hard for the adversary to estimate if $T(p)$ is zero or not, i.e., if such vulnerable pattern exists. Such "zero-indistinguishability" decreases as $T(p)$ grows. Therefore, we define the privacy metric for a vulnerable pattern p as the ratio of the variance of $T(p)$ from adversary's view and the square of the actual support $T(p)$.

Definition 3. *Privacy Guarantee (*prig*): For each vulnerable pattern p, its privacy guarantee* $\mathsf{prig}(p)$ *is defined as its relative estimation error:*

$$\mathsf{prig}(p) = \frac{\sum_{X \in \mathcal{X}_I^J} \sigma^2(X)}{T^2(p)}$$

Another well-known metric of privacy guarantee is entropy, which measures the uncertainty of a random variable. Both variance and entropy are important and independent measures of privacy protection. However as pointed out in [12], variance is more appropriate in measuring individual-centric privacy guarantee where the adversary is interested in determining the precise value of an random variable. We therefore argue that variance is more suitable for our scenario, since we are aiming at protecting the exact support of vulnerable patterns.

Prior Knowledge 2. *The sanitized support of the same frequent itemsets may be published in consecutive stream windows.*

Since our protection is based on independent random perturbation, if the same support value is repeatedly perturbed and published in multiple windows, the adversary can potentially improve the estimation by averaging the observed outcomes, according to the law of large numbers. To block this type of attack, once the support of a frequent itemset is perturbed, one keeps publishing this sanitized value if the actual support remains the same in consecutive windows.

Prior Knowledge 3. *The adversary may have access to other forms of prior knowledge, e.g., the published statistics of the dataset, the samples of a similar dataset, or the support of the top-k frequent itemset, etc.*

All these various forms of prior knowledge can be captured by the notion of *knowledge point*: a knowledge point is a specific frequent itemset X, for which the adversary has an average error less than $\sigma^2(X)$ in estimating $T(X)$. Our definition of privacy guarantee (prig) can readily incorporate this notion, by simply replacing the corresponding variance $\sigma^2(X)$ with the smaller estimation error.

5.2.3 Effectiveness

In summary, the effectiveness of our privacy protection method is evaluated in terms of its resilience against both intra- and inter-window inferences over stream mining output. We note three key implications.

First, the uncertainty of involved frequent itemsets are accumulated in the inferred vulnerable patterns. Moreover, more complicated inferences (i.e., harder to be detected) face higher uncertainty.

Second, the actual support of a vulnerable pattern is usually small (only a unique or less than K records match vulnerable patterns), hence adding trivial uncertainty can make it hard to tell the existence of such pattern in the dataset.

Third, the inter-window inferences follow a two-staged strategy, i.e., first deducing the transition between contingent windows, then inferring the vulnerable patterns. The uncertainty associated with both stages may result in even lower quality estimation.

5.3 Tradeoff between Precision and Privacy

In our Butterfly approach, the tradeoff between privacy protection and output utility can be flexibly adjusted by the variance and bias setting of each frequent itemset. Specifically, the variance controls the overall balance between privacy and utility, while the bias gives a finer control over the balance between precision and other utility metrics, as we will show in the next section. Here we focus on the setting of variance. Intuitively, smaller variance leads to higher precision of the output, however also decreases the uncertainty of the inferred vulnerable patterns, therefore less privacy guarantee.

To ease the discussion, we assume that all the frequent itemsets are associated with the same variance σ^2 and bias β. In the next section when semantic constraints are taken into consideration, we will remove this simplified treatment, and develop more sophisticated setting scheme.

Let C denote the minimum support for frequent itemsets. From the definition of precision measure, it can be derived that for each frequent itemset X, its precision degradation $\mathsf{pred}(X) \leq (\sigma^2 + \beta^2)/C^2$, because $T(X) \geq C$. Let $P_1(C) = (\sigma^2 + \beta^2)/C^2$, i.e., an upper bound of the precision loss of the frequent itemsets. Meanwhile for a vulnerable pattern $p = I(\overline{J \setminus I})$, it can be proved that its privacy guarantee $\mathsf{prig}(p) \geq (\sum_{X \in \mathcal{X}_I^J} \sigma^2)/K^2 \geq (2\sigma^2)/K^2$, because $T(p) \leq K$ and the inference attack involves at least two frequent itemsets. Let $P_2(C, K) = (2\sigma^2)/K^2$, i.e., a lower bound of the privacy guarantee of the inferred vulnerable patterns.

P_1 and P_2 provide a convenient representation to control the tradeoff between precision and privacy protection. Specifically, setting an upper bound ϵ over P_1 guarantees sufficient accuracy of the reported frequent itemsets; While setting a lower bound δ over P_2 provides enough privacy protection for the inferred vulnerable patterns. Therefore one can specify the requirement of the output precision and privacy guarantee as a pair of parameters (ϵ, δ), where $\epsilon, \delta > 0$. That is the setting of β and σ should satisfy $P_1(C) \leq \epsilon$ and $P_2(C, K) \geq \delta$, as

$$\sigma^2 + \beta^2 \leq \epsilon C^2 \tag{1}$$

$$\sigma^2 \geq \delta K^2/2 \tag{2}$$

To make these two inequalities compatible, it should be satisfied that $\epsilon/\delta \geq K^2/(2C^2)$. The term ϵ/δ is called *precision-privacy ratio* (ppr), which tends to indicate the precision loss for user-specified privacy requirement. When the precision is a major concern, one can set ppr as its minimum value $K^2/(2C^2)$ for given K and C, resulting in the highest precision for the given privacy requirement. The minimum ppr also implies that $\beta = 0$ and the two parameters ϵ and δ are coupled. We refer to the perturbation scheme with the minimum ppr as our basic Butterfly approach. In the following section, we will relax this condition, and take into consideration other utility metrics in addition to precision.

