ISSN:0975-9646

A New Approach Slicing for Micro Data Publishing

Dr. S. Govinda Rao[1] , D. Siva Prasad[2], M. Eswara Rao[1]

[1]Dept. of CSE, TP inst. Of Science & Tech., BOBBILI, A.P., India
[2]Dept. of CSE. Rajah RSRKRR College ,BOBBILI, A.P., India

Abstract-More techniques, such as generalization and bucketization, have been introduced for privacy preserving micro data publishing. Recent tasks have cleared that generalization loses some amount of information, especially for large (high-dimensional) data. Bucketization, is the another technique, does not prevent membership disclosure and does not apply for data that do not have a clear separation between quasi-identifying attributes and sensitive attributes. In this paper, we introduce technique called slicing, which partitions the data both horizontally and vertically. We represent that slicing preserves best data utility than generalization and can be used for membership disclosure protection. Another important advantage of slicing is that it can handle high-dimensional data. We shown how slicing can be used for attribute disclosure protection and develop an efficient algorithm for computing the sliced data that obey the ℓ-diversity requirement. Our workload experiments confirm that slicing preserves better utility than generalization and is more effective than bucketization in workloads involving the sensitive attribute. Our experiments also describes that slicing can be used to prevent membership disclosure.

Key words: Generalization, buketization, slicing, ℓ-diversity.

INTRODUCTION

Micro data has been studied extensively in recent years. The micro data contain records each of which contains information about an individual object, such as a person, a household, or an company. Several micro data economization techniques have been proposed. The most popular ones are generalization for k-anonymity and bucketization for ℓ-diversity. In both approaches, attributes are partitioned into three categories:

(a) Attributes are identifiers that can uniquely identify an individual, such as Name or Identification Number;
(b) Attributes are Quasi-Identifiers (QI), which the adversary (possibly from other publicly-available databases) and which, when taken together, can potentially identify an individual, e.g., Date of join, Sex, and Zip code;
(c) Attributes are Sensitive Attributes (SAs), which are unknown to the adversary and are considered sensitive, such as Disease and Salary.

In both generalization and bucketization, one first removes identifiers from the data and then partitions tuples into buckets. The two techniques differ in the next step. Generalization transforms the QI-values in each bucket into "less specific but semantically consistent" values so that tuples in the same bucket cannot be distinguished by their QI values. In bucketization, one separates the SAs from the QIs by randomly permuting the SA values in each bucket. The anonymized data consists of a set of buckets with permuted sensitive attribute values.

EXISTING SYSTEM:

First, many existing clustering algorithms (e.g., k- means) requires the calculation of the "centroids". But there is no notion of "centroids" in our setting where each attribute forms a data point in the clustering space. Second, k-medoid method is very robust to the existence of outliers (i.e., data points that are very far away from the rest of data points). Third, the order in which the data points are examined does not affect the clusters computed from the k-medoid method.

PROBLEMS IN EXITING SYSTEM

1. Existing privacy measures for membership disclosure protection include differential privacy and presence.
2. Existing anonymization algorithms can be used for column generalization, e.g., Mondrian. The algorithms can be applied on the sub table containing only attributes in one column to ensure the anonymity requirement.
3. Existing data analysis (e.g., query answering) methods can be easily used on the sliced data.

PROPOSED SYSTEM:

We are proposing the technique called slicing, which will divide the data both horizontally and vertically. We show that slicing preserves better data utility than generalization and can be used for membership disclosure protection. Another important advantage of slicing is that it can handle high-dimensional data. We show how slicing can be used for attribute disclosure protection and develop an efficient algorithm for computing the sliced data that obey the ℓ-diversity requirement. Our workload experiments confirm that slicing preserves better utility than generalization and is more effective than bucketization in workloads involving the sensitive attribute.

