Relational data anonymization using Reinforcement Learning

Alexandros Giavaras[1]

Department of of Genetics and Genome Biology, University of Leicester, Leicester, UK {a.giavaras}@gmail.com

Abstract. We discuss the anonymization of a tabular data set using reinforcement learning. In particular, we employ three commonly use reinforcement learning algorithms; q-learning, n-step semi-gradient SARSA and actor-critic. We compare the dataset anonymization achieved using the K-anonymity algorithm as this is implemented in the ARX software.

Keywords: Privacy preserving data publishing · Data anonymity · Reinforcement learning · Relational data.

1 Introduction

Modern organizations and modern applications are highly data oriented and this pattern is not anticipated to change soon as more and more enterprises decide to invest to infrastructure e.g. cloud platforms, so that they store data and consequently harness it. User's privacy however, is of paramount importance. Data anonymization, therefore plays a crucial role in maintaining user information private.

Data owners such as hospitals, banks, social network service providers and insurance companies, anonymize their user's data before publishing it to protect the privacy of users whereas anonymous data remains useful for legitimate information consumers

Before a data owner publishes the user's data, he should ensure that the user's private information is protected. A practical approach to achieve this goal is via data anonymization methodologies [REF]. Data anonymization techniques transform the original data set by applying operations on it. Thus they distort, to some extent, the statistics present in the original data [REF] in order to user's privacy.

There are two main settings the data owner can make available the data; an interactive setting and a non-interactive one [REF]. In the former scenario, the data owner does not release the entire data set. Instead an interface is provided through which interested parties may pose queries about the data. The API then returns answers to these queries possibly with added noise [REF].

Three can identify the following types of attacks [REF]

– Journal

- Random
- What else

In this work, we are interested in tabular or relational data sets. Specifically, we assume that the user's data D is considered as a private table which consists of multiple tuples. In this setting, each tuple or row of the data set contains four types of attributes [6]

- Direct identifiers (DI) e.g. a user's name
- Non-sensitive attributes (NSA) e.g. the height or weight of a user
- Quasi identifiers (QI) e.g. age, zip-code
- Sensitive attribute (SA) e.g. any convictions, health status

A direct identifier can directly and uniquely identify a user. A quasi-identifier can be linked with other information in order to reveal a user's identity. A sensitive attribute refers to an attribute that a user does not wish to disclose. A non-sensitive attributes is every attribute other than the aforementioned ones.

Based on the type's of user's attributes mentioned above, there exists three classes of privacy threats that can occur during published data [6]

- Identity disclosure
- Attribute disclosure
- Membership disclosure

Identity disclosure occurs when an adversary can correctly associate an individual in a privacy preserved published data set [6]. Attribute disclosure occurs when an individual is linked with information related to their sensitive attributes [6]. Finally, membership disclosure occurs when an adversary can deduce that an individual's record is present or absent in the published data set. In order to protect the privacy of users, data owners typically apply various transformations on the original data [6]. Examples of such transformations include generalization, suppression, permutation, pertubation and anatomization. Moreover, in certain cases, more than one operations may be applied.

The general concept of relational anonymization is to produce anonymous table say D_{anon} from the original table D. In particular, a privacy model or algorithm is used in order to modify the original values in such a way that on the one hand the user's privacy is or can be protected and on the other hand retaining significant utility in the resulting table D_{anon}. In this regard, several privacy models have been proposed over the years.

Some of the most extensively used approaches are the $k-$anonymity model [9], the $l-$diversity model [5], the $t-$closeness model [4] and the differential privacy model [1]. $k-$anonymity is a well known privacy model which was devised for the protection of identity disclosure [6]. The model protects the user's privacy by placing at least k users in an equivalence class with the same quasi-identifier values [6], **??**. This results in a probability of $1/k$ or user re-identification. A data set satisfies $k-$anonymity if for every tuple t in D_{anon} there exist at least $k-1$ other tuples with the same QIs in an equivalence class [6]. Despite its popularity,