6 Optimized Output Utility

The basic Butterfly approach treats all the frequent itemsets uniformly (the same bias β = 0) without taking consideration of their semantic relationships. Though easy to implement and effective against inferences, this simple scheme may easily violate these semantic constraints directly related to the specific applications of the mining output. In this section, we refine the basic scheme by taking semantic constraints into our map, and develop constraint-aware Butterfly approach. Given the precision and privacy requirement (ϵ, δ), our optimized version preserves the semantics to the maximum extent.

We specifically consider two types of utility-related semantic constraints, *absolute order* and *relative frequency*. By absolute order, we refer to the ranking of frequent itemsets according to their support. In many applications, users pay considerable attention to the absolute order, e.g., querying the top-ten popular purchase patterns. By relative frequency, we mean the ratio of the support of two frequent itemsets. In certain applications, users care much about the relative frequency, e.g., computing the confidence in mining association rules.

To help model the order and ratio of the support of itemsets, we first introduce the concept of *frequency equivalence class* (FEC):

Definition 4. *Frequency Equivalence Class (FEC): A FEC is a set of frequent itemsets, having equivalent support value. Given a FEC fec, we define the support of fec, $T(fec)$ as the support value of any of its members.*

A set of frequent itemsets can be partitioned into a set of disjoint and strictly ordered FECs, based on their support. We say that two FECs, fec_i and fec_j, follow a partial order of $fec_i \prec fec_j$ if $T(fec_i) < T(fec_j)$. Without loss of generality, we assume below that the given set of FECs are sorted according to their support, i.e., $T(fec_i) < T(fec_j)$ for $i < j$.

In complying with the constraints of order or ratio, the equivalence of itemsets in a FEC should be maximally preserved in the perturbed output. Therefore in the optimized Butterfly schemes, the perturbation is performed over each of the FECs, instead of each specific itemset.

Clearly, this revision does not affect the privacy guarantee, considering the fact that the inference of a vulnerable pattern involves at least two frequent itemsets with different support, i.e., at least two FECs.

6.1 Order Preservation

When the absolute order of itemset frequency is an important concern, the perturbation over all FECs can not be uniform, since that would easily render the inversion of the orders of two FECs, especially when their support values are close. To model the probability of inversion, we first introduce the concept of *uncertainty region* of a FEC.

Definition 5. *Uncertainty Region: The uncertainty region of a FEC fec is defined as the set of values that its perturbed support $T'(fec)$ can take: $\{x | Pr[T'(fec) = x] > 0\}$.*

For instance, when adding a random variable drawn from the integers from the interval $[l, u]$ to fec, the uncertainty region is all the integers in the interval $[T(fec) + l, T(fec) + u]$.

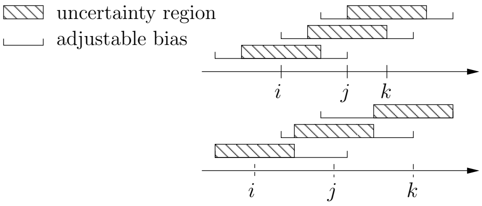

Fig. 4: Illustration of adjusting bias to minimize inversion probability.

6.1.1 Minimizing Inversion Probability

Below we formally define the problem of preserving absolute order. To simplify the notations, we use the following short version: $t_i = T(fec_i)$, $t'_i = T'(fec_i)$, and let β_i denote the bias setting for fec_i.

Without loss of generality, consider two FECs fec_i, fec_j with $t_i < t_j$. The order of fec_i and fec_j can be possibly inverted if their uncertainty regions overlap, that is $Pr[t'_i \geq t'_j] > 0$, and larger $Pr[t'_i \geq t'_j]$ indicates higher probability that this inversion occurs.

One can minimize this inversion probability $Pr[t'_i \geq t'_j]$ by adjusting β_i and β_j. However, this adjustment is bounded by the requirement of precision and privacy specified in Inequalities 1 and 2 introduced in Section 5.3. We therefore introduce the concept of *maximum adjustable bias*:

Definition 6. *Maximum Adjustable Bias: For each FEC fec_i, its bias is allowed to be adjusted within the range of $[-\beta_{mi}, \beta_{mi}]$, β_{mi} is called the maximum adjustable bias, given ϵ and δ, which is defined as $\beta_{mi} = \lfloor\sqrt{\epsilon t_i^2 - \delta K^2/2}\rfloor$, derived from Inequalities 1 and 2.*

The problem of preserving absolute order can therefore be formalized as: Given a set of FECs $\mathcal{FEC} = \{fec_1, \ldots, fec_n\}$ one finds the optimal bias setting for each FEC fec_i within its maximum adjustable bias $[-\beta_{mi}, \beta_{mi}]$, to minimize the sum of pairwise inversion probability: min $\sum_{i<j} Pr[t'_i \geq t'_j]$. This scheme is shown in Fig. 4, where after adjusting the bias setting of the three FECs, the inversion probability is minimized (zero in this case).