Advantages:

1. We introduce a novel data anonymization technique called slicing to improve the current state of the art.
2. We show that slicing can be effectively used for preventing attribute disclosure, based on the privacy requirement of ℓ-diversity.
3. We develop an efficient algorithm for computing the sliced table that satisfies ℓ-diversity. Our algorithm partitions attributes into columns, applies column generalization, and partitions tuples into buckets. Attributes that are highly-correlated are in the same column.
4. We conduct extensive workload experiments. Our results confirm that slicing preserves much better data utility than generalization. In workloads involving the sensitive

attribute, slicing is also more effective than bucketization. In some classification experiments, slicing shows better performance than using the original data (which may over fit the model). Our experiments also show the limitations of bucketization in membership disclosure protection and slicing remedies these limitations.

Slicing Algorithms:

Algorithm tuple-partition(T, ℓ)

```
1. Q = {T}; SB = ∅.
2. while Q is not empty
3. remove the first bucket B from Q; Q = Q − {B}.
4. split B into two buckets B1 and B2, as in Mondrian.
5. if diversity-check(T, Q ∪ {B1,B2} ∪ SB, ℓ)
6. Q = Q ∪ {B1,B2}.
7. else SB = SB ∪ {B}.
8. return SB.
```

Algorithm diversity-check(T,T_, ℓ)

```
1. for each tuple t ∈ T, L[t] = ∅.
2. for each bucket B in T_
3. record f(v) for each column value v in bucket B.
4. for each tuple t ∈ T
5. calculate p(t,B) and find D(t,B).
6. L[t] = L[t] ∪ {hp(t,B),D(t,B)i}.
7. for each tuple t ∈ T
8. calculate p(t, s) for each s based on L[t].
9. if p(t, s) ≥ 1/ℓ, return false.
10. return true.
```

Steps to achieve the specified Proposed Approach

1. Original Data
2. Generalized Data
3. Bucketized Data
4. Multi-set-based Generalization Data
5. One-attribute-per-Column Slicing Data
6. Sliced Data

ORIGINAL DATA:

We conduct extensive workload experiments. Our results confirm that slicing preserves much better data utility than generalization. In workloads involving the sensitive attribute, slicing is also more effective than bucketization. In some classification experiments, slicing shows better performance than using the original data.

Age	Material status	Pin code	District
33	M	535557	VZM
34	UM	535558	VZM
45	M	535546	VSP
50	M	534431	SKLM
52	UM	534435	HYD
60	M	534436	HYD
60	UM	534431	SKLM

Table 1. Original data

GENERALIZED DATA:

Generalized Data, in order to perform data analysis or data mining tasks on the generalized table, the data analyst has to make the uniform distribution assumption that every value in a generalized interval/set is equally possible, as no other distribution assumption can be justified. This significantly reduces the data utility of the generalized data.

Age	Material status	Pin code	District
33-45	*	5355*	VZM
33-45	*	5355*	VSP
33-45	*	5355*	VZM
50-60	*	5344*	SKLM
50-60	*	5344*	HYD
50-60	*	5344*	HYD
50-60	*	5344*	SKLM

Table 2. Geralizied table

BUCKETIZED DATA:

We show the effectiveness of slicing in membership disclosure protection. For this purpose, we count the number of fake tuples in the sliced data. We also compare the number of matching buckets for original tuples and that for fake tuples. Our experiment results show that bucketization does not prevent membership disclosure as almost every tuple is uniquely identifiable in the bucketized data.

Age	Material status	Pin code	District
33	M	535557	VZM
34	M	535558	VZM
45	UM	535546	VSP
50	M	534431	SKLM
52	M	534435	HYD
60	UM	534436	HYD
60	UM	534431	SKLM

Table 3. Bucketized Data:

MULTI-SET-BASED GENERALIZATION DATA:

We observe that this multi-set-based generalization is equivalent to a trivial slicing scheme where each column contains exactly one attribute, because both approaches preserve the exact values in each attribute but break the association between them within one bucket.