the vanilla $k-$anonymity model cannot protect sensitive information disclosure. The $l-$diversity model was proposed to solve these limitations. According to the $l-$diversity model, the equivalence classes satisfy the $l-$diversity property if there are at least l well represented values for the sensitive attributes. Thus D_{anon} is said to have the $l-$diversity property, if every equivalence class is $l-$diverse [6]. Overall, the $l-$diversity model exhibits superior privacy protection when it is compared with $k-$anonymity. However, the algorithm neglects the distribution of the values of the sensitive attributes. Hence, for equivalence classes where one particular sensitive attribute value dominates over others, the privacy may break down for them. Moreover, $l-$diversity degrades anonymous data utility significantly by not considering the distributions of quasi-identifiers and their similarities during anonymization [6]. The $t-$closeness model was proposed to remedy both $k-$anonymity and $l-$diversity anonymization models. In particular, a data set D satisfies the $t-$closeness if its tuples are split into equivalence class such that the distribution of sensitive attributes in all D and the equivalence classes of a $t-$close data set D_{anon} are within $t-$distance units from each other [6]. The $t-$closeness model significantly improves the user's privacy but it reduces significantly reduces the utility of the released data. The differential privacy model is mostly used in interactive settings. It protects the privacy of the user by adding noise to the original data and makes no assumptions about the intrusion scenarios.

In this work, we discuss the applicability of three reinforcement learning algorithms for tabular data anonymization. Specifically, Q-learning, semi-gradient SARSA and advantage actor-critic with shared weights. The rest of this work is structured as follows In section [RL SECTION] we give a brief overview of the reinforcement learning paradigm. Section 3 discusses the approach we employ herein. In section 4 we apply the aforementioned three algorithms and compare the results with a K-anonymity based anonymization. Section 5 summarizes our results discusses improvements and provides some insights for further work.

2 Reinforcement learning overview

In this section, we give a brief overview of the reinforcement learning paradigm. We keep the presentation short and only focus on the essential ingredients of the paradigm. For further information the user is referred to [8].

In the following subsections we review three methodologies to train an agent namely Qlearning, semi-gradient SARSA, and advantage actor-critic.

2.1 Q-learning

Q-learning is one of the early breakthroughs in the field of reinforcement learning [8]. It was first introduced in [10]. Q-learning is an off-policy algorithm where the learned state-action value function $Q(s, a)$ directly approximates the optimal state-action value function q^*. This is done independently of the policy π being followed [8]. Q-learning is an iterative algorithm where we iterate over a number

of episodes. At each episode the algorithm steps over the environment for a user-specified number steps it executes an action which results in a new state. Specifically, the algorithm uses the following update rule

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{1}$$

where γ is the discount factor, α is the learning rate

2.2 Semi-gradient SARSA

In this section we briefly discuss the second algorithm we implemented, namely semi-gradient SARSA algorithm. Furthermore, we consider every column in the data set as a feature that takes values in the range $[0, 1]$. This is possible since we are using normalized distances. Thus, the state vector now consists of multiple features. We use tile coding in order to discretize the state space see [8].

One of the major disadvantages of Qlearning we saw in the previous examples, is that we need to use a tabular representation of the state-action space. This poses limitations on how large the state space can be on current machines; for a data set with, say, 5 columns when each is discretized using 10 bins, this creates a state space of the the order $O(10^5)$. Although we won't address this here, we want to introduce the idea of weighting the columns. This idea comes from the fact that possibly not all columns carry the same information regarding anonimity and data set utility. Implicitly we decode this belief by categorizing the columns as

Thus, in this example, instead to representing the state-action function Q_π using a table we will assume a functional form for it. Specifically, we assume that the state-action function can be approximated by $\hat{Q} \approx q_\pi$ given by

$$\hat{Q}(s, a) = \mathbf{w}^T \mathbf{x}(s, a) = \sum_i^d w_i, x_i(s, a) \tag{2}$$

where $\mathbf{w}$ is the weights vector and ‘$\mathbf{x}(s, a)$ is called the feature vector representing state s when taking action a [8]. We will use tile coding to construct the feature vector $\mathbf{x}(s, a)$ [8]. Our goal now is to find the components of the weight vector. We can use stochastic gradient descent (or SGD) for this. In this case, the update rule is:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[U_t - \gamma \hat{Q}(s_t, a_t, \mathbf{w}_t) \right] \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t, \mathbf{w}_t) \tag{3}$$

