Chapter 20

Privacy-Preserving Data Mining

"*Civilization is the progress toward a society of privacy. The savage's whole existence is public, ruled by the laws of his tribe. Civilization is the process of setting man free from men.*"—Ayn Rand

20.1 Introduction

A significant amount of application data is of a personal nature. These kind of data sets may contain sensitive information about an individual, such as his or her financial status, political beliefs, sexual orientation, and medical history. The knowledge about such personal information can compromise the privacy of individuals. Therefore, it is crucial to design data collection, dissemination, and mining techniques, so that individuals are assured of their privacy. Privacy-preservation methods can generally be executed at different steps of the data mining process:

1. *Data collection and publication:* The privacy-driven modification of a data set may be done at either the data *collection* time, or the data *publication* time. In anonymous data *collection*, a modified version of the data is collected using a software plugin within the collection platform. Therefore, the contributors of the data are assured that their data is not available even to the entity collecting the data. The implicit assumption in the collection-oriented model is that the data collector is not trusted, and therefore the privacy must be preserved at collection time. In anonymous data *publication*, the entire data set is available to a trusted entity, who has usually collected the data in the normal course of business. An example is a hospital that has collected data about its patients. Eventually, the entity may wish to release or *publish* the data to one of more third-parties for data analysis. For example, a hospital may want to use the data to study the long-term impact of various treatment alternatives. A real-world example is the Netflix prize data set [559], in which the anonymized movie ratings of users were published to advance studies on collaborative filtering algorithms. During data publication, identifying or sensitive attribute values need to either be removed or be specified approximately to preserve privacy. Generally, such publication algorithms

C. C. Aggarwal, *Data Mining: The Textbook*, DOI 10.1007/978-3-319-14142-8_20

can control the level of privacy much better than collection algorithms, because of their access to the entire data set on a trusted server.

2. *Output privacy of data mining algorithms:* Privacy can also be violated by the *output* of data mining algorithms. For example, consider a scenario where a user is allowed to determine association patterns, or otherwise query the data through a Web service, but is not provided access to the data set. In such a case, the output of the data mining and query processing algorithms provides valuable information, some of which may be private.

In some applications, organizations may wish to share their data in a private way, so that only patterns in the *shared* data may be mined, but the statistics of the *local* databases are not revealed to the participants. This problem is referred to as *distributed privacy preservation*.

In general, most forms of privacy-preserving data mining reduce the representation accuracy of the data, in order to preserve privacy. This accuracy reduction is performed in a variety of ways, such as data distortion, approximation (generalization), suppression, attribute value swapping, or microaggregation. Clearly, since the data is no longer specified exactly, this will have a detrimental impact on the quality of the data mining results. The effectiveness of the released data for mining applications is often quantified explicitly, and is referred to as its *utility*. A natural trade-off exists between privacy and utility. For example, in a case, where data values are suppressed, one might simply choose to suppress all entries. While such a solution provides perfect privacy, it offers no utility. This observation is also true for privacy-preserving publication algorithms in which noise is added to the data. When a greater amount of noise is added, a higher level of privacy is achieved, but utility is reduced. The goal of privacy-preservation methods is to maximize utility at a fixed level of privacy.

This chapter is organized as follows. Methods for privacy-preserving data collection are addressed in Sect. 20.2. Section 20.3 addresses the problem of privacy-preserving data publishing. This section includes several models such as the k-anonymity model, the ℓ-diversity model, and the t-closeness model. The problem of output privacy is addressed in Sect. 20.4. Methods for distributed and cryptographic privacy are discussed in Sect. 20.5. A summary is given in Sect. 20.6.

20.2 Privacy During Data Collection

The randomization method is designed for privacy-preservation at data *collection* time. The implicit assumption is that the data collector is not trusted, and therefore the privacy must be preserved at data collection time. The basic idea of the approach is to allow users to enter the data through a software platform that is able to add random perturbations to the data. This approach is one of the most conservative models for ensuring data privacy, because the original data records are never stored on any single server.

The random perturbations are added using a *publicly available* distribution. Examples of commonly used perturbing distributions include the uniform and the Gaussian distributions. In other words, the probability distribution used to perturb the data is specified together with the data set if and when the data collector releases the data for public use. This additional distribution information is needed to use the data effectively in the context of data mining algorithms. The basic idea is to reconstruct the distribution of the original data, by "subtracting out" the noise distribution. This aggregate distribution is then used for mining purposes. The overall approach is as follows:

1. *Privacy-preserving data collection:* In this step, random noise is added to the data while collecting data from users, with the use of a software plugin. The collected data is publicly released along with the probability distribution function (and parameters) used to add the random noise.

2. *Distribution reconstruction:* The aggregate distribution of the original data are reconstructed, by "subtracting out" the noise. Thus, at the end of this step, we will have a histogram representing the approximate probability distribution of the data values.

3. *Data mining:* Data mining methods are applied to the reconstructed distributions.

It is important to note that the last step of the process requires the design of data mining algorithms that work with probability distributions of *sets of data records*, rather than *individual* records. Thus, one disadvantage of this approach is that it requires the redesign of data mining algorithms. Nevertheless, the approach can be made to work because many data mining problems such as clustering and classification require only the probability distribution modeling of either the whole data set, or segments (e.g., different classes) of the data.

20.2.1 Reconstructing Aggregate Distributions

The reconstruction of the aggregate distribution of the original data is the key step in the randomization method. Consider the case where the original data values $x_1 \ldots x_n$ are drawn from the probability distribution $\mathbf{X}$. For each original data value x_i, a perturbation y_i is added by the software data collection tool, to yield the perturbed value z_i. The perturbation y_i is drawn from the probability distribution $\mathbf{Y}$, and is independent of $\mathbf{X}$. It is assumed that this distribution is known publicly. Furthermore, the probability distribution of the final set of perturbed values is assumed to be $\mathbf{Z}$. Therefore, the original distribution $\mathbf{X}$, the added perturbation $\mathbf{Y}$ and the final aggregate distribution $\mathbf{Z}$ are related as follows:

$$\mathbf{Z} = \mathbf{X} + \mathbf{Y}$$
$$\mathbf{X} = \mathbf{Z} - \mathbf{Y}$$

Thus, the probability distribution of $\mathbf{X}$ can be reconstructed, if the distributions of $\mathbf{Y}$ and $\mathbf{Z}$ are known explicitly. The probability distribution of $\mathbf{Y}$ is assumed to be publicly available, while discrete samples of $\mathbf{Z}$ are available in terms of $z_1 \ldots z_n$. These discrete samples are sufficient to reconstruct $\mathbf{Z}$ using a variety of methods, such as kernel density estimation. Then, the distribution for $\mathbf{X}$ can be reconstructed using the relationship shown above. The main problem with this approach emerges when the probability distribution of the perturbation $\mathbf{Y}$ has a large variance and the number n of discrete samples of $\mathbf{Z}$ is small. In such a case, the distribution of $\mathbf{Z}$ also has a large variance, and it cannot be accurately estimated with a small number of samples. Therefore, a second approach is to *directly* estimate the distribution of $\mathbf{X}$ from the discrete samples of $\mathbf{Z}$ and the known distribution of $\mathbf{Y}$.

Let $f_{\mathbf{X}}$ and $F_{\mathbf{X}}$ be the probability density and cumulative distributions functions of $\mathbf{X}$. These functions need to be *estimated* with the observed values $z_1 \ldots z_n$. Let $\hat{f}_{\mathbf{X}}$ and $\hat{F}_{\mathbf{X}}$ be the corresponding *estimated* probability density and cumulative distribution functions for $\mathbf{X}$. The key here is to use the Bayes formula with the use of observed values for $\mathbf{Z}$. Consider a simplified scenario in which only a single observed value z_1 is available. This can be used

to estimate the cumulative distribution function $\hat{F}_{\mathbf{X}}(a)$ at any value of the random variable $\mathbf{X} = a$. The Bayes theorem yields the following:

$$\hat{F}_{\mathbf{X}}(a) = \frac{\int_{w=-\infty}^{w=a} f_{\mathbf{X}}(w|\mathbf{X}+\mathbf{Y}=z_1)dw}{\int_{w=-\infty}^{w=\infty} f_{\mathbf{X}}(w|\mathbf{X}+\mathbf{Y}=z_1)dw} \tag{20.1}$$

The conditional in the aforementioned equation corresponds to the fact that the sum of the data values and perturbed values is equal to z_1. Note that the expression $f_{\mathbf{X}}(w|\mathbf{X}+\mathbf{Y}=z_1)$ can be expressed in terms of the unconditional densities of $\mathbf{X}$ and $\mathbf{Y}$, as follows:

$$f_{\mathbf{X}}(w|\mathbf{X}+\mathbf{Y}=z_1) = f_{\mathbf{Y}}(z_1-w)\cdot f_{\mathbf{X}}(w) \tag{20.2}$$

This expression uses the fact that the perturbation $\mathbf{Y}$ is independent of $\mathbf{X}$. By substituting the aforementioned expression for $f_{\mathbf{X}}(w|\mathbf{X}+\mathbf{Y}=z_1)$ in the right-hand side of Eq. 20.1, the following expression is obtained for the cumulative density of $\mathbf{X}$:

$$\hat{F}_{\mathbf{X}}(a) = \frac{\int_{w=-\infty}^{w=a} f_{\mathbf{Y}}(z_1-w)\cdot f_{\mathbf{X}}(w)dw}{\int_{w=-\infty}^{w=\infty} f_{\mathbf{Y}}(z_1-w)\cdot f_{\mathbf{X}}(w)dw} \tag{20.3}$$

The expression for $\hat{F}_{\mathbf{X}}(a)$ was derived using a *single* observation z_1, and needs to be generalized to the case of n different observations $z_1 \ldots z_n$. This can be achieved by averaging the previous expression over n different values:

$$\hat{F}_{\mathbf{X}}(a) = \frac{1}{n}\cdot\sum_{i=1}^{n}\frac{\int_{w=-\infty}^{w=a} f_{\mathbf{Y}}(z_i-w)\cdot f_{\mathbf{X}}(w)dw}{\int_{w=-\infty}^{w=\infty} f_{\mathbf{Y}}(z_i-w)\cdot f_{\mathbf{X}}(w)dw} \tag{20.4}$$

The corresponding density distribution can be obtained by differentiating $\hat{F}_{\mathbf{X}}(a)$. This differentiation results in the removal of the integral sign from the numerator and the corresponding instantiation of w to a. Since the denominator is a constant, it remains unaffected by the differentiation. Therefore, the following is true:

$$\hat{f}_{\mathbf{X}}(a) = \frac{1}{n}\cdot\sum_{i=1}^{n}\frac{f_{\mathbf{Y}}(z_i-a)\cdot f_{\mathbf{X}}(a)}{\int_{w=-\infty}^{w=\infty} f_{\mathbf{Y}}(z_i-w)\cdot f_{\mathbf{X}}(w)dw} \tag{20.5}$$

The aforementioned equation contains the density function $f_{\mathbf{X}}(\cdot)$ on both sides. This circularity can be resolved naturally with the use of an iterative approach. The iterative approach initializes the estimate of the distribution $f_{\mathbf{X}}(\cdot)$ to the uniform distribution. Subsequently, the estimate of this distribution is continuously updated as follows:

Set $\hat{f}_{\mathbf{X}}(\cdot)$ to be the uniform distribution;
repeat
Update $\hat{f}_{\mathbf{X}}(a) = (1/n)\cdot\sum_{i=1}^{n}\frac{f_{\mathbf{Y}}(z_i-a)\cdot\hat{f}_{\mathbf{X}}(a)}{\int_{w=-\infty}^{w=\infty} f_{\mathbf{Y}}(z_i-w)\cdot\hat{f}_{\mathbf{X}}(w)dw}$
until convergence

So far, it has been described, how to compute $f_{\mathbf{X}}(a)$ for a *particular* value of a. In order to generalize this approach, the idea is to discretize the range of the random variable $\mathbf{X}$ into k intervals, denoted by $[l_1, u_1] \ldots [l_k, u_k]$. It is assumed that the density distribution is uniform over the discretized intervals. For each such interval $[l_i, u_i]$, the density distribution is evaluated at the midpoint $a = (l_i + u_i)/2$ of the interval. Thus, in each iteration, k different values of a are used. The algorithm is terminated when the distribution does not

change significantly over successive steps of the algorithm. A variety of methods can be used to compare the two distributions such as the χ^2 test. The simplest approach is to examine the average change in the density values, at the midpoints of the density distribution over successive iterations. While this algorithm is known to perform effectively in practice, it is not proven to be a optimally convergent solution. An expectation maximization (EM) method was proposed in a later work [28], which provably converges to the optimal solution.