We now show how to compute $Pr[t'_i \geq t'_j]$ in our setting. For a discrete uniform distribution over the interval $[l, u]$, $\alpha = u - l$ is called the length of the region. The variance of this distribution is $\sigma^2 = ((\alpha+1)^2 - 1)/12$. According to Inequality 2 in Section 5, one has $\alpha = \lceil\sqrt{1 + 6\delta K^2}\rceil - 1$.

Let d_{ij} be the distance between the estimators $e_i = t_i + \beta_i$ and $e_j = t_j + \beta_j$ of fec_i and fec_j: $d_{ij} = e_j - e_i$. The intersection of the uncertainty regions of fec_i and fec_j is a piece-wise function, having four possible relationships: (i) $e_i < e_j$, fec_i and fec_j do not overlap; (ii) $e_i \leq e_j$, fec_i and fec_j intersect; (iii) $e_i > e_j$, fec_i and fec_j intersect; (iv) $e_i > e_j$, fec_i and fec_j do not overlap. Correspondingly, the inversion probability $Pr[t'_i \geq t'_j]$ is computed as follows:

$$Pr[t'_i \geq t'_j] = \begin{cases} 0 & d_{ij} \geq \alpha + 1 \\ \frac{(\alpha+1-d_{ij})^2}{2(\alpha+1)^2} & 0 < d_{ij} < \alpha + 1 \\ 1 - \frac{(\alpha+1+d_{ij})^2}{2(\alpha+1)^2} & -\alpha - 1 < d_{ij} \leq 0 \\ 1 & d_{ij} \leq -\alpha - 1 \end{cases}$$

Clearly $Pr[t'_i \geq t'_j]$ is a four-piece integer function, making this optimization problem fairly complicated. However, we can prove that it is sufficient to consider the case $d_{ij} \geq 0$ only for any pair of fec_i and fec_j when computing the inversion probability, based on the following lemma (due to the space limit, all the proofs of the lemmas are referred to our technical report):

Lemma 1. *In the solution of min $\sum_{i<j} Pr[t'_i \geq t'_j]$, any pair of FECs fec_i and fec_j with $i < j$ must have $e_i \leq e_j$, i.e., $d_{ij} \geq 0$.*

Therefore, the optimization problem can be simplified as: $\sum_{i<j}(\alpha + 1 - d_{ij})^2$. Note that the discussion so far has not considered the characteristics of each FEC, such as the number of its members. The inversion of two FECs containing five frequent itemsets each, is much more serious than that of two FECs with only one member respectively. Quantitatively, let s_i be the number of members in fec_i, the inversion of fec_i and fec_j means the ordering of $s_i + s_j$ itemsets is disturbed.

Therefore, our aim is to solve the following weighted optimization problem:

$$\begin{aligned} \min \quad & \sum_{i<j}(s_i + s_j)(\alpha + 1 - d_{ij})^2 \\ \text{s.t.} \quad & d_{ij} = \begin{cases} \alpha + 1 & e_j - e_i \geq \alpha + 1 \\ e_j - e_i & e_j - e_i < \alpha + 1 \end{cases} \\ & \forall i < j, e_i \leq e_j \quad \forall i, e_i \in \mathbb{Z}^+, |e_i - t_i| \leq \beta_{mi} \end{aligned}$$

This is a quadratic integer programming (QIP) problem, with piecewise cost function. In general, quadratic programming is **NP**-hard, even without integer constraints [20]. Instead of applying off-the-shelf quadratic optimization tools, we are more interested in on-line algorithms that can flexibly trade accuracy for efficiency. Following we present such a solution based on dynamic programming.

6.1.2 A Near Optimal Solution

Although this optimization problem in the general setting is **NP**-hard, by relaxing the constraint that $\forall i < j, e_i \leq e_j$ to $e_i < e_j$, one can construct the following optimal substructure property, leading to an efficient dynamic programming solution.

Optimal Substructure Our solution is motivated by the key observations that (i) the estimators of all the FECs are in a strict ascending order, i.e.,$\forall i < j$, $e_i < e_j$; (ii) all the FECs have uncertain regions of the same length α. Therefore each FEC can intersect with at most α previous ones. This property leads to the following lemma.