Given that $\hat{Q}$ is a linear function, it follows that

$$\nabla_{\mathbf{w}} \hat{Q}(s, a) = \mathbf{x}(s, a) \tag{4}$$

Moreover, U_t, in the case one one step SARSA is given by

$$U_t = R_t + \gamma \hat{Q}(s_{t+1}, a_{t+1}, \mathbf{w}_t) \tag{5}$$

2.3 Advantage actor-critic

Both the Q-learning algorithm and the SARSA algorithm in are value-based methods; that is they estimate directly value functions. Specifically the state-action function Q. By knowing Q we can construct a policy to follow for example to choose the action that at the given state maximizes the state-action function i.e. a greedy policy. These methods are called off-policy methods.

However, the true objective of reinforcement learning is to directly learn a policy π. One class of algorithms towards this directions are policy gradient algorithms like REINFORCE and advantage actor-critic algorithms. A review of A2C methods can be found in [3].

Typically with policy gradient methods, we approximate directly the policy by a parameterized model. Thereafter, we train the model i.e. learn its parameters by taking samples from the environment. The main advantage of learning a parameterized policy is that it can be any learnable function e.g. a linear model or a deep neural network.

In advantage actor-critic or A2C, we estimate two models; a parameterized policy that is called the actor and a parameterized value function; the critic. The role of the policy or actor network is to indicate which action to take on a given state. In our implementation below, the policy network returns a probability distribution over the action space. The role of the critic model is to evaluate how good is the action that is selected. In our implementation we use a shared-weights model and use a single agent that interacts with multiple instances of the environment. In other words, we create a number of workers where each worker loads its own instance of the data set to anonymize.

The objective function for the policy network is given by equation 6

$$J(\pi_\theta) = E_{\tau\sim\rho_\theta}\left[\sum_{t=0}^{T}\gamma^t R(s_t, a_t)\right] \tag{6}$$

where $R(s_t, a_t)$ is some unknown to the agent reward function, τ is the trajectory the agent observes with probability distribution ρ_θ. The gradient of $J(\pi_\theta)$ is estimated by using the equation below

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{T}\nabla_\theta log(\pi_\theta(a_t, s_t)\right) A(s_t, a_t) \tag{7}$$

where $A(s_t, a_t$ is the so-called advantage function given by

$$A(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t) \tag{8}$$

The advantage function measures how much the agent is better off by taking action a_t when in state s_t as opposed to following the existing policy. In order to estimate the advantage function we use the Generalized Advantage Estimation or GAE [7]. Specifically, we use the following expression for the advantage function

$$A(s_t, a_t)^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+1} \tag{9}$$

Since we are using a shared model, the loss function, we minimize is a weighted sum of the two loss functions of the participating models i.e.

$$L(\theta) = w_1 L_\pi(\theta) + w_2 L_{V_\pi}(\theta) \tag{10}$$

where

$$L_\pi(\theta) = J(\pi(\theta)) \quad L_{V_\pi}(\theta) = MSE(y_i, V_\pi(s_i)) \tag{11}$$

where MSE is the mean square error function and y_i are the state-value targets i.e.

$$y_i = r_i + \gamma V_\pi(s_i^{'}), \quad i = 1, \cdots, N \tag{12}$$

where N is the size of the batch. The algorithm is described in listing 1

Algorithm 1 A2C algorithm

```
Specify initial policy π_w
Specify number of episodes N_ep to play
Specify memory size N_M for memoryBuffer
Initialize parameters for actor and critic networks
for Episode ∈ range( N_ep ) do
    Accumulate experience in the memory buffer
    if len( memoryBuffer ) < N_M then
        continue
    end if
    Obtain the state-value targets y_i = r_i + γV_φ, i = 1, ..., N_M
    Use gradient descent to update φ using the specified loss function L(V_φ, y_i)
    Obtain the advantage value estimates A(s_i, α_i
    Use the advantage value estimates to calculate ∇_θ J
    Update the parameters for the actor and critic models
end for
```

In the following section we give the details of our implementation.