20.2.2 Leveraging Aggregate Distributions for Data Mining

The aggregate distributions determined by the algorithm can be leveraged for a variety of data mining problems, such as clustering, classification, and collaborative filtering. This is because each of these data mining problems can be implemented with aggregate statistics of the data, rather than the original data records. In the case of the classification problem, the probability distributions of each of the classes can be reconstructed from the data. These distributions can then be used directly in the context of a naive Bayes classifier, as discussed in Chap. 10. Other classifiers, such as decision trees, can also be modified to work with aggregate distributions. The key is to use the aggregate distributions in order to design the split criterion of the decision tree. The bibliographic notes contain pointers to data mining algorithms that use the randomization method. The approach cannot be used effectively for data mining problems such as outlier detection that are dependent on individual data record values, rather than aggregate values. In general, outlier analysis is a difficult problem for most private data sets because of the tendency of outliers to reveal private information.

20.3 Privacy-Preserving Data Publishing

Privacy-preserving data *publishing* is distinct from privacy-preserving data *collection*, because it is assumed that all the records are already available to a trusted party, who might be the current owner of the data. This party then wants to release (or *publish*) this data for analysis. For example, a hospital might wish to release anonymized records about patients to study the effectiveness of various treatment alternatives.

This form of data release is quite useful, because virtually any data mining algorithm can be used on the released data. To determine sensitive information about an individual, there are two main pieces of information that an attacker (or *adversary*) must possess.

1. Who does this data record pertain to? While a straightforward way to determine the identity is to use the identifying attribute (e.g., Social Security Number), such attributes are usually stripped from the data before release. As will be discussed later, these straightforward methods of sanitization are usually not sufficient, because attackers may use other attributes, such as the age and ZIP code, to make *linkage* attacks.

2. In addition to identifying attributes, data records also contain *sensitive* attributes that most individuals do not wish to share with others. For example, when a hospital releases medical data, the records might contain sensitive disease-related attributes.

Different attributes in a data set may play different roles in either facilitating identification or facilitating sensitive information release. There are three main types of attributes:

Table 20.1: Example of a data table

SSN	Age	ZIP Code	Disease
012-345-6789	24	10598	HIV
823-627-9231	37	90210	Hepatitis C
987-654-3210	26	10547	HIV
382-827-8264	38	90345	Hepatitis C
847-872-7276	36	89119	Diabetes
422-061-0089	25	02139	HIV

1. *Explicit identifiers:* These are attributes that explicitly identify an individual. For example, the Social Security Number (SSN) of an individual can be considered an explicit identifier. Because this attribute is almost always removed in the data sanitization process, it is not relevant to the study of privacy algorithms.

2. *Pseudo-identifier or quasi-identifier (QID):* These are attributes that do not explicitly identify an individual in *isolation*, but can nevertheless be used *in combination* to identify an individual by joining them with publicly available information, such as voter registration rolls. This kind of attack is referred to as a *linkage* attack. Examples of such attributes include the Age and ZIP code. Strictly speaking, quasi-identifiers refer to the specific combination of attributes used to make a linkage attack, rather than the individual attributes.

3. *Sensitive attribute:* These are attributes that are considered private by most individuals. For example, in a medical data set, individuals would not like information about their diseases to be known publicly. In fact, many laws in the USA, such as the Health Insurance Portability and Accountability Act (HIPAA), explicitly forbid the release of such information, especially when the sensitive attributes can be linked back to specific individuals.

Most of the discussion in this chapter will be restricted to quasi-identifiers and sensitive attributes. To illustrate the significance of these attribute types, an example will be used. In Table 20.1, the medical records of a set of individuals are illustrated. The *SSN* attribute is an explicit identifier that can be utilized to identify an individual directly. Such *directly* identifying information will almost always be removed from a data set before release. However, the impact of attributes such as the age and the ZIP code on identification is quite significant. While these attributes do not directly identify an individual, they provide very useful hints, when combined with other publicly available information. For example, it is possible for a small geographic region, such as a ZIP code, to contain only one individual of a specific gender, race, and date of birth. When combined with publicly available voter registration rolls, one might be able to identify an individual from these attributes. Such a *combination* of publicly available attributes is referred to as a quasi-identifier.

To understand the power of quasi-identifiers, consider a snapshot of the voter registration rolls illustrated in Table 20.2. Even in cases, where the SSN is removed from Table 20.1 before release, it is possible to join the two tables with the use of the age and ZIP code attributes. This will provide a list of the *possible* matches for each data record. For example, Joy and Sue are the only two individuals in the voter registration rolls matching an individual with HIV in the medical release of Table 20.1. Therefore, one can tell with 50 % certainty that Joy and Sue have HIV. This is not desirable especially when an adversary

Table 20.2: Example of a snapshot of fictitious voter registration rolls

Name	Age	ZIP Code
Mary A.	38	90345
John S.	36	89119
Ann L.	31	02139
Jack M.	57	10562
Joy M.	26	10547
Victor B.	46	90345
Peter P.	25	02139
Diana X.	24	10598
William W.	37	90210
Sue G.	26	10547

has other background medical information about Joy or Sue to further narrow down the possibilities. Similarly, William is the only individual in the voter registration rolls, who matches an individual with hepatitis C in the medical release. In cases, where only one data record in the voter registration rolls matches the particular combination of age and ZIP code, sensitive medical conditions about that individual may be fully compromised. This approach is referred to as a *linkage* attack. Most anonymization algorithms focus on preventing *identity disclosure*, rather than explicitly hiding the sensitive attributes. Thus, only the attributes which can be combined to construct quasi-identifiers are changed or specified approximately in the data release, whereas sensitive attributes are released in their exact form.

Many privacy-preserving data publishing algorithms assume that the quasi-identifiers are drawn out of a set of attributes that are not sensitive, because they can only be used by an adversary by performing joins with (nonsensitive) publicly available information. This assumption may, however, not always be reasonable, when an adversary has (sensitive) background information about a target at hand. Adversaries are often familiar with their targets, and they can be assumed to have background knowledge about at least a subset of the sensitive attributes. In a medical application with multiple disease attributes, knowledge about a subset of these attributes may reveal the identity of the subject of the record. Similarly, in a movie collaborative filtering application, where anonymized ratings are released, it may be possible to obtain information about a particular user's ratings on a subset of movies, through personal interaction or other rating sources. If this combination is unique to the individual, then the other ratings of the individual are compromised as well. Thus, sensitive attributes also need to be perturbed, when background knowledge is available. Much of the work in the privacy literature assumes a rigid distinction between the role of publicly available attributes (from which the quasi-identifiers are constructed) and that of the sensitive attributes. In other words, sensitive attributes are not perturbed because it is assumed that revealing them does not incur the risk of a linkage attack with publicly available information. There are, however, a few algorithms that do not make this distinction. Such algorithms generally provide better privacy protection in the presence of background information.

In this section, several models for group-based anonymization, such as k-anonymity, ℓ-diversity, and t-closeness, will be introduced. While the recent models, such as ℓ-diversity, have certain advantages over the k-anonymity model, a good understanding of k-anonymity

is crucial in any study of privacy-preserving data publishing. This is because the basic framework for most of the group-based anonymization models was first proposed in the context of the k-anonymity model. Furthermore, many algorithms for other models, such as ℓ-diversity, build upon algorithms for k-anonymization.

20.3.1 The k-Anonymity Model

The k-anonymity model is one of the oldest ones for data anonymization, and it is credited with the understanding of the concept of quasi-identifiers and their impact on data privacy. The basic idea in k-anonymization methods is to allow release of the sensitive attributes, while distorting only the attributes which are available through public sources of information. Thus, even though the sensitive attributes have been released, they cannot be linked to an individual through publicly available records. Before discussing the anonymization algorithms, some of the most common techniques for data distortion will be discussed.

1. *Suppression:* In this approach, some of the attribute values are suppressed. Depending on the algorithm used, the suppression can be done in a variety of ways. For example, one might omit *some of* the age *or* ZIP code attribute values from a few selected data records in Table 20.1. Alternatively, one might completely omit the entire record for a specific individual (row suppression) or the age attribute from all individuals (column suppression). Row suppression is often utilized to remove outlier records because such records are difficult to anonymize. Column suppression is commonly used to remove highly identifying attributes, or explicit identifiers, such as the SSN.

2. *Generalization:* In the case of generalization, the attributes are specified approximately in terms of a particular range. For example, instead of specifying Age = 26 and Location (ZIP Code) = 10547 for one of the entries of Table 20.1, one might *generalize* it to $Age \in [25, 30]$ and Location (State) = New York. By specifying the attributes approximately, it becomes more difficult for an adversary to perform linkage attacks. While numeric data can be generalized to specific ranges, the generalization of categorical data is somewhat more complicated. Typically, a generalization hierarchy of the categorical attribute values needs to be provided, for use in the anonymization process. For example, a ZIP code may be generalized to a city, which in turn may be generalized to a state, and so on. There is no unique way of specifying a domain hierarchy. Typically, it needs to be semantically meaningful, and it is specified by a domain expert as a part of the input to the anonymization process. An example of a generalization taxonomy of categorical attributes for the location attribute of Table 20.1 is provided in Fig. 20.1. This hierarchy of attribute values has a tree structure, and is referred to as a *value generalization hierarchy*. The notations $A_0 \ldots A_3$ and $Z_0 \ldots Z_4$ in Fig. 20.1 denote the domain generalizations at different levels of granularity. The corresponding *domain generalization hierarchies* are also illustrated in the Fig. 20.1 by the single path between $Z_0 \ldots Z_4$ and $A_0 \ldots A_4$.

3. *Synthetic data generation:* In this case, a synthetic data set is generated that mimics the statistical properties of the original data, at the group level. Such an approach can provide better privacy, because it is more difficult to map synthetic data records to particular groups of records. On the other hand, the data records are no longer *truthful* because they are synthetically generated.

