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, semi-gradient SARSA and advantage actor-critic (A2C). We compare the data set 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. 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 can store and harness the data they collect. User's privacy, however, is of paramount importance. This is illustrated by the GPDR enforced in European union on May 2018 [1] and . According to these regulations, user's privacy is of paramount importance. Therefore, data owners, such as hospitals, banks, social network service providers and insurance companies, should protect the privacy of the users they maintain in their databases.

A practical approach to achieve this goal is via data anonymization methodologies [6]. These techniques transform the original data set by applying operations on it. Typically, these operations transform the data set to some extent. Consequently, the statistics present in the original data are altered in order to establish user's privacy. Moreover, there are two main settings the data owner can make available the data; an interactive setting and a non-interactive one [6]. 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 [6]. In the latter scenario, the data owner releases the distorted data set to the public possibly under specific conditions [6].

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

[1] see for instance https://gdpr.eu/

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 [11], 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]. It is a model to reduce the likelihood of any single person being identified when the data set is linked with external data sources. Overall, the approach is based on generalizing attributes and possibly deleting records. Thus, 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. Indeed, this will be the case if the k records within a similarity group share the same value

of the sensitive attribute. 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 2 we give a brief overview of the reinforcement learning paradigm. Section 3 discusses some details of the methodology 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 [10].

The reinforcement learning paradigm, generally, assumes a Markov decision process, or MDP, that consists of set of states $\mathbb{S}$ a set of actions $\mathbb{A}$, a transition function P that denotes the probability of moving to state $s^{'}$ when the agent is at state $s \in \mathbb{S}$ and takes action $a \in \mathbb{A}$ and a reward function $R(s, a)$ that is unknown to the agent. In this paradigm, the objective is typically to learn a policy π which is defined as a probability distribution over actions conditioned on states i.e. $\pi(a|s)$. In particular, the typical problem formulation in reinforcement learning is to maximize the expected total reward of a policy. For a finite horizon T this is given by

$$J(\pi) = E\left[\sum_{t=0}^{T} R(s_t, a_t)\right] \tag{1}$$

There are many ways that can be used in order to estimate π depending on the shape of the state space $\mathbb{S}$ and/or the shape of the action space $\mathbb{A}$. Each approach has its advantages and disadvantages and the reader is referred to [10] and the references therein as well as to the abundant literature on the subject see for example [7] and [3].

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 [10]. It was first introduced in [12]. 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 [10]. Thus, the Q-learning algorithm does not approximate the policy π directly. Instead it reconstructs it from the estimate Q.

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. The success of the algorithm, as in most reinforcement learning algorithms, is largely dependent on the exploration and exploitation dilemma [10]. In order to obtain an optimal solution, the agent cannot pursue exclusively the current estimate of the policy i.e. exploit without failing the task at hand. Therefore, some degree of exploration should be specified. This is typically done using an $\epsilon-$greedy policy for action selection during training where with probability ϵ the agent selects randomly an action from $\mathbb{A}$ to execute. The algorithm is summarized in listing 1 (see also [10]).

Algorithm 1 Q-learning algorithm

```
Specify initial policy π_INIT
Specify number of episodes N_EPS
Specify number of iterations per episode N_ITR
Initialize Q(s, a) ∀s ∈ 𝕊, a ∈ 𝔸
Specify learning rate α ∈ (0, 1]
for Episode ∈ range( N_EPS ) do
    Reinitialize the environment
    for Itr ∈ range(N_ITR) do
        Choose a from current s using policy π_INIT derived from Q (e.g. ε−greedy)
        Take action a and observe the reward r and the next state s'
        Update Q using equation 2
    end for
end for
```

The algorithm uses the following update rule, equation 2,

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{2}$$

where γ is the discount factor, α is the learning rate. Moreover, it assumes a generic initial policy to be specified as input. Most of the time, and what we use herein, this will be an $\epsilon-$greedy policy that utilizes Q.

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 [10].

