**NeurIPS Hide-and-seek Privacy Challenge documentation questionnaire**

**Team name**

|  |
| --- |
| Golden\_Fleece |

**Submission filenames(s)**

|  |  |
| --- | --- |
| Hider | hider\_genetic\_bounds4p5\_noise0p2.zip |
| Seeker |  |

**What class of algorithms does your solution belong to?** (e.g. GANs, VAEs, noise-injection, nearest neighbor, etc.)

|  |  |
| --- | --- |
| Hider | Genetic + Adversarial Training |
| Seeker |  |

**Describe your algorithm in one sentence** (e.g. “Noise is added to the original data and then this data is returned.”)

|  |  |
| --- | --- |
| Hider | Apply the genetic algorithm to find synthetic data that meets the utility threshold and has maximal noise |
| Seeker |  |

**Describe your algorithm in words** (e.g. “Noise is drawn from a Gaussian distribution, with mean 0 and variance s, where the dimension is determined by the size of the dataset. This noise is added to the original data to produce a noisy version of the dataset and this noisy dataset is then returned as the synthetic data.”)

|  |  |
| --- | --- |
| Hider | For each user, create several candidates for their synthetic data. Add noise to these candidates, and combine the best ones that meet the utility threshold. Loop this process until either we reach some max number of iterations, or all of the candidates fail to meet the utility threshold (in which case the previous candidates that passed the utility threshold are used). |
| Seeker |  |

**Specify any loss functions used** (e.g. “No loss functions used.”)

|  |  |
| --- | --- |
| Hider | No loss functions used, except in our adversarial models, which are based on the utility metric models so they are trained with RMSE. |
| Seeker |  |

**Specify any hyperparameters and how they are optimized (or preset values)** (e.g. “The noise size, s, is set to 0.1.”)

|  |  |
| --- | --- |
| Hider | * Cluster sizes: This determines how many times we run the genetic algorithm – to keep runtime low, we shrink the number of clusters. We set this to 500. * Number of feature predictors used in adversarial training: We set this to 20. In general, having more ensures that we pass the true utility threshold in the competition * Max iterations for the genetic algorithm: We set to 10, to keep run times within bounds * Noise increments: We set to 0.2, which determines how much noise we add each iteration of the genetic algorithm, * Population size: We set this to 5, meaning each user data may only have 5 candidates in the genetic algorithm. * Population selection size: We set this to 2, meaning that at the end of each iteration of the genetic algorithm, we pick the two best synthesized data candidates for a user * Maximum number of failures: We set this to 5, which determines how many iterations we allow all candidates to fail.   Most of these hyperparameters are made based on run-time requirements (i.e. population size), but some are based on baseline seeker performance (i.e. having higher noise increments means that at the end of the max iterations, we have more noise). |
| Seeker |  |

**Specify any pre-trained models used by your algorithm** (e.g. “None.”)

|  |  |
| --- | --- |
| Hider | None. Even the utility models are actually only trained when our solution is run. |
| Seeker |  |

**Pseudo-code for your algorithm**

e.g. **Inputs:** Dataset, D, random seed

**Hyperparameters:** s (default 0.1)

1. Determine dataset dimension: n x d x T

2. Draw N ~ N(0, s), an n x d x T dimensional Gaussian

3. Return D + N

|  |  |
| --- | --- |
| Hider | **Inputs:** Dataset D  **Hyperparameters:**  c (# of clusters), p (population size) # for brevity, just consider these two.   1. Cluster all users based on their data using KMeans to create c clusters 2. Select a user from each cluster, call these the cluster representatives 3. Train utility models (feature prediction, one-step-ahead) on real data. 4. For each cluster representative, apply a genetic algorithm to their data: 5. Create an initial population of size p of synthesized data (data + some small amount of noise)  Check if this population passes the fitness test (does prediction using the utility models ensure MSE is within a threshold compared to the original data?)  Loop until we reach max iterations or all candidates fail X number of times:  Cross over the candidates that pass the fitness test (combine their feature data)  Mutate each candidate (add noise incrementally each iteration)  Check if the candidates in this generation pass the fitness test 6. For each cluster representative, take the noise N added to their data and apply it to all users in the same cluster 7. Return the original dataset D with the noise N added. |
| Seeker |  |

Finally, alongside this document **please also submit a commented version of your code**. Please include:

- Docstrings for each new class/function defined

- Inline comments for your main function/class

The goal of these comments is to tie the code to the description you have provided here. Please do not alter the actual content of your code - only add comments/docstrings.

**Submitting your documentation and commented code**

Please submit your commented code within a .zip or equivalent file type (1 file per solution), and share it with us as an attachment alongside this Word doc.

You can send these via email (to [nm736@cam.ac.uk](mailto:nm736@cam.ac.uk); [james.jordon@wolfson.ox.ac.uk](mailto:james.jordon@wolfson.ox.ac.uk); [es583@cam.ac.uk](mailto:es583@cam.ac.uk)) or DM James Jordon/Evgeny Saveliev on Slack (you can join the workspace [with this URL](https://join.slack.com/t/hideandseekpr-fbc8582/shared_invite/zt-k2h9xye8-RQNen128uXIG2TRsLa_ppA)).