4. *Specification as probabilistic and uncertain databases:* In this case, one might specify an individual data record as a probability distribution function. This is different from the

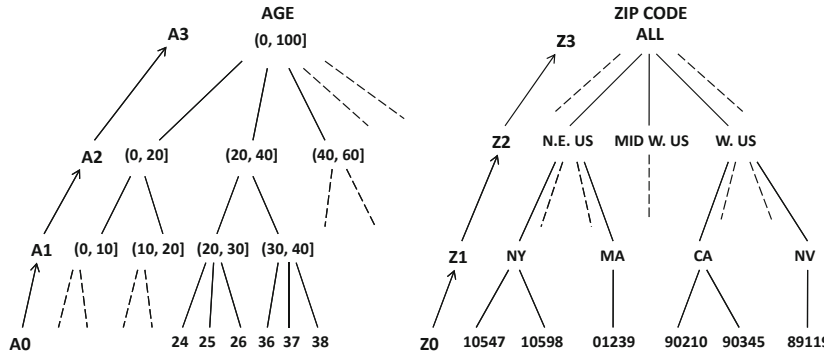


Figure 20.1: A value- and corresponding domain-generalization hierarchy for the age and ZIP code attributes

Table 20.3: Example of a 3-anonymized version of Table 20.1

Row Index	Age	ZIP Code	Disease
1	[20, 30]	Northeastern US	HIV
2	[30, 40]	Western US	Hepatitis C
3	[20, 30]	Northeastern US	HIV
4	[30, 40]	Western US	Hepatitis C
5	[30, 40]	Western US	Diabetes
6	[20, 30]	Northeastern US	HIV

aggregate distribution approach of randomization because the probability distribution is data-record specific, and is designed to ensure k-anonymity. While this approach has not been studied intensively, it has the potential to allow the use of recent advances in the field of probabilistic databases for anonymization.

Among the aforementioned methods, the generalization and suppression methods are most commonly used for anonymization. Therefore, most of the discussion in this section will be focused on these methods. First, the notion of k-anonymity will be defined.

Definition 20.3.1 (k-anonymity) *A data set is said to be k-anonymized, if the attributes of each record in the anonymized data set cannot be distinguished from at least $(k-1)$ other data records.*

This group of indistinguishable data records is also referred to as an *equivalence class*. To understand how generalization and suppression can be used for anonymization, consider the data set in Table 20.1. An example of a 3-anonymized version of this table is illustrated in Table 20.3. The SSN has been fully suppressed with column-wise suppression and replaced with an anonymized row index. Such explicit identifiers are almost always fully suppressed in anonymization. The two publicly available attributes corresponding to the age and ZIP code are now generalized and specified approximately. The subjects of the row indices 1, 3, and 6 can no longer be distinguished by using linkage attacks because their publicly available attributes are identical. Similarly, the publicly available attributes of row indices 2, 4, and 5 are identical. Thus, this table contains two *equivalence classes* containing three records each, and the data records cannot be distinguished from one another within these

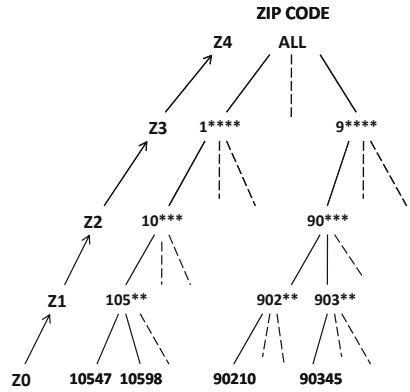


Figure 20.2: An alternate value- and corresponding domain-generalization hierarchy for the ZIP code attribute

equivalence classes. In other words, an adversary can no longer match the identification of individual data records with voter registration rolls *exactly*. If any matching is found, then it is guaranteed that at least $k = 3$ records in the data set will match any particular individual in the voter registration roll.

The ZIP code is generalized with the use of the prespecified value generalization hierarchy of Fig. 20.1. The generation of a domain generalization hierarchy for a categorical attribute can be done in several ways, and depends on the skill of the analyst responsible for the privacy modifications. An alternate example of a domain generalization hierarchy for the ZIP code attribute is illustrated in Fig. 20.2. A value generalization hierarchy on the continuous attributes does not require any special domain knowledge because it can be directly created by the analyst, using the actual distribution of the continuous values in the underlying data. This requires a simple hierarchical discretization of the continuous attributes.

The goal of the privacy-preservation algorithms is to replace the original values in the data (numeric or discrete), with one of the discrete values illustrated in the taxonomy trees of Fig. 20.1. Thus, the data is *recoded* in terms of a new set of discrete values. In most cases, the numeric attributes do retain their ordering, because the corresponding ranges are ordered. Different algorithms use different rules in the recoding process. These different ways of recoding attributes may be distinguished as follows:

- *Global versus local recoding:* In global recoding, a given attribute value is always replaced with the same discrete counterpart from the domain generalization hierarchy over all data records. Consider the aforementioned example of Fig. 20.1, in which ZIP code can be generalized either to state or region. In global recoding, the particular ZIP code value of 10547 needs to be *consistently* replaced by either *Northeastern US*, or *New York* over all the data records. However, for a different ZIP code such as 90210, a different level of hierarchy may be selected than for the 10547 value, as long as it is done consistently for a *particular* data value (e.g., 10547 or 90210) across all data records. In local recoding, different data records may use different generalizations for the same data value. For example, one data record might use *Northeastern US*, whereas another data record might use *New York* for 10547. While local recoding might *seem* to be better optimized, because of its greater flexibility, it does lose a different kind of information. In particular, because the same ZIP code might map to different values, such as *New York* and *Northeastern US*, the similarity computation

between the resulting data records may be less accurate. Most of the current privacy schemes use global recoding.

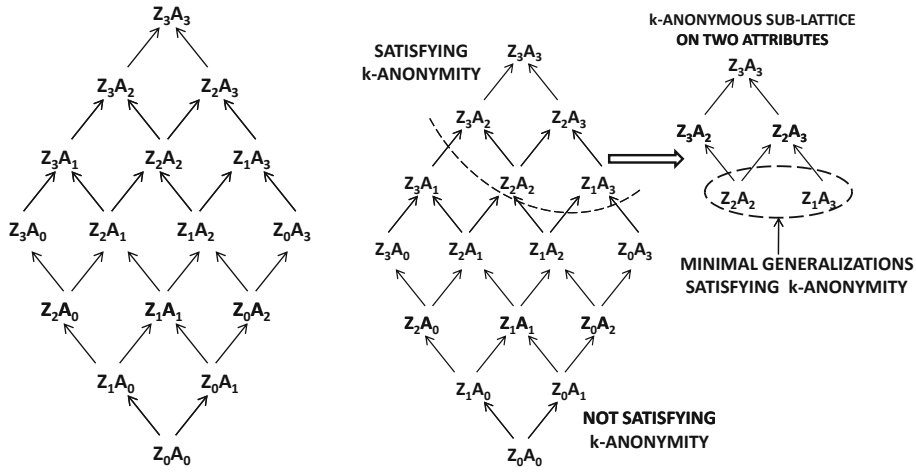
- *Full-domain generalization:* Full-domain generalization is a special case of global recoding. In this approach, all values of a particular attribute are generalized to the same level of the taxonomy. For example, a ZIP code might be generalized to its state for all instances of the attribute. In other words, if the ZIP code 10547 is generalized to *New York*, then the ZIP code 90210 must be generalized to *California*. The various hierarchical alternatives for full-domain generalization of the age attribute are denoted by A_0, A_1, A_2, and A_3 in Fig. 20.1. The possible full-domain generalization levels of the ZIP code are denoted by Z_0, Z_1, Z_2, and Z_3. In this case, Z_3 represents the highest level of generalization (column suppression), and Z_0 represents the original values of the ZIP code attribute. Thus, once it is decided that the anonymization algorithm should use Z_2 for the ZIP code attribute, then *every instance of the ZIP code attribute (Z_0) in the data set is replaced with its generalized value in Z_2.* This is the reason that the approach is referred to as full-domain generalization, as the entire domain of data values for a particular attribute is generalized to the same level of the hierarchy. Full-domain generalization is the most common approach used in privacy-preserving data publishing.

Full-domain generalization is intuitively appealing because it ensures that the different values of an attribute have the same level of granularity throughout the data set. The earliest methods, such as Samarati's original algorithm, and *Incognito*, were all full-domain generalization algorithms.

20.3.1.1 Samarati's Algorithm

Samarati's algorithm was first proposed in the context of the definition of k-anonymity. Samarati's original AG-TS (*Attribute Generalization and Tuple Suppression*) algorithm for k-anonymity provides the basic domain generalization framework, which is the basis for group-based anonymization. It has already been discussed, how the domain generalization of a single attribute can be represented as a path. For example, the path from Z_0 to Z_3 in Fig. 20.1 represents the generalization of the ZIP code attribute. The notion of domain generalization can also be defined for *combinations* of attributes. However, in the case of attribute combinations, the relationships are no longer expressed as a path, but as a special kind of directed acyclic graph, known as a lattice. In this case, each node specifies a (full-domain) generalization level for the different attributes For example, $< A_1, Z_2 >$ denotes the domain generalization level of age to A_1 and ZIP code to Z_2. In other words, every data record is generalized to the level $< A_1, Z_2 >$. Note that $< A_1, Z_2 >$ also represents the generalization level of the (anonymized) Table 20.3 based on the domain-generalization hierarchies specified in Fig. 20.1.

Thus, each node in the lattice specifies a possible level of full-domain generalization, in terms of which the original data is represented. The edges in this graph represent the *direct* generalization relationships among these tuples of domains. A directed path in the lattice, from lower to higher levels, represents a sequence of generalizations. Conversely, a lower-level node is a specialization of a higher-level node. For example, the node $< A_1, Z_1 >$ is a direct specialization of either $< A_1, Z_2 >$, or $< A_2, Z_1 >$ because a single attribute in either can be specialized once to immediately yield $< A_1, Z_1 >$. An example of the domain generalization hierarchy for the age and ZIP code combination is illustrated in Fig. 20.3a. *The goal of the full-domain anonymization algorithm is to discover the node $< A_i, Z_j >$ in*



(a) 2-attribute lattice (b) k-anonymous portion

Figure 20.3: Domain generalization hierarchies over combinations of attributes

this tuple-based domain generalization hierarchy that preserves k-anonymity with the least amount of generalization. After such a node $< A_i, Z_j >$ has been discovered, the privacy algorithm generalizes all ages to the level A_i and all ZIP codes to the level Z_j.

In practice, some of the tuples may need to be suppressed in order to prevent undesirably high levels of generalization. This is because these may represent outlier tuples that cannot be incorporated in any group without significantly increasing the generalization level. For example, an individual with an age of 125 may need to be suppressed because of the outlier value of this attribute. Therefore, one of the parameters to the algorithm is a threshold $MaxSup$, which specifies the maximum number of tuples that can be suppressed. The goal is therefore to discover a node that is as low as possible in the lattice of Fig. 20.3a, such that k-anonymity is satisfied after suppressing at most $MaxSup$ tuples. The *height* of a node in the lattice is defined as its path distance in the lattice from the most specific level of representation. In the example of Fig. 20.3, the height of node $< Z_i, A_j >$ is $(i + j)$. A *minimally generalized node* may be defined as a node, for which the height is as small as possible. Therefore, in this example, one way of determining minimal generalizations, is to discover a k-anonymizable node $< Z_i, A_j >$, such that the height $(i + j)$ is as small as possible.