Lemma 2. *Assume that the bias of the last α FECs $\{fec_{n-\alpha+1}{:}fec_n\}^2$ are fixed as $\{\beta^*_{n-\alpha+1}{:}\beta^*_n\}$ respectively, and $\{\beta^+_1{:}\beta^+_{n-\alpha}\}$ are optimal w.r.t. $\{fec_1{:}fec_n\}$, then for given $\{\beta^+_{n-\alpha}, \beta^*_{n-\alpha+1}{:}\beta^*_{n-1}\}$, $\{\beta^+_1{:}\beta^+_{n-\alpha-1}\}$ must be optimal w.r.t. $\{fec_1{:}fec_{n-1}\}$.*

Based on this optimal sub-structure, we propose a dynamic programming solution, which adds FECs sequentially according to their order. Let $C_{n-1}(\beta_{n-\alpha}{:}\beta_{n-1})$ represent the achievable minimum cost by adjusting $\{fec_1{:}fec_{n-\alpha-1}\}$ with the last α FECs fixed as $\{\beta_{n-\alpha}{:}\beta_n\}$, and let c_{ij} denote $(s_i+s_j)(\alpha+1-d_{ij})^2$. When adding fec_n, the minimum cost $C_n(\beta_{n-\alpha+1} : \beta_n)$ is computed using the rule:

$$C_n(\beta_{n-\alpha+1} : \beta_n) = \min_{\beta_{n-\alpha}} C_{n-1}(\beta_{n-\alpha} : \beta_{n-1}) + \sum_{i=n-\alpha}^{n-1} c_{in}$$

The optimal setting is the one with the global minimum value among all the combinations of $\{\beta_{n-\alpha+1} : \beta_n\}$.

To lower the complexity, on adding each FEC, one only consider its intersection with its previous γ FECs, instead of α ones ($\gamma < \alpha$). This approximation is close to the exact solution when the distribution of FECs is not extremely dense, which is usually the case, and verified by our experiments. Formally, a (γ/α)-approximate solution is defined as:

$$C_n(\beta_{n-\gamma+1} : \beta_n) = \min_{\beta_{n-\gamma}} C_{n-1}(\beta_{n-\gamma} : \beta_{n-1}) + \sum_{i=n-\gamma}^{n-1} c_{in}$$

Computation Complexity Let β_{max} be the maximum value of maximum adjustable bias of all FECs: $\beta_{max} = \max_i \beta_{mi}$,

[2] Following without ambiguity we use $\{x_i : x_j\}$ as a short version of $\{x_i, x_{i+1}, \ldots, x_{j-1}, x_j\}$.

then the time and space complexity of this scheme are both bounded by $O(\beta_{max}^{\gamma} n)$, i.e., $O(n)$ in term of total number of FECs. In practice, the time/space complexity is usually much lower than this bound, given the facts that under the constraint $\forall i < j, e_i < e_j$, a number of combinations are invalid, and β_{max} is an over-estimation of the average maximum adjustable bias.

The whole order-preserving bias setting scheme runs as follows: one first initializes the cost function for the first γ FECs; Then by running the dynamic programming procedure, one computes the cost function for each newly added FEC. The optimal configuration is the one with the global minimum value $C_n(\beta_{n-\gamma+1} : \beta_n)$.

6.2 Ratio Preservation

In a number of applications, it is desirable to maximally preserve the relative frequency (ratio) information during the output perturbation process, given the privacy and precision requirement. Similar to the order preservation optimization, one can achieve this by carefully adjust the bias setting of FECs. First, we give a formal definition of the ratio preservation problem.

6.2.1 Maximizing (k, $1/k$) Probability of Ratio

Without loss of generality, consider two FECs fec_i and fec_j. To preserve the relative frequency of fec_i and fec_j, one is interested in making the ratio of the perturbed support t'_i/t'_j appear in the approximation area of the actual value t_i/t_j with high probability, e.g., the interval $[k\frac{t_i}{t_j}, \frac{1}{k}\frac{t_i}{t_j}]$, where $k \in (0, 1)$, indicating the tightness of this interval.

Definition 7. $(k, 1/k)$ *Probability: The* $(k, 1/k)$ *probability of the ratio of two random variables* t'_i *and* t'_j, $Pr_{(k,1/k)}\left[\frac{t'_i}{t'_j}\right]$ *is defined as* $Pr_{(k,1/k)}\left[\frac{t'_i}{t'_j}\right] = Pr\left[k\frac{t_i}{t_j} \leq \frac{t'_i}{t'_j} \leq \frac{1}{k}\frac{t_i}{t_j}\right]$, *where* $k \in (0, 1)$, *is an indicator of the tightness of the approximation area, the larger* k, *the tighter the interval.*

This $(k, 1/k)$ probability can quantitatively measure the impact of the perturbation over the ratio. Therefore the problem of ratio preservation is formalized as follows:

$$\begin{array}{ll} \max & \sum_{i<j} Pr_{(k,1/k)}\left[\frac{t'_i}{t'_j}\right] \\ \text{s.t.} & \forall i, e_i \in \mathbb{Z}^+, |e_i - t_i| \leq \beta_{mi} \end{array}$$

In the case of discrete uniform distribution, the $(k, 1/k)$ probability of the ratio of two random variables is a non-linear piecewise function, resulting in a non-linear integer optimization problem. In general sense, non-linear optimization problem is **NP**-hard, even without integer constraints. Instead of applying expensive non-linear optimization tools, we are more interested in efficient solutions that can find the near-optimal configuration with complexity linear in the term of the number of FECs. We below present such a bias setting scheme that performs well in practice.