Age	Material status	Pin code	District
33:1,34:1,45:1	M:2,UM:1	535546:1,535558:1,535557:1	VZM
33:1,34:1,45:1	M:2,UM:1	535546:1,535558:1,535557:1	VSP
33:1,34:1,45:1	M:2,UM:1	535546:1,535558:1,535557:1	VZM
50:1,52:1,60:2	M:2,UM:2	534431:2,534435:1,534436:1	HYD
50:1,52:1,60:2	M:2,UM:2	534431:2,534435:1,534436:1	SKLM
50:1,52:1,60:2	M:2,UM:2	534431:2,534435:1,534436:1	SKLM
50:1,52:1,60:2	M:2,UM:2	534431:2,534435:1,534436:1	HYD

Table 4. Multi-set based generalized table

One-attribute-per-column slicing data:

We observe that while one-attribute-per-column slicing preserves attribute distributional information, it does not preserve attribute correlation, because each attribute is in its own column. In slicing, one group correlated attributes together in one column and preserves their correlation. For example, in the sliced table shown in Table correlations between Age and Material status and correlations between Pin code and District are preserved. In fact, the sliced table encodes the same amount of information as the original data with regard to correlations between attributes in the same column.

Age	Material status	Pin code	District
33	M	535557	VZM
34	M	535558	VZM
45	UM	535546	VSP
50	M	534431	SKLM
52	M	534435	HYD
60	UM	534436	HYD
60	UM	534431	SKLM

Table 5. One attribute per column slicing

Sliced data:

Another important advantage of slicing is its ability to handle high-dimensional data. By partitioning attributes into columns, slicing reduces the dimensionality of the data. Each column of the table can be viewed as a sub-table with a lower dimensionality. Slicing is also different from the approach of publishing multiple independent sub-tables in that these sub-tables are linked by the buckets in slicing.

(Age, Material status)	(Pin code, District)
(33,M)	(535557,VZM)
(34,M)	(535558,VZM)
(45,UM)	(535546,VSP)
(50,M)	(534431,SKLM)
(52,M)	(534435,HYD)
(60,UM)	(534436,HYD)
(60,UM)	(534431,SKLM)

Table 6. Sliced table

Conclusion

In this paper, we consider slicing where each attribute is in exactly one column. An extension is the notion of overlapping slicing, which duplicates an attribute in more than one column. These releases more attribute correlations. For example, in Table, one could choose to include the Disease attribute also in the first column. That is, the two columns are {Age, Marital status} and {Pin code, District}. This could provide better data utility, but the privacy implications need to be carefully studied and understood. It is interesting to study the tradeoff between privacy and utility.

We plan to study membership disclosure protection in more details. Our experiments show that random grouping is not very effective. We plan to design more effective tuple grouping algorithms.

Slicing is a promising technique for handling high-dimensional data. By partitioning attributes into columns, we protect privacy by breaking the association of uncorrelated attributes and preserve data utility by preserving the association between highly-correlated attributes. For example, slicing can be used for anonymizing transaction databases, which has been studied recently.

Finally, while a number of anonymization techniques have been designed, it remains an open problem on how to use the anonymized data. In our experiments, we randomly generate the associations between column values of a bucket. This may lose data utility. Another direction to design data mining tasks using the anonymized data computed by various anonymization techniques.

References

1. Accoria. Rock web server and load balancer. http://www.accoria.com.
2. Amazon Web Services. Amazon Web Services (AWS). http://aws.amazon.com.
3. V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on web-server systems. IEEE Internet Computing, 3(3):28{39, 1999.
4. L. Cherkasova. FLEX: Load Balancing and Management Strategy for Scalable Web Hosting Service. IEEE Symposium on Computers and Communications, 0:8, 2000.
5. F5 Networks. F5 Networks. http://www.f5.com.
6. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol { http/1.1. In IETF RFC 2616, 1999.
7. Google Inc. Google App Engine. http://code.google.com/appengine/.
8. HaProxy. HaProxy load balancer. http://haproxy.1wt.eu/.
9. G. Hunt, E. Nahum, and J. Tracey. Enabling content-based load distribution for scalable services. Technical report, 1997.
10. E. Katz, M. Butler, and R. McGrath. A scalable HTTP server: The NCSA prototype. In Proc. First International Conference on the World Wide Web, Apr. 1994

.