When there are d attributes $< Q_{i_1} \ldots Q_{i_d} >$, the sum $\sum_{k=1}^{d} i_k$ over all attributes represents the height of that particular combination of generalizations. It is easy to see that any specialization of a node $< Q_{i_1} \ldots Q_{i_d} >$ that does not satisfy k-anonymity will also not satisfy k-anonymity. Similarly, any generalization of a node satisfying k-anonymity will also satisfy k-anonymity. Therefore, the subgraph of the lattice satisfying k-anonymity and the subgraph violating k-anonymity are both connected subgraphs, and a border can be constructed between them. An example of such a border[1] is illustrated in Fig. 20.3b, and the corresponding minimal generalizations are illustrated in the same figure. Note that the minimal generalization is not unique, and that two possible minimal generalizations $< Z_2, A_2 >$ and $< Z_1, A_3 >$ are possible in this example. The reason for using minimally generalized nodes is to maximize the utility of the data for analytical algorithms. Other

[1]This border is for illustration purposes only, and does not correspond to any data set in this chapter.

more refined definitions can be used for quantifying utility that use the distribution of the attribute values more explicitly. The bibliographic notes contain pointers to some of these definitions.

Samarati's algorithm uses a simple binary search over the lattice of domain generalization tuples. Let $[0, h_{max}]$ represent the range of heights of the lattice. It is then checked whether any of the generalizations at level $h_{max}/2$ satisfies the k-anonymity constraint. If this is indeed the case, then the height $h_{max}/4$ is checked. Otherwise, the height $3 \cdot h_{max}/4$ is checked. This approach is repeated, until the lowest height at which a k-anonymous solution exists, is found. All the corresponding domain generalizations are reported, and any of these can be used for transforming the data. An important step in Samarati's algorithm is the process of using the original database to check whether a particular node in the lattice satisfies k-anonymity. However, a discussion of this step is omitted here, because similar steps are discussed below in the context of the *Incognito* algorithm.

20.3.1.2 Incognito

The lattice of Fig. 20.3 shares a number of conceptual similarities with the lattice of frequent itemset mining algorithms, as discussed in Chap. 4. Therefore, some of the anonymization algorithms for discovering full-domain generalization also have similar characteristics to those of frequent itemset mining algorithms. The *Incognito* algorithm leverages a number of principles from frequent pattern mining to efficiently discover the k-anonymous portion of the lattice.

An important observation is that the size of the lattice is exponentially related to the number of quasi-identifiers. This can lead to increasing computational complexity in many practical scenarios. While it has been shown by Meyerson and Williams [385] that optimal k-anonymization is NP-hard, it is possible to reduce the computational burden by careful exploration of the lattice. The *Incognito* algorithm is based on the observation that the k-anonymity of a subset of generalized attributes is a necessary (but not sufficient) condition for the k-anonymity of a superset of attributes with matching generalization levels of the common elements. Henceforth, this property will be referred to as *attribute subset closure*. This property is a specific case of the *generalization property* which states that any generalization of a k-anonymous node in the lattice will always be k-anonymous.

These properties can be used to both generate candidates and prune the search process in a manner that is similar to the *Apriori* algorithm for frequent itemset mining. Therefore, nodes that are not k-anonymous with respect to a set of attributes, can be discarded, together with their specializations in the lattice hierarchy. Furthermore, generalizations of subsets of attributes that do satisfy the k-anonymity constraint, do not need to be checked because they are guaranteed to be k-anonymous.

The *Incognito* approach uses a levelwise approach, in which the following steps are repeated iteratively, until the k-anonymous sublattice containing all d attributes has been constructed. The set $\mathcal{F}_i$ denotes the set of all sublattices on i attributes that satisfies k-anonymity. The algorithm starts by initializing $\mathcal{F}_1$ to the portions of the single-attribute domain generalization hierarchies satisfying k-anonymity. This is quite simple, because single attribute hierarchies are paths. Thus, $\mathcal{F}_1$ is simply the top portion of the path, such that each generalized attribute value contains at least k tuples. Subsequently, as in frequent pattern mining, the algorithm repeatedly generates candidate sublattices in $\mathcal{C}_{i+1}$ by joining sublattices in $\mathcal{F}_i$ that have exactly $(i-1)$ attributes in common. The process of joining two sublattices will be described later. Note that $\mathcal{C}_{i+1}$ is a set of candidate sublattices on $(i+1)$ attributes. Each of these sublattices is then pruned of some of its nodes, using an

Apriori-style approach. Specifically, nodes of sublattices in $\mathcal{C}_{i+1}$ whose generalizations are not k-anonymous in $\mathcal{F}_i$ can be pruned. This step will also be described in detail later.

After candidate generation and pruning, the portion of each sublattice that satisfies k-anonymity is retained by checking the constituent nodes against the base data records. Thus, each sublattice in $\mathcal{C}_{i+1}$ reduces further in size. At this point, the set $\mathcal{C}_{i+1}$ has been transformed to the set $\mathcal{F}_{i+1}$. Thus, the following steps are repeated for increasing values of the index i:

1. Generate $\mathcal{C}_{i+1}$, the set of *candidate* sublattices on $(i+1)$ attributes. This is achieved by joining all pairs of k-anonymous sublattices in $\mathcal{F}_i$ that share $(i-1)$ attributes. The details of a join between a pair of sublattices will be described later.

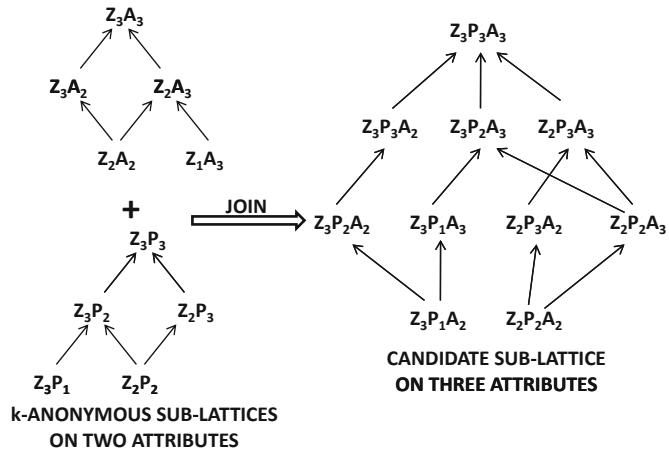
2. Prune the nodes from each sublattice in $\mathcal{C}_{i+1}$ that cannot possibly satisfy k-anonymity by using the *attribute subset closure* property with respect to the set of k-anonymous combinations in $\mathcal{F}_i$. The details of how the nodes may be pruned from a sublattice, will be described later.

3. Check each node in each (already pruned) sublattice of $\mathcal{C}_{i+1}$ against the base data, and remove those that do not satisfy k-anonymity. A node does not need to be checked, if one of its specializations already satisfies k-anonymity. This step transforms the set of candidate sublattices $\mathcal{C}_{i+1}$ to the set of k-anonymous sublattices $\mathcal{F}_{i+1}$ by removing the anonymity-violating sublattices.

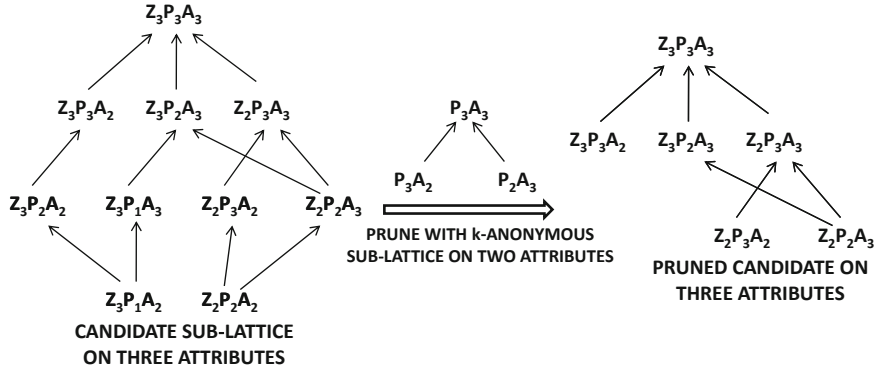
If there are a total of d attributes, then the set $\mathcal{F}_d$ will contain a single sublattice of nodes satisfying k-anonymity. The nodes with the smallest height in this sublattice are reported. Note that the detailed implementation of the *Incognito* algorithm uses a slightly different approach for actually tracking the sublattices, by tracking the lattice nodes and edges in separate tables. The i-dimensional tables containing the generalization levels of lattice nodes of $\mathcal{F}_i$ are joined on their $(i-1)$ common attributes to create the $(i+1)$-dimensional tables containing the nodes of $\mathcal{C}_{i+1}$. Subsequently, the lattice edges are added between the generated nodes based on the hierarchy relationships. Nevertheless, the simpler logical description provided here matches the *Incognito* algorithm.

Next, the details of the join and pruning operations will be discussed with the use of an example. In this case, three attributes will be used for greater clarity. As discussed earlier, let A_r and Z_r represent different generalization levels of the age and ZIP code attributes, for varying values of the index r. Let P_r represent the generalization levels of an additional attribute corresponding to the profession. Higher values of the index r indicate a greater level of generalization. Consider the scenario where all three k-anonymous two-attribute sublattices on these three attributes are already available in $\mathcal{F}_2$. It is possible to use any pair of sublattices from these three possibilities, in order to perform the join. This will result in a *candidate* sublattice on all three attributes.

Consider the case, where the sublattices on (ZIP code, Age) and (ZIP code, Profession) are joined. The nodes in the new candidate sublattice will now have three attributes (ZIP code, Profession, Age) instead of two. The nodes for the new candidate sublattice are constructed by joining the nodes of the two k-anonymous sublattices. A pair of nodes $< Z_r, A_j >$ and $< Z_s, P_l >$ will be joined, if and only if $r = s$. In other words, the generalization level of the ZIP code attribute needs to be the same in both cases. This will result in the new node $< Z_r, P_l, A_j >$. In general, for pairs of nodes with k attributes, a join will be successfully executed, if and only if (a) they share $(k-1)$ attributes, and (b) the generalization levels of the $(k-1)$ common attributes are the same. An example of a join with two k-anonymous sublattices is illustrated in Fig. 20.4a.



(a) *Incognito* join between two sub-lattices



(b) *Incognito* pruning

Figure 20.4: *Incognito* joins and pruning

In the previous example, the sublattice for the profession–age combination was not used for the join. However, it is still useful for pruning. This is because, if a node $< P_i, A_j >$ is not present in this sublattice, then any node of the form $< Z_m, P_i, A_j >$ will also not be k-anonymous. Therefore, such nodes can be removed from the constructed candidate sublattice together with their specializations. An example of a pruning step on the candidate sublattice is illustrated in Fig. 20.4b. This pruning is based on the attribute-subset closure property, and it is reminiscent of *Apriori* pruning in frequent itemset mining. As in the case of frequent itemset mining, all k-attribute subsets of each candidate $(k+1)$-sublattice in $\mathcal{C}_{k+1}$ need to be checked. If a node violates the closure property in any of these checks, then it is pruned.