6.2.2 A Near Optimal Solution

In order to maximize the $(k, 1/k)$ probability $Pr_{(k,1/k)}\left[\frac{t'_i}{t'_j}\right]$, one can alternatively minimize the probability $Pr\left[\frac{t'_i}{t'_j} \geq \frac{1}{k}\frac{t_i}{t_j}\right]$ + $Pr\left[\frac{t'_j}{t'_i} \geq \frac{1}{k}\frac{t_j}{t_i}\right]$. From Markov's Inequality, one knows that the probability $Pr\left[\frac{t'_i}{t'_j} \geq \frac{1}{k}\frac{t_i}{t_j}\right]$ is bounded as

$$Pr\left[\frac{t'_i}{t'_j} \geq \frac{1}{k}\frac{t_i}{t_j}\right] \leq \frac{E\left[\frac{t'_i}{t'_j}\right]}{\frac{1}{k}\frac{t_i}{t_j}} = k\frac{t_j}{t_i}E\left[\frac{t'_i}{t'_j}\right]$$

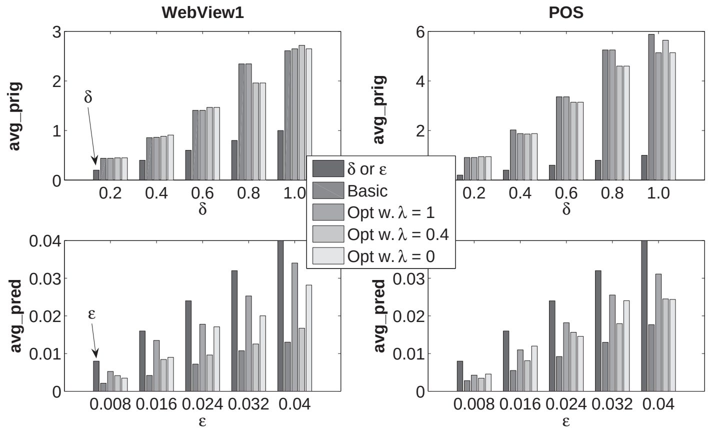

Fig. 5: Average privacy guarantee (avg_prig) and precision degradation (avg_pred).

The maximization of the $(k, 1/k)$ probability is therefore transformed into the following expression (k is omitted since it does not affect the optimization result): $\min \frac{t_j}{t_i}E\left[\frac{t'_i}{t'_j}\right] + \frac{t_i}{t_j}E\left[\frac{t'_j}{t'_i}\right]$. The intuition here is that neither the expectation $\frac{t_j}{t_i}E\left[\frac{t'_i}{t'_j}\right]$ nor $\frac{t_i}{t_j}E\left[\frac{t'_j}{t'_i}\right]$ should deviate far from 1.

According to its definition, $E\left[\frac{t'_i}{t'_j}\right]$ is computed as

$$E\left[\frac{t'_i}{t'_j}\right] = \frac{e_i}{\alpha+1}(H_{e_j+\frac{\alpha}{2}} - H_{e_j-\frac{\alpha}{2}}) \approx \frac{\alpha}{\alpha+1}\frac{e_i}{e_j}$$

where H_n is the nth harmonic number, and the facts that $H_n = \ln n + \Theta(1)$, and that $\forall x, y \in \mathbb{R}^+$, $(1 + x/y)^{y+x/2} \approx e^x$ are applied.

Bottom-up Bias Setting The minimization of the $(k, 1/k)$ probability of t'_i/t'_j is now simplified as

$$\min \quad \frac{t_j}{t_i}\frac{e_i}{e_j} + \frac{t_i}{t_j}\frac{e_j}{e_i}$$

Assume that e_i is fixed, by differentiating the expression w.r.t. e_j, and setting the derivative as zero, one gets the solution of e_j as $e_j/e_i = t_j/t_i$, i.e., $\beta_j/\beta_i = t_j/t_i$.

Following this solution is our bottom-up bias setting scheme: for each FEC fec_i, its bias β_i should be set in proportion to its support t_i. Note that the larger $t_i + \beta_i$ compared with α, the more accurate the approximation applied here, therefore β_i should be set as its maximum possible value. The whole scheme runs as follows: one first sets the bias of the minimum FEC fec_1 as its maximum β_{m1}, and for each remaining FEC fec_i, its bias β_i is set in proportion to $\frac{t_i}{t_{i-1}}$. In this schema, for any pair of fec_i and fec_j, their bias satisfy that $\beta_i/\beta_j = t_i/t_j$. Further, we have the following lemma to show that for each FEC fec_i, β_i falls within the interval $[-\beta_{mi}, \beta_{mi}]$.

Lemma 3. *For two FECs* fec_i *and* fec_j *with* $t_i < t_j$, *if the setting of* β_i *is feasible for* fec_i, *namely within the interval* $[-\beta_{mi}, \beta_{mi}]$, *then the setting* $\beta_j = \beta_i\frac{t_j}{t_i}$ *is feasible for* fec_j.

6.3 A Hybrid Scheme

While order preserving (OP) and ratio preserving (RP) bias settings achieve the maximum utility at their ends, in some applications where both semantic relationships are important, it is desired to balance the two factors in order to achieve the overall optimal quality.

We therefore develop a hybrid scheme that takes advantage of the two approaches, and can flexibly trade between order and ratio preservation.

Specifically, for each FEC fec, let β_{op} and β_{fp} denote its bias setting obtained by the order preserving and frequency preserving approaches respectively. We incorporate them using a linear combination: $\beta = \lambda\beta_{op} + (1-\lambda)\beta_{fp}$.