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 anonymity 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

$$\hat{Q}(s, a) = \mathbf{w}^T \mathbf{x}(s, a) = \sum_i^d w_i, x_i(s, a) \tag{3}$$

where $\mathbf{w}$ is the weights vector and '$\mathbf{x}(s, a)$ is called the feature vector representing state s when taking action a [10]. We will use tile coding to construct the feature vector $\mathbf{x}(s, a)$ [10]. 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:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[U_t - \gamma \hat{Q}(s_t, a_t, \mathbf{w}_t) \right] \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t, \mathbf{w}_t) \tag{4}$$

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

$$\nabla_{\mathbf{w}} \hat{Q}(s, a) = \mathbf{x}(s, a) \tag{5}$$

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

$$U_t = R_t + \gamma \hat{Q}(s_{t+1}, a_{t+1}, \mathbf{w}_t) \tag{6}$$

The algorithm is summarized in listing 2, see also [10],

Algorithm 2 Semi-gradient SARSA algorithm

```
Specify initial differential policy π_INIT
Specify number of episodes N_EPS
Specify number of iterations per episode N_ITR
Initialize w ∈ ℝ^d arbitrarily
Specify learning rate α ∈ (0, 1]
for Episode ∈ range( N_EPS ) do
    Reinitialize the environment
    for Itr ∈ range(N_ITR) do
        Choose a on current s using policy π_INIT derived from Q (e.g. ε−greedy)
        Take action a and observe the reward r and the next state s'
        if s' is terminal then
            Update w using equation 7
            break and go to next episode
        end if
        Choose a' as a function of π_INIT
        Update w using equation 8
        s ← s'
        a ← a'
    end for
end for
```

The update equations mentioned in listing 2 are given below respectively.

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[r_t - \gamma \hat{Q}(s_t, a_t, \mathbf{w}_t) \right] \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t, \mathbf{w}_t) \tag{7}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[r_t + \gamma \hat{Q}(s^{'}, a^{'}, \mathbf{w}_t) - \hat{Q}(s, a), \mathbf{w}_t) \right] \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t, \mathbf{w}_t) \tag{8}$$

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 9

$$J(\pi_\theta) = E_{\tau \sim \rho_\theta} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right] \tag{9}$$

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

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T} \nabla_\theta log(\pi_\theta(a_t, s_t) \right) A(s_t, a_t) \tag{10}$$

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

$$A(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t) \tag{11}$$

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 [9]. Specifically, we use the following expression for the advantage function

$$A(s_t, a_t)^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+1} \tag{12}$$

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.

$$L(\theta) = w_1 L_\pi(\theta) + w_2 L_{V_\pi}(\theta) \tag{13}$$

where

$$L_\pi(\theta) = J(\pi(\theta)) \quad L_{V_\pi}(\theta) = MSE(y_i, V_\pi(s_i)) \tag{14}$$

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

$$y_i = r_i + \gamma V_\pi(s_i^{'}), \quad i = 1, \cdots, N \tag{15}$$

where N is the size of the batch. The algorithm is described in listing 3

Algorithm 3 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 $\in$ range(N_{ep}) **do**
    Accumulate experience in the memory buffer
    if len($memoryBuffer$) $< N_M$ **then**
        continue
    end if
    Obtain the state-value targets $y_i = r_i + \gamma V_\phi, i = 1, \ldots, N_M$
    Use gradient descent to update ϕ using the specified loss function $L(V_\phi, y_i)$
    Obtain the advantage value estimates $A(s_i, \alpha_i$
    Use the advantage value estimates to calculate $\nabla_\theta J$
    Update the parameters for the actor and critic models
end for

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

3 Methodology

The previous section summarized the algorithms that we employ in this work. Reinforcement learning agents interact with an environment and utilize the reward returned from the environment in order to make progress. In our case the environment consists of the data set D to be anonymized. The data set may contain numeric, string and integer attributes or a mixture of all these. We need, therefore, to somehow model the data set so that is suitable for applying the algorithms described previously. We do as follows.