3 Methodology

In this section we present our approach. Thus, we assume that a data set D is available and we wish to adapt it such that privacy concerns are satisfied to a given degree. As already mentioned, anonymizing a given data set entails some degree of distortion of the latter. The aim is to strike a balance between the utility that a data set offers to a researcher and the...expose the private details [2] i.e. PUT.

Furthermore, we consider the following two scenarios; the data set will be fully released to users or to the public or the users query the data set behind a

restrictive API. The schema of the data set contains strings, continuous, discrete and categorical variables.

We use the following discrete action space

$$A = [\text{IDENTITY}, \text{GENERALIZATION}, \text{SUPPRESSION}] \tag{13}$$

Actions are applied per column and are organized as serial hierarchies. An $IDENTITY$ action leaves the column unaltered and should be used on insensitive columns. A GENERALIZATION action progressively generalizes a column. The user should explicitly specify for each possible value that a column can take the generalization hierarchy. For example, consider the column ETHNICITY

The Q-learning algorithm from sub-section 2.1 uses a tabular representation to approximate the state-action value function. This means we cannot use it when we have continuous states or actions, which would lead to an array of infinite length. In order to overcome this, we assume that the total distortion of the data set lies in the range $[0, 1]$ of the real numbers; where the edge points mean no distortion and full distortion of the data set/column respectively. We further discretize this range into bins and for each entailed value of the distortion we use the corresponding bin as a state index. This is known as state aggregation. Another approach we could use is to discretize the distortion of each column into bins and create tuples of indices representing a state.

As already mentioned, tabular representations ensue certain disadvantages when working with continuous spaces.

We use Actor-Critic reinforcement learning algorithm (why?). We develop the deep neural networks using PyTorch. Data represented as a string e.g.

3.1 Environment state discretization

We assume that we have a $Dist_{min}$ and a $Dist_{max}$. The former represents the minimum distortion that should be applied in order to have an anonymized dataset. The latter represents the maximum distortion that should be exercised in order to have some utility from the dataset. The agent's goal, thus, is to drive the dataset distortion within this range.

Fig. 1: Possible factorisations of $P(hair, nationality, gender)$

American	**Swedish**
0.5	0.5

Table 1: $P(nationality)$

	Blond	**Brown**	**Dark**
American	0.2	0.6	0.2
Swedish	0.8	0.2	0

Table 2: $P(hair|nationality)$

	Male	**Female**
American	0.5	0.5
Swedish	0.45	0.55

Table 3: $P(gender|nationality)$

	Blond	Brown	[Dark, Red]
American	0.2	0.5	3
[British, French]	0.4	0.3	3
Swedish	0.8	0.2	0

Table 4: $P(hair|nationality)$ with $k = 2$ and $j = 1$

Algorithm 2 Steiner tree extraction

```
function Walk(node, required, path, relevant)
    if required is empty then
        return {}
    else if node in nodes then
        required ← required \ {node}
        relevant ← relevant ∪ path
    end if
    path ← path ∪ {node}
    for child ∈ node.children() do
        relevant ← relevant ∪ Walk(child, required, path, relevant)
    end for
    return relevant
end function

function ExtractSteinerTree(tree, nodes)
    nodes ← nodes ∪ tree.root()
    relevant ← Walk(tree, nodes, {}, {})
    return tree.subset(relevant))
end function
```

4 Results

In this section we apply the developed algorithms to a mock data set. The data set consists of 10000 tuples with 11 attributes. These attributes are:

- National insurance number
- Name
- Surname

There are various metrics that can be used for the evaluation of privacy and utility of relational data anonymization techniques. some privacy evaluation metrics include (i) anonymous and original data set linking and calculating the probability of successful matches based on the quasi-identifiers in both data sets (ii) privacy protection evaluation in presence of background knowledge (iii) privacy protection evaluation with the help of privacy-sensitive rules. On the other hand, some utility evaluation include accuracy or error rate, $F-$measures, precision, recall and information loss. In particular, the weighted certainty penalty, generalized information loss and KL-divergence.