The parameter $\lambda \in [0, 1]$, controls the balance between the two quality metrics. Intuitively, larger λ tends to assign more importance over absolute order, but less over relative frequency, and vise versa.

7 Experimental Analysis

In this section, we investigate the effectiveness and efficiency of the proposed Butterfly approaches. Specifically, the experiments are designed to measure the following three properties: (i) privacy guarantee: the effectiveness against both intra-/inter-window inferences; (ii) result utility: the degradation of the output precision, the order and ratio preservation, and the tradeoff among these utility metrics; (iii) efficiency: The time taken to perform our approaches. We start by describing the datasets and the setup of the experiments.

7.1 Experimental Setting

We tested our approaches over two real-life datasets. The first one is BMS-WebView-1, which contains a few months of clickstream data from an e-commerce web site, and the second one is BMS-POS, which contains several years of point-of-sale from a large number of electronic retailers. Both datasets have been used in stream frequent pattern mining [10].

We built our Butterfly prototype on top of *Moment* [10], a stream frequent pattern mining framework, which finds closed frequent itemsets over a sliding window model. By default, the minimum support C and vulnerable support K are set as 25 and 5 respectively, and the window size is set as 2K. Note the setting here is designed to test the effectiveness of our approach with high ratio of vulnerable/minimum threshold (K/C). All the experiments are performed over a workstation with Intel Xeon 3.20GHz and 4GB main memory, running Red Hat Linux 9.0 operating system. The algorithm is implemented in C++ and compiled using g++ 3.4.

7.2 Experimental Results

To provide an in-depth understanding of our privacy preservation schemes, we evaluated four different versions of our approach: the basic version, the optimized version with λ = 0, 0.4 and 1 respectively, over both real datasets. Note that λ = 0 corresponds to the ratio preserving scheme, while λ = 1 corresponds to the order preserving one.

Privacy and Precision To evaluate the effectiveness of output privacy protection of our approach, one needs to find all potential privacy breaches in the mining output. This is done by running an analysis program over the results returned by the mining algorithm, and finding all possible vulnerable patterns that can be inferred through either intra-window or inter-window inferences.

Concretely, given a stream window, let $\mathcal{P}_{hv}$ denote all the hard vulnerable patterns that are inferable from the mining output. After the perturbation, we evaluate the square relative deviation of the inferred value and the estimator for each $p \in \mathcal{P}_{hv}$ for 100 continuous windows. we use the following average privacy (avg_prig) metric to measure the effectiveness of the privacy preservation:

$$\mathsf{avg\_prig} = \sum_{p \in \mathcal{P}_{hv}} \frac{(T'(p) - E[T'(p)])^2}{T^2(p)|\mathcal{P}_{hv}|}$$

The decrease in precision of the output is measured by the average precision degradation (avg_pred) of all frequent itemsets $\mathcal{I}$:

$$\mathsf{avg\_pred} = \sum_{I \in \mathcal{I}} \frac{(T'(I) - T(I))^2}{T^2(I)|\mathcal{I}|}$$

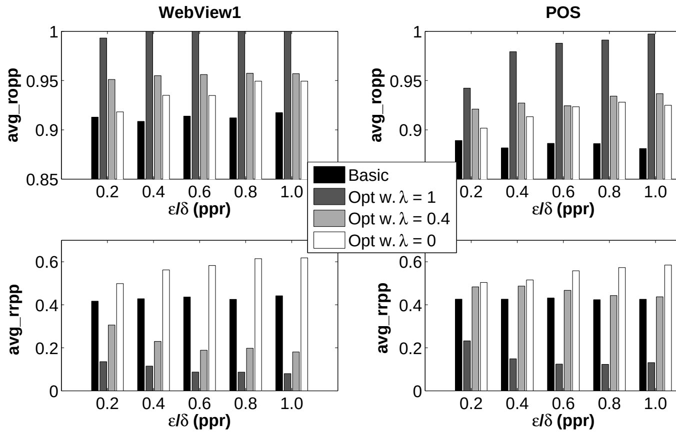

Fig. 6: Average order preservation (avg_ropp) and ratio preservation (avg_rrpp).

In this set of experiments, we fix the precision-privacy ratio ϵ/δ = 0.04, and measure avg_prig and avg_pred for different setting of ϵ (δ).

Specifically, the two plots in the top tier of Fig. 5 show that as the value of δ increases, all four versions of the Butterfly approaches provide similar amount of average privacy protection for both datasets, all above the minimum privacy guarantee δ. The two plots in the lower tier show that as σ increases from 0 to 0.04, the output precision decreases, however all four versions of the Butterfly approaches have average precision degradation below the the system-supplied maximum threshold ϵ. Among them, the basic Butterfly offers the lowest precision loss, which can be explained by the fact that the basic version trades the minimum precision loss for the privacy guarantee, without considering semantic constraints.

Order and Ratio For given privacy and precision requirement (ϵ, δ), we measure the effectiveness of our approaches in preserving order and ratio of frequent itemsets.