Finally, the generated nodes in $\mathcal{C}_{k+1}$ need to checked against the original database to determine whether they satisfy k-anonymity. For example, in order to determine whether $< Z_1, A_1 >$ satisfies k-anonymity based on the value generalization in Fig. 20.1, one needs to determine the number of individuals satisfying each of the pairs of conditions such as (ZIP code $\in$ NY, $0 <$ Age ≤ 10), (ZIP code $\in$ NY, $10 <$ Age ≤ 20), (ZIP code $\in$ MA, $0 <$ Age ≤ 10), and so on. Therefore, for each node in the lattice,

a vector of frequency values need to be computed. This vector is also referred to as a *frequency vector* or *frequency set*. The process of frequency vector computation can be expensive because the original database may need to be scanned to determine the number of tuples satisfying these conditions. However, several strategies can be used to reduce the burden of computation. For example, if the frequency vector of $< Z_1, A_1 >$ has already been computed, one can use *roll-up* to directly compute the frequency vectors of the generalization $< Z_2, A_1 >$ without actually scanning the database. This is because the frequency of the set (ZIP code $\in$ Northeastern US, $0 <$ Age ≤ 10) is the sum of the frequencies of (ZIP code $\in$ NY, $0 <$ Age ≤ 10), (ZIP code $\in$ NJ, $0 <$ Age ≤ 10), (ZIP code $\in$ MA, $0 <$ Age ≤ 10), and so on. The simplest approach is to use a breadth-first strategy on the lattice of each set of $(k+1)$ attributes, by determining the frequency vectors of specific (lower-level) nodes in the lattice before determining the frequency vectors of more general (higher-level) nodes. The frequency vectors of higher-level nodes can be computed efficiently from those of lower-level nodes by using the roll-up property.

Note that a separate breadth-first search needs to be performed for each subset of $(k+1)$ attributes in $\mathcal{C}_{k+1}$ to compute its frequency vectors. Furthermore, once a node has been identified by the breadth-first search to be k-anonymous, its generalizations in the lattice are guaranteed to be k-anonymous. Therefore, they are automatically marked as k-anonymous and are not explicitly checked. The original algorithm also supports a number of other optimizations, referred to as *Incognito super-roots* and *Bottom-up precomputation*. The bibliographic notes contain pointers to these methods.

20.3.1.3 Mondrian Multidimensional k-Anonymity

One of the disadvantages of the methods discussed so far is that the domain generalization hierarchies for various attributes are constructed independently as a preprocessing step. Thus, after the hierarchical discretization (domain generalization) for a numeric attribute has been fixed by the preprocessing step, it is utilized by the anonymization algorithm. This rigidity in the anonymization process creates inefficiencies in data representation, when the various data attributes are correlated in multidimensional space. For example, the salary distribution for older individuals may be different from that of younger individuals. A preprocessed domain generalization hierarchy is unable to adjust to such attribute correlations in the data set. In general, the best trade-offs between privacy and utility are achieved when the multidimensional relationships among data points are leveraged in the anonymization process. In other words, the attribute ranges for each attribute in a data point $\overline{X}$ should be generated in a dynamic way depending on the specific *multidimensional* locality of $\overline{X}$.

The *Mondrian* method generates multidimensional rectangular regions, containing at least k data points. This is achieved by recursively dividing the bounding boxes with axis-parallel cuts, until each region contains no more than k data points. This approach is not very different from the methodology used by many traditional index structures, such as kd-trees. An example of the partitioning induced by the *Mondrian* algorithm is illustrated in Fig. 20.5. In this case, a 5-anonymous partitioning is illustrated. Thus, each group contains *at least* five data points. It is easy to see that the same attribute value is represented by different ranges in different portions of the data, in order to account for the varying density of different regions. It is this flexibility that gives *Mondrian* a more compact representation of the anonymized groups than the other methods.

The *Mondrian* algorithm dynamically maintains the set $\mathcal{B}$ of multidimensional generalizations that satisfy k-anonymity and cover the data set. The *Mondrian* algorithm starts with a rectangular box B of all the data points. This represents the generalization

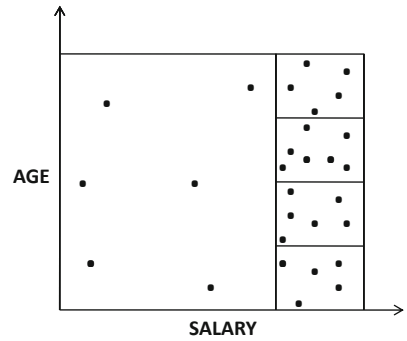


Figure 20.5: A sample 5-anonymous *Mondrian* multidimensional partitioning

of the entire data set to a single multidimensional region, and therefore trivially satisfies k-anonymity. The algorithm therefore starts by initializing $\mathcal{B} = \{B\}$. The algorithm repeatedly uses the following steps:

1. Select a rectangular region $R \in \mathcal{B}$ containing at least $2 \cdot k$ data points, such that a valid split into a pair of k-anonymous subsets exists.

2. Split the rectangular region R along any of the dimensions with an axis-parallel split, so that each of R_1 and R_2 contains at least k data points.

3. Update $\mathcal{B} \Leftarrow \mathcal{B} \cup \{R_1, R_2\} - R$

This iterative process is repeated, until the rectangular regions cannot be split any further without violating k-anonymity. There is some flexibility in the choice of the dimension for performing the split. A natural heuristic is to split the longest dimension of the selected rectangular region. After the dimension has been selected, the split should be performed so that the data points are partitioned as evenly as possible. In the absence of ties on attribute values, the data points can be divided almost equally into the two regions.

The rectangular regions in $\mathcal{B}$ define the equivalence classes that are utilized for k-anonymization. If each numeric attribute value is *unique*, it can be shown that every region will contain at most $2 \cdot k - 1$ data points. However, if there are ties among attribute values, and tied values need to be assigned to be the same partition, then an upper bound of $m + 2d \cdot (k-1)$ can be shown on the number of data points in each partition. Here m is the number of identical copies of any data record. On the other hand, if ties on an attribute value can be flexibly assigned to any partition, then the maximum number of points in any rectangular partition, at the end of the process will be $2 \cdot k - 1$. The reader is referred to the bibliographic notes for the pointer to the proof of this bound. After the data has been divided into rectangular regions, the following approaches can be used for reporting the anonymized data points:

1. The averages along each dimension may be reported for each anonymized equivalence set.

2. The multidimensional bounding box of the data points may be reported.

The *Mondrian* algorithm has been shown to be more effective than the *Incognito* algorithm, because of the greater flexibility provided by the multidimensional approach to partitioning.

The *Mondrian* approach is naturally designed for numeric attributes with an ordering on the values. However, the approach can also be generalized to categorical attributes by designing appropriate split rules for the attributes.

20.3.1.4 Synthetic Data Generation: Condensation-Based Approach

The condensation-based approach generates synthetic data that matches the original data distribution, while maintaining k-anonymity. This means that k synthetic records are generated for each group of k records, by using the statistics of that group. The overall condensation approach may be described as follows:

1. Use any clustering approach to partition the data into groups of data records, such that each group contains at least k data records. Denote the number of created groups by m.

2. Compute the mean and covariance matrix for each group of data records. For a d-dimensional data set, the covariance matrix of a group represents the $d \times d$ covariances between pairs of attributes.

3. Compute the eigenvectors and eigenvalues of each covariance matrix. It is evident from the discussion of Principal Component Analysis (PCA) in Chap. 2, that the eigenvectors define a group-specific axis system, along which the data records are uncorrelated. The variance of the data along each eigenvector is equal to the corresponding eigenvalue. The synthetic data set to be generated, is modeled as mixture of m clusters, where the mean of each cluster is the mean of the corresponding group of original data records.

4. Generate synthetic data records for each of the m clusters. For each cluster, the number and mean of the synthetic records matches its base group. Data records are generated independently along the eigenvectors, with variance equal to the corresponding eigenvalues. The uniform distribution is typically used for synthetic data generation, because it is assumed that the data distribution does not change significantly within the small locality defined by a group. While the uniform distribution is a *local* approximation, the *global* distribution of the generated records generally matches the original data quite well.

The approach can also be generalized to data streams, by maintaining group statistics incrementally. The idea here is that group sizes are allowed to vary between k and $2 \cdot k - 1$. Whenever a group reaches the size of $2 \cdot k$, they are split into two groups. The details of group splitting will be discussed later.

To maintain the covariance statistics incrementally in the streaming scenario, an approach similar to the cluster-feature vector of *CluStream* (see Chap. 7) is used. The only difference is that the product-wise sum statistics are also maintained incrementally. For any pair of attributes i and j, the value of $\text{Sum}(i, j)$ is equal to sum of the product of attribute values i and j over the different data points. This can be easily maintained incrementally in a data stream. Then, for a set of $r \in (k, 2 \cdot k - 1)$ data points in a group, the covariance between attributes i and j may be estimated as follows:

$$\text{Covariance}(i, j) = \text{Sum}(i, j)/r - \text{Mean}(i) \cdot \text{Mean}(j) \tag{20.6}$$

$$\text{Covariance}(i, j) = \text{Sum}(i, j)/r - \text{Sum}(i) \cdot \text{Sum}(j)/r^2 \tag{20.7}$$

All the statistics in the aforementioned equation are additive, and can easily be maintained incrementally in the stream setting.

It remains to be explained how the groups are split, once the group sizes reach $2 \cdot k$. It is assumed that each group of size $2 \cdot k$ is split into two groups of size k along the longest eigenvector. The reason for choosing the longest eigenvector is to ensure the compactness of the newly created groups. The splitting of groups can be a challenge, because the original data records are not available in the streaming scenario to recalculate the statistics of each of the split groups. Therefore, an approximation (i.e., modeling assumption) is needed. The condensation approach works with the modeling assumption that the data records of a group are independently distributed along each eigenvector according to a uniform distribution. For group sizes that are much smaller than the number of points in the data set, this is not an unreasonable assumption. This is because density distributions do not change drastically over small regions of the data.

This modeling assumption of a uniform distribution is used to re-calculate the new means of each of the child groups of equal size k. This is because the range of the uniform distribution along the longest eigenvector can be approximated from its variance (eigenvalue), based on the modeling assumption. Note that the variance of a uniform distribution is one twelfth the square of its range. Therefore, if λ_{max} be the largest eigenvalue, then the range R of the uniform distribution is computed as follows:

$$R = \sqrt{12\lambda_{max}} \tag{20.8}$$

This range R is then split into two equal parts to create the two new group means. Thus, the two new group means are at a distance of $R/4$ from the old group mean in opposite directions along the longest eigenvector.

The newly created groups are assumed to have the same eigenvectors as the parent group, because the splitting is performed along an uncorrelated direction. Therefore, the directions of correlation are not assumed to change after splitting. The largest eigenvalue of the original (parent) group is replaced by an eigenvalue in each of the child groups, which is one fourth[2] the original value. Thus, if P is the $d \times d$ matrix with orthonormal columns containing the eigenvectors, and Σ is the diagonal matrix of eigenvalues (after adjustment of the largest eigenvalue), then the covariance matrix of the newly created split groups can be computed as follows:

$$C = P\Sigma P^T \tag{20.9}$$

This relationship is based on the standard PCA diagonalization discussed in Chap. 2. Note that the covariance matrices of both the split groups are the same. The covariance matrices and newly generated group means can be used to back-calculate the sum of pairwise attribute products of each group according to Eq. 20.6. Thus, as more data points arrive, these product values can continue to be updated incrementally.