Our aim is to adapt D such that privacy concerns are satisfied to a given minimum degree. We denote this degree as D_{min}. Moreover, anonymizing a data set entails some degree of distortion. We denote this with D_{max}. The aim is to strike a balance between the utility that a data set offers to a researcher and the satisfaction of the privacy concerns [2]. Thus, our general objective is to use reinforcement learning to device a policy π such that the overall data set distortion lies in the range $[D_{min}, D_{max}]$. In this work, we measure the overall data set distortion as the sum of the induced column distortions. However, other metrics can be used.

The data set D is allowed to consist of columns with different types of attributes although each column can only have a certain type of data. In order to achieve uniformity, we use normalized distances for the string attributes. We further normalize the numeric columns to lie in the range $[0, 1]$.

3.1 Action space

The action space, $\mathbb{A}$, is an integral part of any reinforcement learning algorithm as the agent operate according to a specific actions in it. In our case the action space can be specified as the set of transformations described in section 1 namely

- Generalization
- Suppression
- Permutation
- Pertubation
- Anatomization

Moreover, we enhance this action space by assuming an identity action i.e. leave the column at the current state as well as a restore action i.e. restore the column to the original state. Note also that we can encode in the action space various models such $l-$diversity or $t-$closeness. In the current implementation an action is applied to a single column.

3.2 Rewards

The environment has to send to the agent the reward signal r. Specifying properly the rewards is a challenging problem. One option could be to assign a reward every time an action is executed. This leads to dense rewards and allows for more supervision ensuing faster training. However, this may potentially prevent the agent from coming up with novel strategies. The other extreme is to assign the reward upon the end of the episode. This results in more sparse rewards that makes training more time consuming. However, it allows the agent to come up with possibly unexpected policies.

In this work, we use the first approach for simplicity. Moreover, we use the following simple reward scheme. If the total distortion entailed by an action of the agent is less than D_{min} or greater than D_{max} the agent is assigned a reward of -2. Otherwise, i.e. if the total distortion is in $[D_{min}, D_{max}]$, the agent is given a reward of 5. Moreover, an episode is terminated when all columns in the data set have been visited once regardless of the action applied to them.

3.3 State space discretization

For the A2C algorithm use a simple linear fully connected layers to model both the critic and the actor. We feed the network with a vector describing the distortions of each column. The columns that are classified as direct identifiers, are accounted for in the vector but given that they are suppressed, i.e. giving a distortion of 100%, we scale this distortion by a user defined factor. This allows us to partly take into account this side of the data set and allow the models to make progress.

The Q-learning and the semi-gradient SARSA algorithms employ a tabular representation of Q. In this case we can use state aggregation [10]. In particular, we can either discretize the range $[0, 1]$ into a user specified number of bins in which case the state will simply be a bin index or we can use the same approach for every column in which case the state will be a tuple of indices. The latter approach may be prohibitive for data sets with a large number of columns and/or when the number of bins used is large. For a data set with 5 columns and using 10 bins for each column, the state space will be at the order of $O(10^5)$. This

will be particularly prominent for the semi-gradient SARSA algorithm for which we use tiling, see [10], in order to model the feature vector $\mathbf{x}$. In this case, the data set is overlaid by a set of layers. A layer consists of tiles that represent the discretization of every column. Thus, for a data set of N_{cols} columns and using N_{bins} bins for discretizing every column, a layer consists of $|\mathbb{A}|(N_{bins})^{N_{cols}}$ in total. Each layer is a translation of the layer before it. The translation width is simply the tile width.

Given the above, for the Q-learning and semi-gradient SARSA algorithms we just discretize the quasi-identifying columns.

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*
- *Gender*
- *Dob*
- *Ethnicity*
- *Education*
- *Salary*
- *Mutation status*
- *Preventative treatment*
- *Diagnosis*

From the attributes above, the *Name*, *Surname*, *Gender*, *Ethnicity* and *Preventative treatment* are strings. The *Diagnosis*, *Mutation status*, and *Education* are integers. The *Salary* attribute is treated as floating point number which we restrict in the normalized range $[0, 1]$ Following [6], we set the *National insurance number*, *Name* and *Surname* as direct identifiers and completely suppress them in the resulting anonymized data set. The rest of the variables some are treated as quasi-identifiers, *Ethnicity*, *Gender*, *Salary*, as sensitive attributes, *Dob*, *Education*, *Mutation status*, *Preventative treatment* or as insensitive attributes *Diagnosis*. Currently, in our approach, only the quasi-identifiers are transformed whilst the sensitive attributes are left intact.