The quality of order preservation is evaluated by the ratio of the number of order preserved pairs over the number of possible pairs, referred to as the rate of order preserved pairs (ropp):

$$\mathsf{ropp} = \frac{\sum_{I,J \in \mathcal{I} \cap T(I) \leq T(J)} \mathbf{1}_{T'(I) \leq T'(J)}}{C^2_{|\mathcal{I}|}}$$

where $\mathbf{1}$ is the indicator function, returning 1 if the condition is met, and 0 otherwise.

Analogously, the quality of ratio preservation is evaluated by the fraction of the number of $(k, 1/k)$ probability preserved pair over the number of possible pairs, referred to as the rate of ratio preserved pairs (rrpp) (k is set 0.95 in all the experiments):

$$\mathsf{rrpp} = \frac{\sum_{I,J \in \mathcal{I} \cap T(I) \leq T(J)} \mathbf{1}_{k\frac{T(I)}{T(J)} \leq \frac{T'(I)}{T'(J)} \leq \frac{1}{k}\frac{T(I)}{T(J)}}}{C^2_{|\mathcal{I}|}}$$

In this set of experiments, we vary the precision-privacy ratio ϵ/δ for fixed δ = 0.4, and measure the ropp and rrpp for four versions of the Butterfly approaches (the parameter γ = 2 in all the experiments, a detailed discussion of γ will be followed), as shown in Fig. 6.

As predicted by our theoretical analysis, the order-preserving (λ = 1) and ratio-preserving (λ = 0) bias settings are quite effective, both outperform all other approaches at their ends. The ropp and rrpp increase as the ratio of ϵ/δ grows, due to the fact that larger ϵ/δ offers more adjustable bias therefore leading to better quality.

It is also noticed that order-preserving scheme has the worst performance in the term of avg_rrpp, even worse than the basic approach. This can be explained by that in order to distinguish overlapping FECs, the order-preserving scheme may significantly disturb the ratio of pairs of FECs. In all these

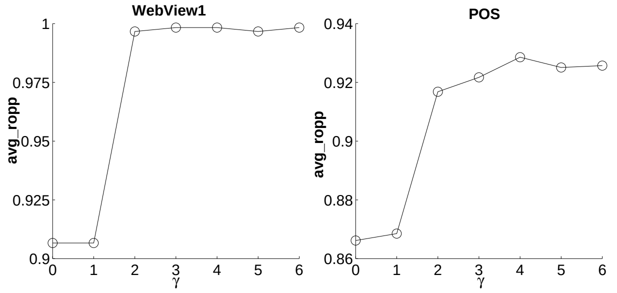

Fig. 7: Average rate of order-preserved pairs versus the setting of γ.

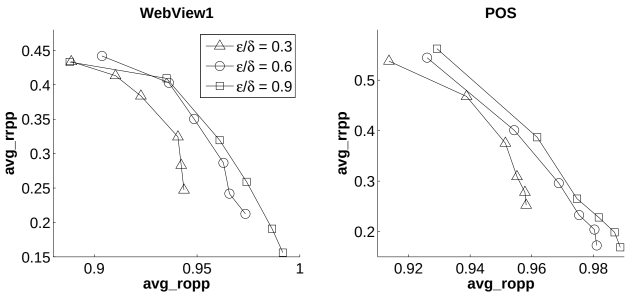

Fig. 8: Tradeoff between order-preservation and ratio-preservation.

cases, the hybrid scheme λ = 0.4 achieves the second best in terms of avg_rrpp and avg_ropp, and an overall best quality when order and ratio preservation are equally important.

Tuning of Parameters γ and λ Here we give a detailed discussion over the setting of the parameters γ and λ.

Specifically, γ controls the depth of the dynamic programming in the order-preserving bias setting. Intuitively, larger γ leads to better order preservation, but also higher time/space complexity. We desire to characterize the increase of order-preservation quality in term of γ to find the setting that balances the computation complexity and the quality.

For both datasets, we measure the ropp versus the setting of γ, with result shown in Fig. 7. It is noted that the quality of order-preservation increases sharply at certain points γ = 2 or 3, and the trend becomes much flatter after that. This is explained by the fact that in most real datasets, the distribution of FECs is not extremely dense, hence under proper setting of (ϵ, δ), a FEC can intersect with only 2 or 3 neighboring FECs on average. Therefore, a setting of small γ is usually sufficient for most datasets.

The setting of λ balances the order and ratio-preservation. For each dataset, we evaluate ropp and rrpp with different setting of λ (0.2, 0.4, 0.6, 0.8 and 1) and precision-privacy ratio ϵ/δ (0.3, 0.6 and 0.9), as shown in Fig. 8.

These plots give good estimate of the gain of order preservation, given the ratio preservation one is willing to sacrifice. A larger ϵ/δ gives more room for this adjustment. In most cases, the setting of λ = 0.4 offers a good balance between the two metrics. The trade-off plots could be made more accurate by choosing more settings of λ and ϵ/δ to explore more points in the space.

Efficiency In evaluating the efficiency, we measure the computation overhead of our Butterfly approaches over the original mining algorithm, given different setting of minimum support C. We divide the execution time into three parts: the mining algorithm (Mining alg), the optimization part seeking the optimal setting (Opt) and the basic perturbation part (Basic). The window size is set 5K for both datasets.