The condensation-based approach is one of the few methods that can be applied to data streams with a relatively low risk of disclosure, because of its approach of using synthetic data. It is often difficult for an adversary to know which group of k synthetic records was generated from a particular base group of original records. In the case of generalization-based anonymization, it is relatively easy to identify groups of related data records, representing equivalence classes. Thus, synthetic data sets provide some additional privacy protection. Note that it is possible to generate larger data sets using this approach if needed. For example, for each group of k records, one might generate $\alpha \cdot k$ synthetic data records,

[2]Splitting a uniform distribution into two equal parts reduces its variance by a factor of 4.

using the statistics of that group. This scales up the size of the data with a factor of α, and further reduces the mapping between the generated data and the original data. Furthermore, additional noise can be incorporated during synthetic data generation to ensure greater protection.

These additional options do come at a price. The *truthfulness* of the published data is lost. The published data records are *synthetic* and therefore do not map onto any particular individual. In many aggregation- or modeling-based applications, this is not necessarily an issue, because the aggregate properties of the data are retained. In some medical data handling scenarios, legal restrictions may prohibit release of downgraded data, when there is a direct mapping between individuals and data records, even at a group level. The condensation approach provides a solution in some of these scenarios, because the released data records are synthetic, and are generally difficult to map onto specific groups.

The condensation approach shares a number of conceptual similarities with the *Mondrian* approach, except that it allows the use of any constrained clustering algorithm, rather than rectangular partitions constructed with single dimensional cuts. The utility of the resulting anonymization depends on the effectiveness of the clustering. Single dimensional cuts will not be able to construct high-quality clusters with increasing dimensionality. Furthermore, unlike *Mondrian*, synthetic data is generated to achieve greater anonymity.

The condensation approach does not distinguish between publicly available attributes (used in combination to construct quasi-identifiers) and sensitive attributes, and applies the approach to all the attributes. As will be evident from the subsequent discussion on the dimensionality curse in Sect. 20.3.4, the distinction between quasi-identifier and sensitive attributes is more fluid, than is often assumed in the literature on data privacy. Because it is not possible to know the level of background knowledge available to adversaries about the sensitive attributes, all attributes should be perturbed. When the sensitive attributes are released without any perturbation, they become immediately available for identification attacks, as long as background knowledge is available. For example, a number of privacy attacks on data sets such as the Netflix data set [402], have been performed using attributes that would normally not have been considered publicly available. This work [402] also makes the argument that such strong distinctions between publicly available and sensitive attributes are dangerous to make in real-world settings where the data and background knowledge available to the public continues to increase over time.

20.3.2 The ℓ-Diversity Model

While the k-anonymity model provides the basic framework for privacy-preserving data publishing, there are scenarios in which it can lead to inadvertent sensitive attribute disclosure. Consider the 3-anonymized table illustrated in Table 20.3. In this case, the row indices 1, 3, and 6 are in the same anonymized group, and cannot be distinguished from one another. However, all three individuals have the value of "HIV" on the sensitive attribute. Therefore, even though the *identity* of the specific individual from this group cannot be inferred, it can be inferred that any individual in this group has HIV. Therefore, if a voter registration roll is used to join this group to three unique individuals, then it can be inferred that all three of them have HIV. This represents a breach of sensitive *attribute* information about each of these three individuals. In other words, while the k-anonymity model prevents *identity disclosure*, it does not prevent *attribute disclosure*.

The main reason for this breach is that the sensitive information is not diverse enough within the anonymized groups. Since the goal of privacy-preserving data publishing is to prevent the revelation of sensitive information, a model that does not use the sensitive

attribute values within the group formation process, cannot achieve this goal. The ℓ-diversity model is designed to ensure that the sensitive attributes within an equivalence class are sufficiently diverse.

Definition 20.3.2 (ℓ-diversity Principle) *An equivalence class is said to be ℓ diverse, if it contains ℓ "well-represented" values for the sensitive attribute. An anonymized table is said to be ℓ-diverse, if each equivalence class in it is ℓ-diverse.*

It is important to note that the notion of "well represented" can be instantiated in several different ways. Therefore, the aforementioned definition provides the basic *principle* behind this approach, but cannot be considered a hard definition. There are several ways in which the notion of "well-represented" can be instantiated. These correspond to the notions of entropy ℓ-diversity and recursive ℓ-diversity. These definitions are described below.

Definition 20.3.3 (Entropy ℓ-diversity) *Let $p_1 \ldots p_r$ be the fraction of the data records belonging to different values of the sensitive attribute in an equivalence class. The equivalence class is said to be entropy ℓ-diverse, if the entropy of its sensitive attribute value distribution is at least $log(\ell)$.*

$$-\sum_{i=1}^{r} p_i \cdot log(p_i) \geq log(\ell) \tag{20.10}$$

An anonymized table is said to satisfy entropy ℓ-diversity, if each equivalence class in it satisfies entropy ℓ-diversity.

It can be shown that the sensitive attributes in an equivalence class must have at least ℓ distinct values for the table to be ℓ-diverse (see Exercise 7). Therefore, any ℓ-diverse group has at least ℓ elements, and is ℓ-anonymous as well.

One problem with this definition of ℓ-diversity is that it may be too restrictive in many settings, especially when the distributions of the sensitive attribute values are uneven. The entropy of a table can be shown to be at least equal to the minimum entropy of the constituent equivalence classes into which it is partitioned (see Exercise 8). Therefore, to ensure ℓ-diversity of each equivalence class, the sensitive attribute distribution in the entire table must also be ℓ-diverse. This is a restrictive assumption in many settings, because most real distributions of sensitive attributes are very skewed. For example, in a medical application, the sensitive (disease) attribute is likely to have uneven frequencies between normal individuals and various diseases. Greater attribute skew reduces the (global) entropy ℓ-diversity of the sensitive-attribute distribution across the entire table. When this global ℓ-diversity is less than ℓ, it is no longer possible to create a globally ℓ-diverse partition without suppressing many data records.

Therefore, a more relaxed notion of recursive (c, ℓ)-diversity has been proposed. The basic goal of the definition is to ensure that the most frequent attribute value in an equivalence class does not dominate the less frequent sensitive values in it. An additional parameter c is used to control the relative frequency of the different values of the sensitive attribute within an equivalence class.

Definition 20.3.4 (Recursive (c, ℓ)-diversity) *Let $p_1 \ldots p_r$ be the fraction of the data records belonging to the r different values of the sensitive attribute in an equivalence class, such that $p_1 \geq p_2 \geq \ldots \geq p_r$. The equivalence class satisfies recursive (c, ℓ)-diversity, if the following is true:*

$$p_1 < c \cdot \sum_{i=\ell}^{r} p_i \tag{20.11}$$

An anonymized table is said to satisfy recursive (c, ℓ)*-diversity, if each equivalence class in it satisfies entropy* (c, ℓ)*-diversity.*

The idea is that the least frequent tail of the sensitive attribute values must contain sufficient cumulative frequency compared to the most frequent sensitive attribute value. The value of r has to be at least ℓ, for the right-hand side of the aforementioned relationship to be non-zero.

A key property of ℓ-diversity is that any generalization of an ℓ-diverse table is also ℓ-diverse. This is true for both definitions of ℓ-diversity.

Lemma 20.3.1 (Entropy ℓ-diversity monotonicity) *If a table is entropy* ℓ*-diverse, then any generalization of the table is entropy* ℓ*-diverse as well.*

Lemma 20.3.2 (Recursive (c, ℓ)**-diversity monotonicity)** *If a table is recursive* (c, ℓ)*-diverse, then any generalization of the table is recursive* (c, ℓ)*-diverse as well.*

The reader is advised to work out Exercises 9(a) and (b), which are related to these results. Thus, ℓ-diversity exhibits the same monotonicity property exhibited by k-anonymity algorithms. This implies that the algorithms for k-anonymity can be easily generalized to ℓ-diversity by making minor modifications. For example, both Samarati's algorithm and the *Incognito* algorithm can be adapted to the ℓ-diversity definition. The only change to any k-anonymity algorithm is as follows. Every time a table is tested for k-anonymity, it is now tested for ℓ-diversity instead. Therefore, algorithmic development of ℓ-diverse anonymization methods is typically executed by simply adapting existing k-anonymization algorithms.

20.3.3 The t-closeness Model

While the ℓ-diversity model is effective in preventing direct inference of sensitive attributes, it does not fully prevent the gain of *some* knowledge by an adversary. The primary reason for this is that ℓ-diversity does not account for the distribution of the sensitive attribute values in the original table. For example, the entropy of a set of sensitive attribute values with relative frequencies $p_1 \ldots p_r$ will take on the maximum value when $p_1 = p_2 = \ldots = p_r = 1/r$. Unfortunately, this can often represent a serious breach of privacy, when there is a significant skew in the original distribution of sensitive attribute values. Consider the example of a medical database of HIV tests, where the sensitive value takes on the two values of "HIV" or "normal," with relative proportions of 1 : 99. In this case, a group with an equal distribution of HIV and normal patients will have the highest entropy, based on the ℓ-diversity definition.

Unfortunately, such a distribution is highly revealing when the distribution of the sensitive values in the *original data* is taken into account. Sensitive values are usually distributed in a skewed way, across most real data sets. In the medical example discussed above, it is already known that only 1 % of the patients in the *entire data set* have HIV. Thus, the equal distribution of HIV-infected and normal patients within a group, provides a significant *information gain* to the adversary. *The adversary now knows, that this small group of patients has a much higher expected chance of having HIV, than the base population.*

In this context, a notion of *Bayes optimal privacy* exists, which ensures that the additional posterior information gained after release of information is as small as possible. Unfortunately, the notion of Bayes optimal privacy is practically and computationally difficult to implement. The t-closeness model may be viewed as a practical and heuristic approach that attempts to achieve similar goals as the notion of Bayes optimal privacy. This is achieved by using the distance functions between distributions. Informally, the goal is to create an

anonymization, such that the distance between the sensitive attribute distributions of each anonymized group and the base data is bounded by a user-defined threshold.

Definition 20.3.5 (t-closeness Principle) *Let $\overline{P} = (p_1 \ldots p_r)$ be a vector representing the fraction of the data records belonging to the r different values of the sensitive attribute in an equivalence class. Let $\overline{Q} = (q_1 \ldots q_r)$ be the corresponding fractional distributions in the full data set. Then, the equivalence class is said to satisfy t-closeness, if the following is true, for an appropriately chosen distance function $Dist(\cdot, \cdot)$:*

$$Dist(\overline{P}, \overline{Q}) \leq t \tag{20.12}$$

An anonymized table is said to satisfy t-closeness, if all equivalence classes in it satisfy t-closeness.

The previous definition does not specify any particular distance function. There are many different ways to instantiate the distance function, depending on application-specific goals. Two common instantiations of the distance function are as follows:

1. *Variational distance:* This is simply equal to half the Manhattan distance between the two distribution vectors:

$$Dist(\overline{P}, \overline{Q}) = \frac{\sum_{i=1}^{r} |p_i - q_i|}{2} \tag{20.13}$$

2. *Kullback-Leibler (KL) distance:* This is an information-theoretic measure that computes the difference between the cross-entropy of $(\overline{P}, \overline{Q})$, and the entropy of $\overline{P}$.

$$Dist(\overline{P}, \overline{Q}) = \sum_{i=1}^{r} (p_i \cdot \log(p_i) - p_i \cdot \log(q_i)) \tag{20.14}$$

   Note that the entropy of the first distribution is $-\sum_{i=1}^{r} p_i \cdot \log(p_i)$, whereas the cross-entropy is $-\sum_{i=1}^{r} p_i \cdot \log(q_i)$.

