184.702 Machine Learning – Exercise 3.1 Advanced topics in security / privacy of Machine Learning

Note: please first read the generic introduction document for the 3rd exercise!

Topic 3.1: Advanced topics in security / privacy of Machine Learning

You need to choose only one topic, don't be scared of the long document! :-)

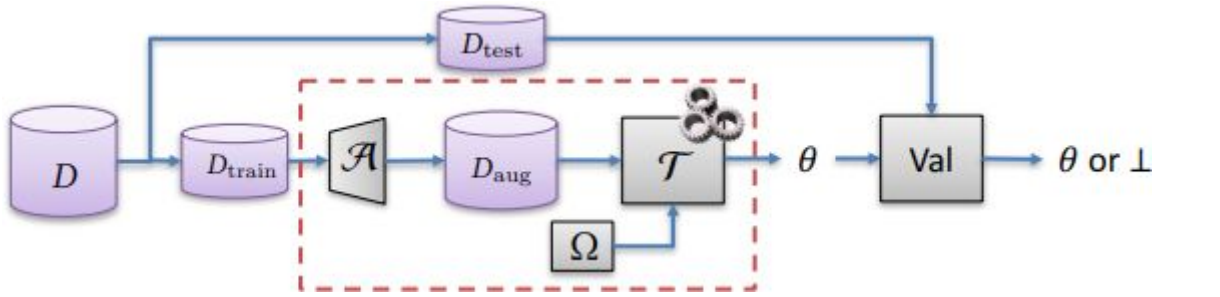
The aim of this type of exercise is to work in detail with one of the emerging topics in security & privacy of machine learning. Some of the topics are rather experimental as they are novel or based on very recent results, and the exact outcome can not always be predicted. Thus, the main aim is not to achieve perfect results, but to gain an interesting learning experience. We will grade your work based on your effort, and on the originality of your approach.

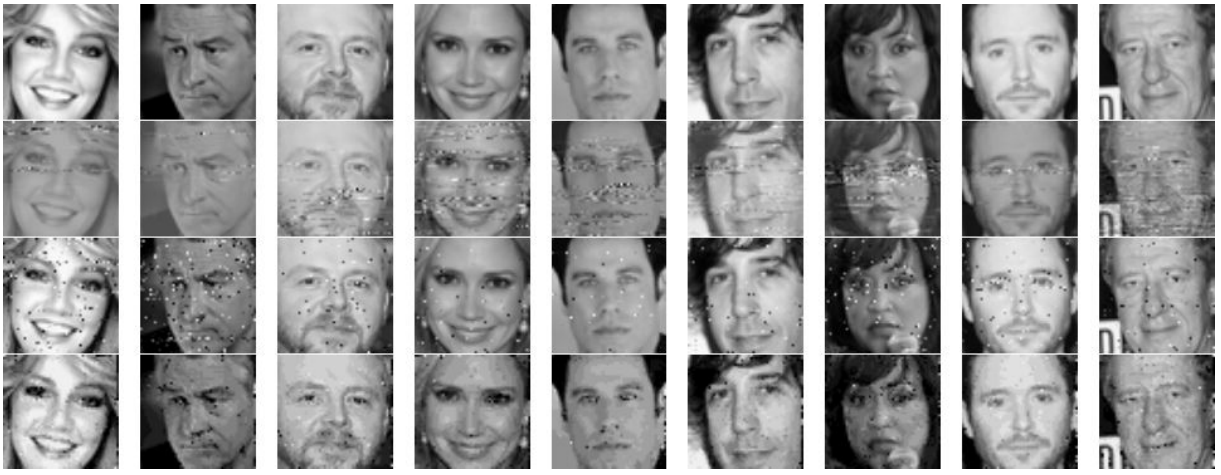
Adversarial Machine Learning

Some datasets that are mentioned in the rest of the topic description

- Yale Face Database http://vision.ucsd.edu/content/yale-face-database
- Labeled Faces in the Wild, http://vis-www.cs.umass.edu/lfw/, using the task for face recognition. See also https://scikit-learn.org/0.19/datasets/#the-labeled-faces-in-the-wild-face-recognition-dataset
- PubFig dataset, http://www.cs.columbia.edu/CAVE/databases/pubfig
- PubFig83: http://vision.seas.harvard.edu/pubfig83/
- AT&T / Olivetti Faces (https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, https://scikit-learn.org/stable/datasets/index.html#olivetti-faces-dataset)
- MNIST: http://yann.lecun.com/exdb/mnist/, https://scikit-learn.org/0.19/datasets/mldata.html
- CIFAR (10, 100): https://www.cs.toronto.edu/~kriz/cifar.html, https://github.com/ivanov/scikits.data/blob/master/datasets/cifar10.py
- FashionMNIST: https://github.com/zalandoresearch/fashion-mnist
- The German Traffic Sign Recognition Benchmark (GTSRB), http://benchmark.ini.rub.de
- ImageNet: http://image-net.org/download