The result plotted in Fig. 9 shows clearly that the overhead of our approaches, especially the basic perturbation operation is non-significant, therefore, it can be readily implemented in current stream mining systems. While the current version of our methods are window-based, in the future work we aim at developing incremental version, and expect even lower overhead.

Note that in most cases, the running time of both mining algorithm and optimization part grow significantly as C decreases, however the growth of the overhead of Butterfly is much less evident than that of the mining algorithm. This is explained by that as the minimum support decreases, the number of frequent itemsets increases super-linearly, but the number of FECs has much lower growth rate, which dominates the performance of the Butterfly approaches.

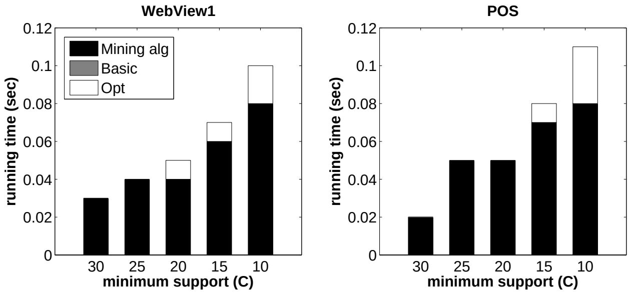

Fig. 9: Overhead of Butterfly approaches in mining systems.

8 Conclusions

In this work, we investigated the problem of protecting output privacy in data stream mining. We presented the inference and disclosure scenarios where the adversary performs attack over the mining output. Motivated by the basis of the attack model, we proposed Butterfly, a two-tier countermeasure: In the first tier, it counters the malicious inferences by amplifying the uncertainty of vulnerable patterns, at the cost of trivial decrease in the output precision; In the second tier, for given privacy and precision requirement, it maximally preserves th utility-related semantics of output, therefore achieving the optimal balance between privacy guarantee and output quality. The effectiveness and efficiency of our approaches are validated by extensive experiments over real-life datasets.

References

[1] D. Agrawal and C. Aggarwal. “On the Design and Quantification of Privacy Preserving Data Mining Algorithms”. In *PODS*, 2001.
[2] M. Atzori, F. Bonchi, F. Giannotti and D. Pedreschi. “Anonymity Preserving Pattern Discovery”. The *VLDB Journal*, 16(4), 2006.
[3] R. Agrawal and R. Srikant. “Privacy Preserving Data Mining”. In *SIGMOD*, 2000.
[4] N. Adam and J. Wortman. “Security-control Methods for Statistical Databases”. In *ACM Computing Surveys*, 21(4), 1999.
[5] S. Bu, L. Lakshmanan, R. Ng and G. Ramesh. “Preservation of Pattern and Input-Output Privacy”. In *ICDE*, 2007.
[6] T. Calders. “Computational Complexity of Itemset Frequency Satisfiability”. In *PODS*, 2004.
[7] K. Chen and L. Liu. “Privacy Preserving Data Classification with Rotation Perturbation”. In *ICDM*, 2005.
[8] F. Chin and G. Ozsoyoglu. “Statistical Database Design”. In *ACM Trans. on Database Systems*, 6(1), 1981.
[9] R. Conway and D. Strip. “Selective Partial Access to A Database”. In *ACM Annual Conf.*, 1976.
[10] Y. Chi, H. Wang, P. Yu and R. Muntz. “Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window”. In *ICDM*, 2004.
[11] A. Evfimevski, R. Srikant, R. Agarwal and J. Gehrke. “Privacy Preserving Mining of Association Rules”. In *SIGKDD*, 2002.
[12] B. Hore, S. Mehrotra and G. Tsudik. “A Privacy-Preserving Index for Range Queries”. In *VLDB*, 2004.
[13] M. Kantarcioglu, J. Jin and C. Clifton. “When do Data Mining Results Violate Privacy?”. In *KDD*, 2004.
[14] Y. Lindell and B. Pinkas. “Privacy Preserving Data Mining”. In *CRYPTO*, 2000.
[15] F. Li, J. Sun, S. Papadimitrious, G. Mihaila and I. Stanoi. “Hiding in the Crowd: Privacy Preservation on Evolving Streams through Correlation Tracking”. In *ICDE*, 2007.
[16] A. Machanavajjhala, J. Gehrke, D. Kifer and M. Venkitasubramaniam. “l-diversity: Privacy beyond k-anonymity”. In *ICDE*, 2006.
[17] L. Sweeney. “k-anonymity: a model for protecting privacy”. In *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5), 2002.
[18] A. Shoshani. “Statistical databases: Characteristics, Problems and Some Solutions”. In *VLDB*, 1982.
[19] T. Calders and B. Goethals. “Mining All Non-Derivable Frequent Itemsets”. In *PKDD*, 2002.
[20] S. Vavasis. “Quadratic Programming Is in NP”. In *Information Processing Letter*, 36(2), 1990.
[21] J. Vaidya and C. Clifton. “Privacy Preserving Association Rule Mining in Vertically Partitioned Data”. In *SIGKDD*, 2002.
[22] K. Wang, B. Fang and F. Yu. “Handicapping Attacker’s Confidence: An Alternative to k-Anonymization”. In *Knowl. and Info. Syst.*, 11(3), 2007.