While these are the two most common distance measures used, other distance measures can be used in the context of different application-specific goals.

For example, one may wish to prevent scenarios in which a particular equivalence class contains semantically related sensitive attribute values. Consider the scenario, where a particular equivalence class contains diseases such as gastric ulcer, gastritis, and stomach cancer. In such cases, if a group contains only these diseases, then it provides significant information about the sensitive attribute of that group. The t-closeness method prevents this scenario by changing the distance measure, and taking the distance between different *values* of the sensitive attribute into account in the distance-computation process. In particular, the *Earth Mover Distance* can be used effectively for this scenario.

The *earth mover's distance (EMD)* is defined in terms of the "work" (or *cost*) required to transform one distribution to the other, if we allow sensitive attribute values in the original data to be flipped. Obviously, it requires less "work" to flip a sensitive value to a semantically similar value. Formally, let d_{ij} be the amount of "work" required to transform the ith sensitive value to the jth sensitive value, and let f_{ij} be the fraction of data records which are flipped from attribute value i to attribute value j. The values of d_{ij} are provided by a domain expert. Note that there are many different ways to flip the distribution $(p_1 \ldots p_r)$ to the distribution $(q_1 \ldots q_r)$, and it is desired to use the least cost sequence of flips to

compute the distance between $\overline{P}$ and $\overline{Q}$. For example, one would rather flip "gastric ulcer" to "gastritis" rather than flipping "HIV" to "gastritis" because the former is likely to have lower cost. Therefore, f_{ij} is a variable in a linear programming optimization problem, which is constructed to minimize the overall cost of flips. For a table with r distinct sensitive attribute values, the cost of flips is given by $\sum_{i=1}^{r}\sum_{j=1}^{r} f_{ij} \cdot d_{ij}$. The earth mover's distance may be posed as an optimization problem that minimizes this objective function subject to constraints on the aggregate flips involving each sensitive attribute value. The constraints ensure that the aggregate flips do transform the distribution $\overline{P}$ to $\overline{Q}$.

$$\begin{aligned} Dist(\overline{P}, \overline{Q}) = \text{Minimize } & \sum_{i=1}^{r}\sum_{j=1}^{r} f_{ij} \cdot d_{ij} \\ & \text{subject to:} \\ & p_i - \sum_{j=1}^{r} f_{ij} + \sum_{j=1}^{r} f_{ji} = q_i \qquad \forall i \in \{1 \ldots r\} \\ & f_{ij} \geq 0 \qquad \forall i, j \in \{1, \ldots r\} \end{aligned}$$

The earth mover's distance has certain properties that simplify the computation of generalizations satisfying t-closeness.

Lemma 20.3.3 *Let E_1 and E_2 be two equivalence classes, and let $\overline{P_1}$, $\overline{P_2}$ be their sensitive attribute distributions. Let $\overline{P}$ be the distribution of $E_1 \cup E_2$, and $\overline{Q}$ be the global distribution of the full data set. Then, it can be shown that:*

$$Dist(\overline{P}, \overline{Q}) \leq \frac{|E_1|}{|E_1| + |E_2|} \cdot Dist(\overline{P_1}, \overline{Q}) + \frac{|E_2|}{|E_1| + |E_2|} \cdot Dist(\overline{P_2}, \overline{Q}) \tag{20.15}$$

This lemma is a result of the fact that the optimal objective function of a linear programming formulation is convex, and $\overline{P}$ can be expressed as a convex linear combination of $\overline{P_1}$ and $\overline{P_2}$ with coefficients $\frac{|E_1|}{|E_1|+|E_2|}$ and $\frac{|E_2|}{|E_1|+|E_2|}$, respectively. This convexity result also implies the following:

$$Dist(\overline{P}, \overline{Q}) \leq \max\{Dist(\overline{P_1}, \overline{Q}), Dist(\overline{P_2}, \overline{Q})\}$$

Therefore, when two equivalence classes satisfying t-closeness are merged, the merged equivalence class will also satisfy t-closeness. This implies the monotonicity property for t-closeness.

Lemma 20.3.4 (t-closeness monotonicity) *If a table satisfies t-closeness, then any generalization of the table satisfies t-closeness as well.*

The proof of this lemma follows from the fact that the generalization A of any table B, contains equivalence classes that are the union of equivalence classes in B. If each equivalence class in B already satisfies t-closeness, then the corresponding union of these equivalence classes must satisfy t-closeness. Therefore, the generalized table must also satisfy t-closeness. This monotonicity property implies that all existing algorithms for k-anonymity can be directly used for t-closeness. The k-anonymity test is replaced with a test for t-closeness.

20.3.4 The Curse of Dimensionality

As discussed at various places in this book, the curse of dimensionality causes challenges for many data mining problems. Privacy preservation is also one of the problems affected by the curse of dimensionality. There are two primary ways in which the curse of dimensionality impacts the effectiveness of anonymization algorithms:

1. *Computational challenges:* It has been shown [385], that optimal k-anonymization is NP-hard. This implies that with increasing dimensionality, it becomes more difficult to perform privacy preservation. The NP-hardness result also applies to the ℓ-diversity and t-closeness models, using a very similar argument.

2. *Qualitative challenges:* The qualitative challenges to privacy preservation are even more fundamental. Recently, it has been shown that it may be difficult to perform effective privacy preservation without losing the utility of the anonymized data records. This is an even more fundamental challenge, because it makes the privacy-preservation process less practical. The discussion of this section will be centered on this issue.

In the following, a discussion of the qualitative impact of the dimensionality curse on group-based anonymization methods will be provided. While a formal mathematical proof [10] is beyond the scope of this book, an intuitive version of the argument is presented. To understand why the curse of dimensionality increases the likelihood of breaches, one only needs to understand the well-known notion of *high dimensional data sparsity*. For ease in understanding, consider the case of numeric attributes. A generalized representation of a table can be considered a rectangular region in d-dimensional space, where d is the number of quasi-identifiers. Let $F_i \in (0, 1)$ be the fraction of the range of dimension i covered by a particular generalization. For the anonymized data set to be useful, the value of F_i should be as small as possible. However, the fractional volume of the space, covered by a generalization with fractional domain ranges of $F_1 \ldots F_d$, is given by $\prod_{i=1}^{d} F_i$. This fraction converges to 0 exponentially fast with increasing dimensionality d. As a result, the fraction of data points within the volume also reduces rapidly, especially if the correlations among the different dimensions are weak. For large enough values of d, it will be difficult to create d-dimensional regions containing at least k data points, unless the values of F_i are chosen to be close to 1. In such cases, any value of an attribute is generalized to almost the entire range of values. Such a highly generalized data set therefore loses its utility for data mining purposes. This general principle has also been shown to be true for other privacy models, such as perturbation, and ℓ-diversity. The bibliographic notes contain pointers to some of these theoretical results.

A real-world example of this scenario is the Netflix Prize data set, in which Netflix released ratings of individuals [559] for movies to facilitate the study of collaborative filtering algorithms. Many other sources of data could be found, such as the Internet Movie Database (IMDb), from which the ratings information could be matched with the Netflix prize data set. It was shown that the identity of users could be breached with a very high level of accuracy, as the number of ratings (specified dimensionality) increased [402]. Eventually, Netflix retracted the data set.

20.4 Output Privacy

The privacy-preservation process can be applied at any point in the data mining pipeline, starting with data collection, publishing, and finally, the actual application of the data mining process. The *output* of data mining algorithms can be very informative to an adversary. In particular, data mining algorithms with a large output and exhaustive data descriptions are particularly risky in the context of disclosure. For example, consider an association rule mining algorithm, in which the following rule is generated with high confidence:

$$(\text{Age} = 26, \text{ZIP Code} = 10562) \Rightarrow \text{HIV}$$

This association rule is detrimental to the privacy of an individual satisfying the condition on the left hand side of the aforementioned rule. Therefore, the discovery of this rule may result in the unforseen disclosure of private information about an individual. In general, many databases may have revealing relationships among subsets of attributes because of the constraints and strong statistical relationships between attribute values.

The problem of association rule hiding may be considered a variation of the problem of statistical disclosure control, or database inference control. In these problems, the goal is to prevent inference of sensitive values in the database from other related values. However, a crucial difference does exist between database inference control and association rule hiding. In database inference control, the focus is on hiding some of the entries, so that the privacy of other entries is preserved. In association rule hiding, the focus is on hiding the rules themselves, rather than the entries. Therefore, the privacy preservation process is applied on the output of the data mining algorithm, rather than the base data.

In association rule mining, a set of sensitive rules are specified by the system administrator. The task is to mine all association rules, such that none of the sensitive rules are discovered, but all nonsensitive rules are discovered. Association rule hiding methods are either *heuristic methods*, *border-based methods*, or *exact methods*. In the first class of methods, a subset of transactions are removed from the data. The association rules are discovered on the set of sanitized transactions. In general, if too many transactions are removed, then the remaining nonsensitive rules, which are discovered, will not reflect the true set of rules. This may lead to the discovery of rules that do not reflect the true patterns in the underlying data. In the case of *border-based methods*, the border of the frequent pattern mining algorithm is adjusted, so as to discover only nonsensitive rules. Note that when the borders of the frequent itemsets are adjusted, it will lead to the exclusion of nonsensitive rules along with the sensitive rules. The last class of problems formulates the hiding process as a constraint satisfaction problem. This formulation can be solved using integer programming. While these methods provide exact solutions, they are much slower, and their use is limited to problems of smaller size.

A related problem in output privacy is that of *query auditing*. In query auditing, the assumption is that users are allowed to issue a sequence of queries to the database. However, the response to one or more queries may sometimes lead to the compromising of sensitive information about smaller sets of individuals. Therefore, the responses to some of the queries are withheld (or *audited*) to prevent undesirable disclosure. The bibliographic notes contain specific pointers to a variety of query auditing and association rule hiding algorithms.

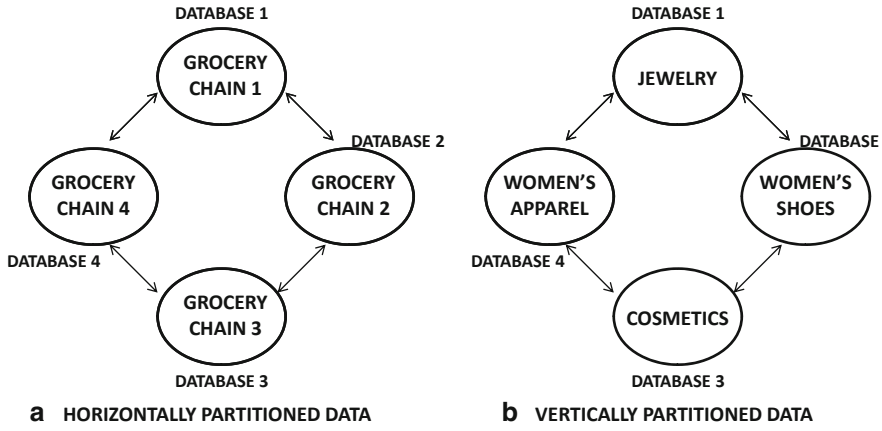


Figure 20.6: Examples of horizontally and vertically partitioned data

20.5 Distributed Privacy

In distributed privacy-preserving data mining, the goal is to mine shared insights across multiple participants owning different portions of the data, without compromising the privacy of local statistics or data records. The key is to understand that the different participants may be partially or fully adversaries/competitors, and may not wish to provide full access of their *local* data and statistics to one another. However, they might find it mutually beneficial to extract *global* insights over all the data owned by them.