The following figures summarize the data in the quasi-identifier columns.

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.

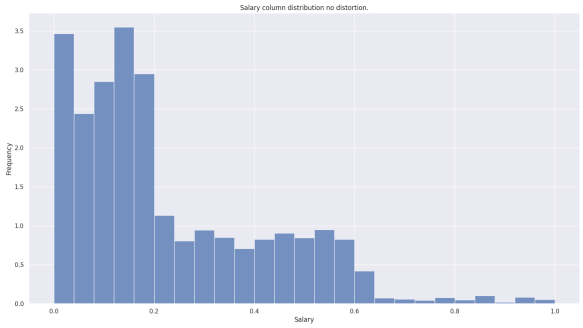

Fig. 1: Distribution of *Salary* column prior to anonymization.

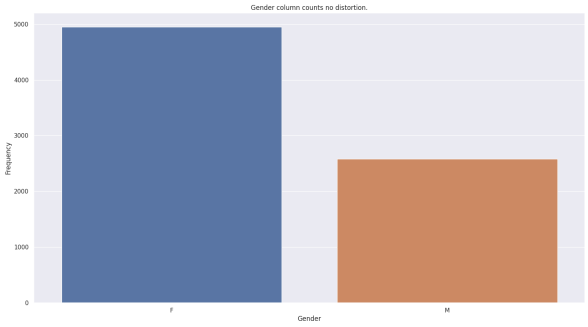

Fig. 2: Counts of *Gender* column prior to anonymization.

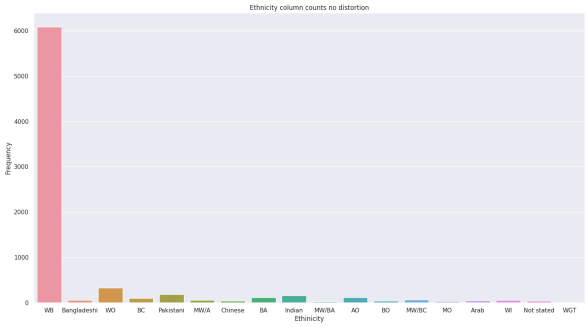

Fig. 3: Counts of *Ethnicity* column prior to anonymization.

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 [8] implementation[2].

[2] The code for the numerical experiments presented herein can be found at `https://rl-anonymity-with-python.readthedocs.io/en/latest/index.html`. Further-

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 attributes. Similarly, we use the following L_2-based norm for calculating the distortion of numeric columns

$$dist(\mathbf{v}_1, \mathbf{v}_2) = \sqrt{\frac{||\mathbf{v}_1 - \mathbf{v}_2||^2_{L_2}}{n}} \tag{16}$$

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.

We used two settings for the K-anonymity solver. In the first configuration we set the suppression limit to zero per cent. The second configuration used a two per-cent suppression limit. Although ARX allows for specifying privacy models the sensitive attributes, we did not use this options and actually classify all these variables as insensitive attributes. We set $k = 5$ for both configurations. Finally both configurations use an entropy metric as a quality model. The K-anonymity algorithm resulted in a distorted data where 2516 rows were dropped completely. set with 49 equivalence classes, whilst 48 records were compressed.

The following figures show the distribution of the quasi-identifier columns. As it can be seen the algorithm maintains the distribution of the *Gender* column but completely suppresses the *Salary* column. The algorithm produced an anonymized data set with 22 equivalence classes with an average size of 343 tuples.

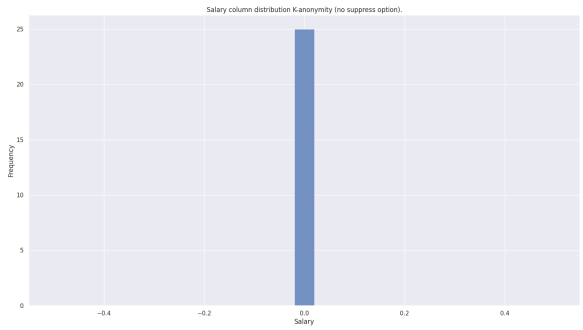

Fig. 4: Distribution of *Salary* column with k-anonymity and no suppression.