-
-

and we compare the column and overall data set distortion with the distortion obtained when applying K-anonymity on the original data set. For the latter algorithm we use the ARX software implementation[1].

We prepossess the input data set by normalizing the numeric columns. We will use the cosine normalized distance to measure the distortion of columns with string data. Similarly, we use the following L_2-based norm for calculating the distortion of numeric columns

$$dist(\mathbf{v}_1, \mathbf{v}_2) = \sqrt{\frac{||\mathbf{v}_1 - \mathbf{v}_2||^2_{L_2}}{n}} \tag{14}$$

where n is the size of the vector $\mathbf{v}$. For all the tests below we used $\gamma = 0.99$, $\alpha = 0.1$ and train the algorithm for 1000 episodes. We report running averages over 100 episodes of the reward and the overall data set distortion. The latter is measured as the sum of the column distortions.

The K-anonymity solver was configured as follows.... The K-anonimity algorithm resulted in a distorted data where 2516 rows were dropped completely. set with 49 equivalence classes, whilst 48 records were compressed. The table below shows the achieved column distortions and the total data set distortion

	ethnicity	**salary**	**Total**
Q-learning	0.2753043496334219	0.06317501926703974	0.3384793689004616
K-anonymity	0.14584312774851615	0.2741636517496087	0.42000677949812487

Table 5: Overall and per column distortions for mock data set.

5 Conclusions and discussion

In this work, we presented some preliminary results for relational data set anonymization using reinforcement learning techniques. Specifically, we investigated the applicability of three algorithms namely Q-learning, semi-gradient SARSA and advantage actor-critic.

The simple Q-learning algorithm exhibited elements of learning as shown in figure **??**. On

We believe that the results presented herein are encouraging. However, further is needed. Specifically, we need to investigate further into the distance functions employed for measuring the column distortions. Furthermore, the performance of the algorithms in general depends on how an action is applied on a column. In addition, we would like to incorporate into the action space $l-diversity$ and $t-$closeness so that we can apply these actions on the sensitive attributes which currently we leave intact. Additionally, we would like to explore how to incorporate $k-$anonymity in an effective way in our framework. Finally, we would like to investigate how the reinforcement learning paradigm can be used behind

[1] The code for the numerical experiments presented herein can be found at `https://rl-anonymity-with-python.readthedocs.io/en/latest/index.html`. Furthermore, the code for the backing the work herein can be found at `https://github.com/pockerman/rl_anonymity_with_python`

a query API to build belief models on the users knowledge on the data set sitting behind the API.

References

1. Dworkd, C.: Differential privacy: A survey of results. In: International conference on Theory and applications of models of computation. pp. 1–19. Springer (2009)
2. Erdemir, E., Dragotti, P.L., Gunduz, D.: Privacy-aware location sharing with deep reinforcement learning. In: Proceedings of the 2019 IEEE WIFS conference. pp. 1–6. IEEE (2019)
3. Grondman, I., Busoniu, L., Lopes, G, B.R.: A survey of actor-critic reinforcement learning: Standard and natural policy gradients (2012)
4. Li, N., L, T., Gehrke, J., Venkatasubramaniam, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: IEEE 23rd International conference on data engineering. pp. 106–115. IEEE (2007)
5. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on knowledge discovery from data **1**, 3–es (2007)
6. Majeed, A., Lee, S.: Anonymization techniques for privacy preserving data publishing. a comprehensive servey. Proceedings of the 2019 IEEE WIFS conference **4**, 1–6 (2016)
7. Schulman, K., Moritz, P., Levine, S., Jordan M, A.P.: High-dimensional continuous control using generalized advantage estimation.
8. Sutton, R.S., Barto, A.G.: Reinforcement Learning. An Introduction. The MIT Press (2020)
9. Sweeney, L.: k-anonymity: A model for protecting privacy. International journal of uncertainty, fuzziness and knowledge-based systems **10**, 557–570 (2002)
10. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King's College Cambridge (1989)