The data may be partitioned either *horizontally* or *vertically* across the different participants. In horizontal partitioning, the data records owned by different adversaries have the same attributes, but different adversaries own different portions of the database. For example, a set of supermarket chains may own similar data related to customer buying behavior, but the different stores may show somewhat different patterns in their transactions because of factors specific to their particular business. In *vertical* partitioning, the different sites may contain different attributes for the same individual. For example, consider a scenario in which a database contains transactions by various customers. A particular customer may buy different kinds of items at stores containing complementary products such as jewelery, apparel, cosmetics, etc. In such cases, the aggregate association analysis across different participants can provide insights, that cannot be inferred from any particular database. Examples of horizontal and vertically partitioned data are provided in Figs. 20.6a and b, respectively.

At the most primitive level, the problem of distributed privacy-preserving data mining overlaps closely with a field in cryptography for determining secure multi-party computations. In this field, functions are computed over inputs provided by multiple recipients without actually sharing the inputs with one another. For example, in a two-party setting, Alice and Bob may have two inputs x and y, respectively, and may wish to compute the function $f(x, y)$ without revealing x or y to each other. This problem can also be generalized across k parties for computing the k argument function $h(x_1 \ldots x_k)$. Many data mining algorithms may be viewed in the context of repetitive computations of primitive functions such as the scalar dot product, secure sum, secure set union, etc. For example, the scalar dot product of the binary representation of an itemset and a transaction can be used to determine whether or not that itemset is supported by that transaction. Similarly, scalar

dot products can be used for similarity computations in clustering. To compute the function $f(x, y)$ or $h(x_1 \ldots, x_k)$, a *protocol* needs to be designed for exchanging information in such a way that the function is computed without compromising privacy.

A key building-block for many kinds of secure function evaluations is the 1 out of 2 oblivious-transfer protocol. This protocol involves two parties: a *sender*, and a *receiver*. The sender's input is a pair (x_0, x_1), and the receiver's input is a bit value $\sigma \in \{0, 1\}$. At the end of the process, the receiver learns x_σ only, and the sender learns nothing. In other words, the sender does not learn the value of σ.

In the oblivious transfer protocol, the sender generates two encryption keys, K_0 and K_1, but the protocol is able to ensure that the receiver knows only the decryption key for K_σ. The sender is able to generate these keys by using an encrypted input from the receiver, which encodes σ. This coded input does not reveal the value of σ to the sender, but is sufficient to generate K_0 and K_1. The sender encrypts x_0 with K_0, x_1 with K_1, and sends the encrypted data back to the receiver. At this point, the receiver can only decrypt x_σ, since this is the only input for which he or she has the decryption key. The 1 out of 2 oblivious transfer protocol has been generalized to the case of k out of N participants.

The oblivious transfer protocol is a basic building block, and can be used in order to compute several data mining primitives related to vector distances. Another important protocol that is used by frequent pattern mining algorithms is the *secure set union protocol*. This protocol allows the computation of unions of sets in a distributed way, without revealing the actual sources of the constituent elements. This is particularly useful in frequent pattern mining algorithms, because the locally large itemsets at the different sites need to be aggregated. The key in these methods is to disguise the frequent patterns at each site with enough number of fake itemsets, in order to disguise the true locally large itemsets at each site. Furthermore, it can be shown that this protocol can be generalized to compute different kinds of functions for various data mining problems on both horizontally and vertically partitioned data. The bibliographic notes contain pointers to surveys on these techniques.

20.6 Summary

Privacy-preserving data mining can be executed at different stages of the information processing pipeline, such as data collection, data publication, output publication, or distributed data sharing. The only known method for privacy protection at data collection, is the randomization method. In this method, additive noise is incorporated in the data at data collection time. The aggregate reconstructions of the data are then used for mining.

Privacy-preserving data publishing is typically performed using a group-based approach. In this approach, the sensitive attributes are treated in a different way from the attributes that are combined to construct quasi-identifiers. Only the latter types of attributes are perturbed, in order to prevent identification of the subjects of the data records. Numerous models, such as k-anonymity, ℓ-diversity, and t-closeness are used for anonymization. The eventual goal of all these methods is to prevent the release of sensitive information about individuals. When the dimensionality of the data increases, privacy preservation becomes very difficult, without a complete loss of utility.

In some cases, the output of data mining applications, such as association rule mining and query processing, may lead to release of sensitive information. Therefore, in many cases, the output of these applications may need to be restricted in to prevent the release

of sensitive information. Two such well known techniques are association rule hiding, and query auditing.

In distributed privacy, the goal is to allow adversaries or semi-adversaries to collaborate in the sharing of data, for global insights. The data may be vertically partitioned across columns, or horizontally partitioned across rows. Cryptographic protocols are typically used in order to achieve this goal. The most well-known among these is the oblivious transfer protocol. Typically, these protocols are used to implement primitive data mining operations, such as the dot product. These primitive operations are then leveraged in data mining algorithms.

20.7 Bibliographic Notes

The problem of privacy-preserving data mining has been studied extensively in the statistical disclosure control and security community [1, 512]. Numerous methods, such as swapping [181], micro-aggregation [186], and suppression [179], have been proposed in the conventional statistical disclosure control literature.

The problem of privacy-preserving data mining was formally introduced in [60] to the broader data mining community. The work in [28] established models for quantification of privacy-preserving data mining algorithms. Surveys on privacy-preserving data mining may be found in [29]. The randomization method was generalized to other problems, such as association rule mining [200]. Multiplicative perturbations have also been shown to be very effective in the context of privacy-preserving data mining [140]. Nevertheless, numerous attack methods have been designed for inferring the values of the perturbed data records [11, 367].

The k-anonymity model was proposed by Samarati [442]. The binary search algorithm is also discussed in this work. This paper also set up the basic framework for group-based anonymization, which was subsequently used by all the different privacy methods. The NP-hardness of the k-anonymity problem was formally proved in [385]. A survey of k-anonymous data mining may be found in [153]. The connections between the k-anonymity problem and the frequent pattern mining problem were shown in [83]. A set enumeration method was proposed in [83] that is similar to the set enumeration methods popularly used in frequent pattern mining. The *Incognito* and *Mondrian* algorithms, discussed in this chapter, were proposed in [335] and [336]. The condensation approach to privacy-preserving data mining was proposed in [8]. Some recent methods perform a probabilistic version of k-anonymity on the data, so that the output of the anonymization is a probability distribution [9]. Thus, such an approach allows the use of probabilistic database methods on the transformed data. Many metrics have also been proposed for utility-based evaluation of private tables, rather than simply using the minimal generalization height [29, 315].

The ℓ-diversity and t-closeness models were proposed in [348] and [372], respectively with a focus on sensitive attribute disclosure. A different approach for addressing sensitive attributes is proposed in [91]. A detailed survey of many of the privacy-preserving data publishing techniques may be found in [218]. A closely related model to group-based anonymization is *differential privacy*, where the differential impact of a data record on the privacy of other data records in the database is used to perform the privacy operations [190, 191]. While differential privacy provides theoretical more robust results than many group-based models, its practical utility is yet to be realized. The curse of dimensionality in the context of anonymization problems was first observed in [10]. Subsequently, it was shown that the curse extends to other privacy models such as perturbation and ℓ-diversity [11, 12, 372].

A practical example [402] of how high-dimensional data could be used to make privacy attacks is based on the Netflix data set [559]. Interestingly, this attack uses the sensitive ratings attributes and background knowledge to make identification attacks. Recently, a few methods [514, 533] have been proposed to address the curse of dimensionality in a limited way.

The problem of output privacy is closely related to the problem of inference control and auditing in statistical databases [150]. The most common problems addressed in this domain are those of association rule hiding [497], and query auditing [399]. Distributed methods transform data mining problems into secure multi-party computation primitives [188]. Typically, these methods are dependent on the use of the oblivious transfer protocol [199, 401]. Most of these methods perform distributed privacy-preservation on either horizontally partitioned data [297] or vertically partitioned data [495]. An overview of the various privacy tools for distributed information sharing may be found in [154].

20.8 Exercises

1. Suppose that you have a 1-dimensional dataset uniformly distributed in $(0, 1)$. Uniform noise from the range $(0, 1)$ is added to the data. Derive the final shape of the perturbed distribution.

2. Suppose that your perturbed data was uniformly distributed in $(0, 1)$, and your perturbing distribution was also uniformly distributed in $(0, 1)$. Derive the original data distribution. Will this distribution be accurately reconstructed, in practice, for a finite data set?

3. Implement the Bayes distribution reconstruction algorithm for the randomization method.

4. Implement the (a) *Incognito*, and (b) *Mondrian* algorithms for k-anonymity.

5. Implement the condensation approach to k-anonymity.

6. In dynamic condensation, one of the steps is to split a group into two equal groups along the longest eigenvector. Let λ be the largest eigenvalue of the original group, $\overline{\mu}$ be the original d-dimensional mean, and $\overline{V}$ be the longest eigenvector, which is normalized to unit norm. Compute algebraic expressions for the means of the two split groups, under the uniform distribution assumption.

7. Show that the sensitive attribute in both the entropy- and recursive- ℓ-diversity models must have at least ℓ distinct values.

8 Show that the global entropy of the sensitive attribute distribution is at least equal to the minimum entropy of an equivalence class in it. [**Hint:** Use convexity of entropy]

9. Many k-anonymization algorithms such as *Incognito* depend upon the monotonicity property. Show that the monotonicity property is satisfied by (a) entropy ℓ-diversity, and (b) recursive ℓ-diversity.

10. Implement the (a) *Incognito*, and (b) *Mondrian* algorithms for entropy- and recursive ℓ-diversity, by making changes to your code in Exercise 4.

11. Show that the monotonicity property is satisfied by (a) t-closeness with variational distances, and (b) t-closeness with KL-measure.

12. Consider any group-based anonymity quantification measure $f(\overline{P})$, in which the anonymity condition is of the form $f(\overline{P}) \geq thresh$. (An example of such a measure is entropy in ℓ-diversity.) Here, $\overline{P} = (p_1 \ldots p_r)$ is the sensitive attribute distribution vector. Show that if $f(\overline{P})$ is concave, then the anonymity definition will satisfy the monotonicity property with respect to generalization. Also show that convexity ensures monotonicity in the case of anonymity conditions of the form $f(\overline{P}) \leq thresh$.

13. Implement the (a) *Incognito*, and (b) *Mondrian* algorithms for variational distance-based, and KL distance-based t-closeness, by making changes to your code for Exercise 4.

14. Suppose that you had an anonymized binary transaction database containing the items bought by different customers on a particular day. Suppose that you knew that the transactions of your family friend contained a particular subset B of items, although you did not know the other items bought by her. If every item is bought independently with probability 0.5, show that the probability that at least one of n other customers buys exactly the same pattern of items, is given by *at most* $n/2^B$. Evaluate this expression for $n = 10^4$ and $B = 20$. What does this imply in terms of the privacy of her other buying patterns?

15. Repeat Exercise 14 for movie ratings taking on one of R possible values instead of 2. Assume that each rating possibility has identical probability of $1/R$, and the ratings of different movies are independent and identically distributed. What are the corresponding probabilities of re-identification with B known ratings, and n different individuals?

16. Write a computer program to re-identify the subject of a database with B known sensitive attributes.