In contrast, setting the suppression limit to 2% results in an anonymized data set with 77 equivalent classes with an average size of 97 tuples. Moreover the algorithm suppresses 107 rows from the original data set. The following figures illustrate the distribution of the quasi-identifiers in the resulting data set.

more, the code for the backing the work herein can be found at `https://github.com/pockerman/rl_anonymity_with_python`

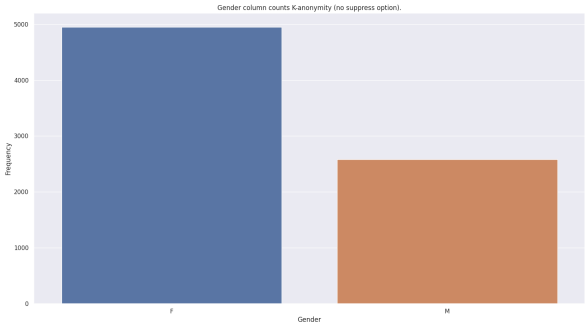

Fig. 5: Counts of *Gender* column with k-anonymity and no suppression.

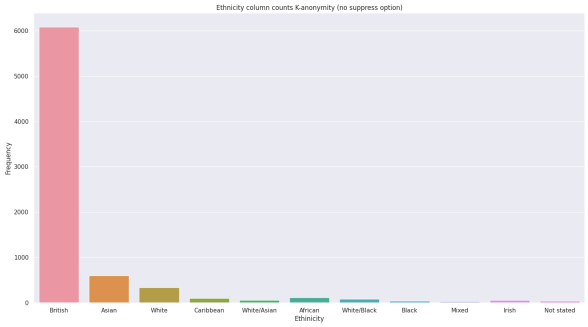

Fig. 6: Counts of *Ethnicity* column with k-anonymity and no suppression.

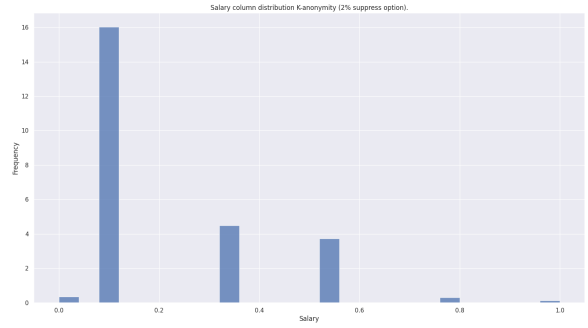

Fig. 7: Distribution of *Salary* column with k-anonymity and 2% suppression.

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 1: Overall and per column distortions for mock data set.

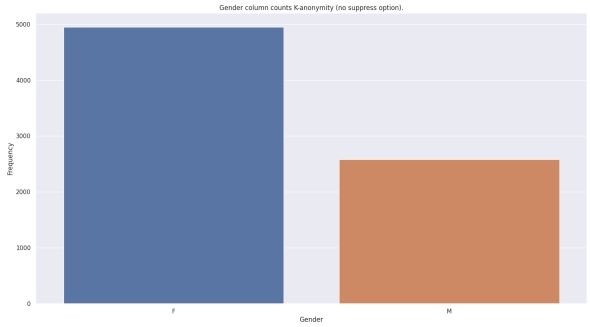

Fig. 8: Counts of *Gender* column with k-anonymity and 2% suppression.

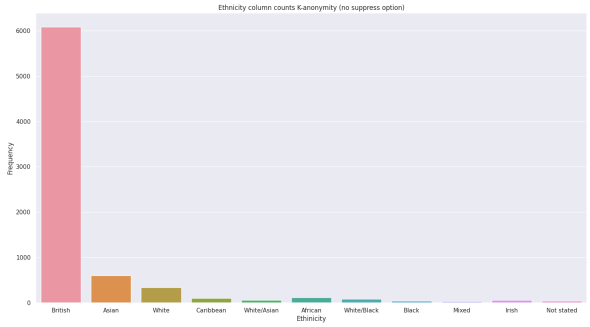

Fig. 9: Counts of *Ethnicity* column with k-anonymity and 2% suppression.

Figures 10, 11 and 12 shows the distorted columns distributions for the Q-learning algorithm.

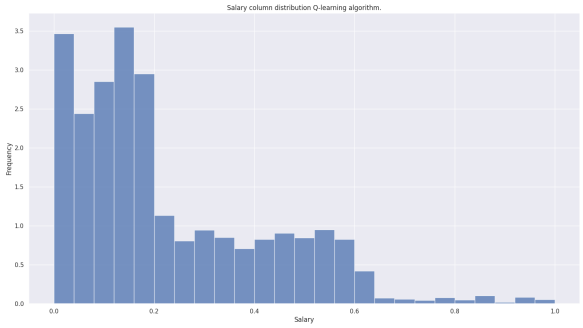

Fig. 10: Distribution of *Salary* column with Q-learning.

Figures 13 and 14 show the running rewards average and the running total distortion average over the training episodes for the Q-learning algorithm.

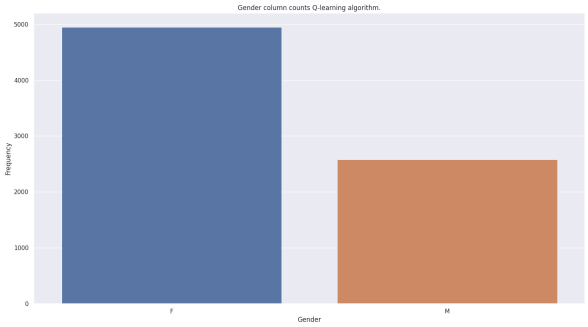

Fig. 11: Counts of *Gender* column with Q-learning.

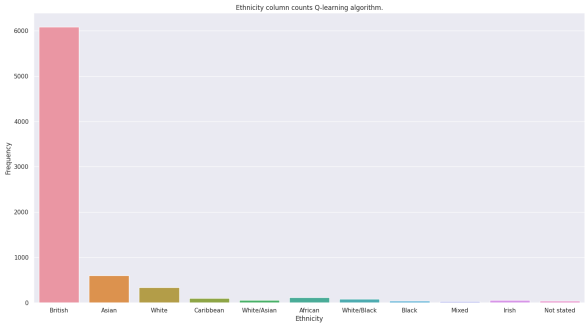

Fig. 12: Counts of *Ethnicity* column with Q-learning.

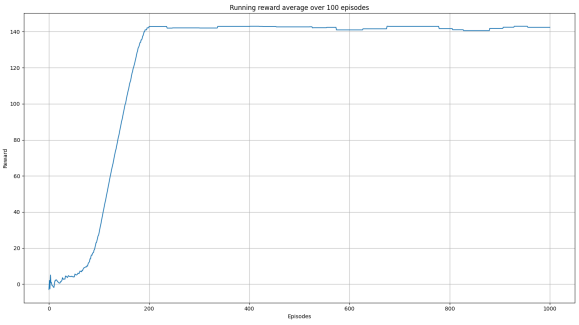

Fig. 13: Q-learning running average of rewards

Similarly, figures 17 and 18 show the running rewards average and the running total distortion average over the training episodes for the semi-gradient SARSA algorithm.

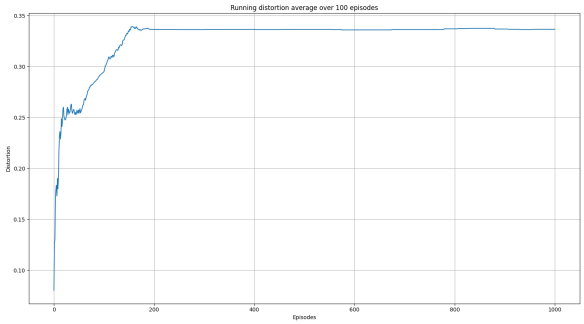

Fig. 14: Q-learning running average of total distortion

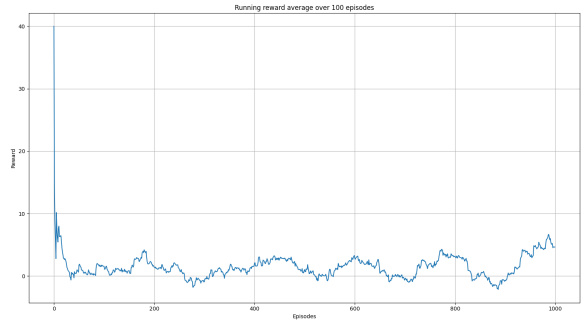

Fig. 15: Semi-gradient SARSA running average of rewards

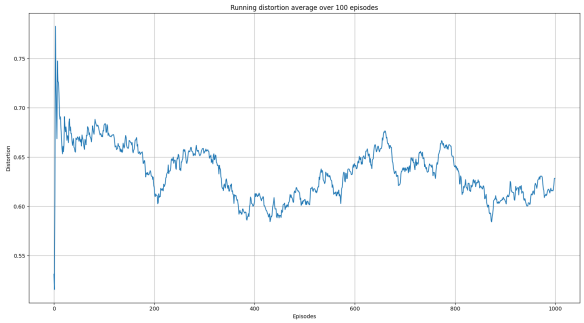

Fig. 16: Semi-gradient SARSA running average of total distortion

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.

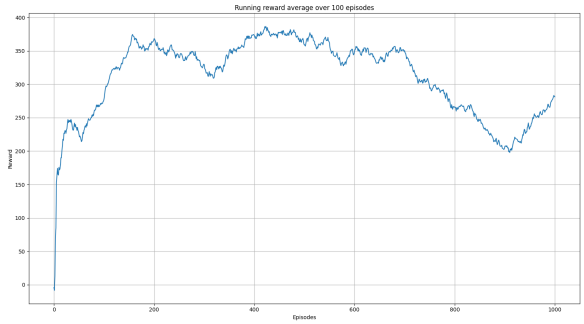

Fig. 17: A2C running average of rewards

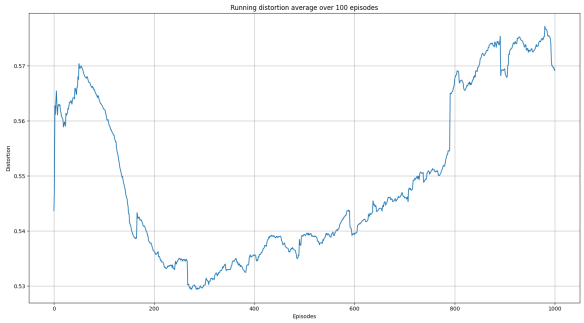

Fig. 18: A2C running average of total distortion

The simple Q-learning algorithm exhibited elements of learning as shown in figures 13 and 14. Moreover, the Q-learning algorithm resulted in an anonymized data set similar to the data set that is achieved when using the k-anonymity algorithm without suppression. The two algorithms differed in how they treat the *Salary* column.

One of the disadvantages of our methodology is that currently it requires to keep two copies of the data set; one that is not transformed and another that it undergoes transformation.

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 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. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Proceedings of The 33rd International Conference on Machine Learning. pp. 1928–1937. PMLR (2016)
8. Prasser, F., Kohlmayer, F., Lautenschlager, R., KA, K.: Arx-a comprehensive tool for anonymizing biomedical data. In: AMIA Annual Symposium proceedings. pp. 984–993. AMIA (2014)
9. Schulman, K., Moritz, P., Levine, S., Jordan M, A.P.: High-dimensional continuous control using generalized advantage estimation.
10. Sutton, R.S., Barto, A.G.: Reinforcement Learning. An Introduction. The MIT Press (2020)
11. Sweeney, L.: k-anonymity: A model for protecting privacy. International journal of uncertainty, fuzziness and knowledge-based systems **10**, 557–570 (2002)
12. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King's College Cambridge (1989)