Topic 3.1.1.1: Data exfiltration





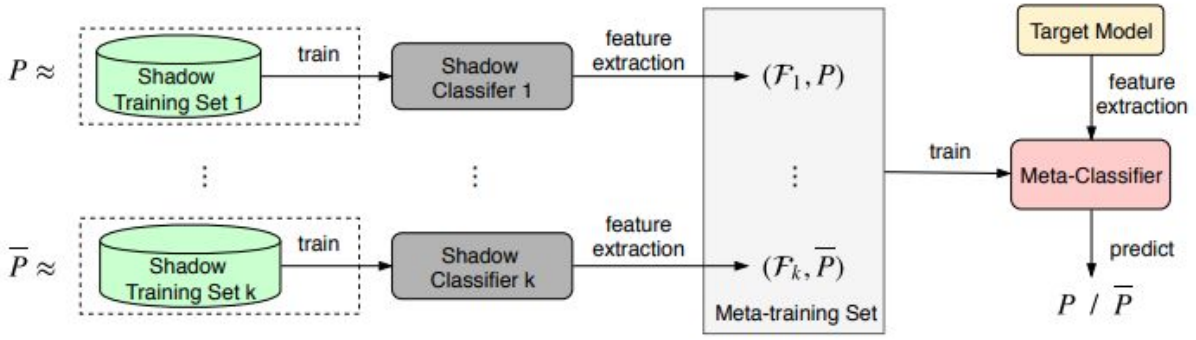
Data exfiltration tries to utilise a machine learning model to leak information about the training data to the end-user of the model. This can be interesting for an attacker when the model gets trained on private training data, and there is no other means to leak information about the data to the outside (the attacker), than via the model.

In this task, you shall recreate experiments either in a white-box or black-box settings as described in the paper by *Song et al: Machine Learning Models that Remember Too Much. CCS 2017*

You can perform your experiments on the same dataset as the original authors, or choose one of the ace-recognition datasets (AT&T faces, Yale faces, PugFig, LFW, ...)

Topic 3.1.1.2: Property inference

Property inference tries to infer (generic) properties of a training data set, from a trained machine learning model. This is in most cases global properties the model owner did not intend to share.



Your task is to perform experiments similar to the ones presented in **one of the papers** below, on datasets of your choice (can be the same / similar as in the paper)

Papers:

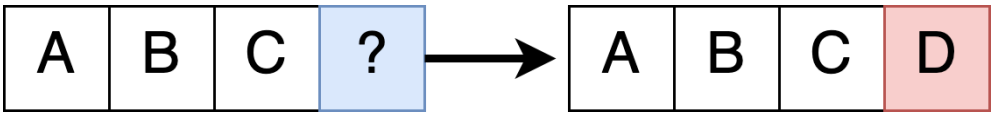
- Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, Nikita Borisov. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations, https://dl.acm.org/doi/10.1145/3243734.3243834
- U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft, “Blurme: Inferring and obfuscating user gender based on ratings”. ACM Conference on Recommender Systems, (RecSys). 2012.

Topic 3.1.1.3: Attribute Inference (disclosure) from ML Models

(note: this topic is rather experimental, and you will, as for all the other topics, graded on your effort, not necessarily on a (positive) outcome, which might not be achievable).

One type of information disclosure is called **attribute disclosure**: it consists of **inferring the value of an attribute** (e.g., ethnicity) **that was hidden** (i.e., not directly shared by the user).

Attribute disclosure is independent of identifying a specific record in a database - inferring additional information about a user from data shared by **other users**. This can be sensitive information about a user such as ethnicity or political affiliation, inferred by mining available data.

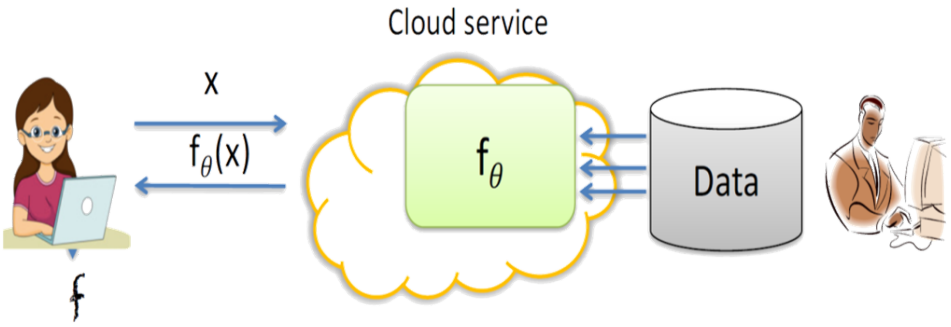


While attribute disclosure until now mostly considered the setting where a released **data set** is the base for inference, similar attacks might be possible from a **released machine learning model**, which invariably encodes some information about the training data.

Your task in this exercise is to test if this is possible for certain machine learning models. Given an incomplete sample X and a trained model f(X), is it possible to infer the unknown values of X? You can first assume that X (with its full information) was also used in actually training the model, but further testing whether it would be also possible for an X not used in the training. One simple approach would be testing multiple values to “impute” the missing value, and observe the response from the model to that.

Using models that generally overfit to the training data seems more promising. As such, a neural network, or also a decision tree, might be good candidates.

Topic 3.1.1.4: Model Stealing / Extraction



Model "stealing" means to obtain a trained model from a source that generally only provides a "prediction API", i.e. a means to obtain a prediction from the model when providing a specific input, but does NOT disclose the actual model (i.e. "ML-as-a-service"). This might be because the model may be deemed confidential due to their sensitive training data, commercial value, or use in security applications. A user might train models on potentially sensitive data, and then charge others for access on a pay-per-query basis.

In model stealing/model extraction, an adversary with black-box access (but no knowledge of the model parameters or training data), aims to duplicate the functionality of the model.

In this task, you shall experiment with these model stealing attacks for two different types of classifiers (can be from the paper) on three different datasets (two shall be different than in the paper, one to be used as validation that you achieve similar results). You can either use a model available locally (as black-box, i.e. just for querying & confidence values), or one of the mentioned online services. While the model is "extracted", record how the accuracy of the duplicated model changes, and how many queries (and how much time) it takes to train the model; plot these, in similar fashion as in the paper, for each dataset/classifier.

Regarding implementation, you can use one of the types of model stealing attacks listed below:

A. Type 1: Attacks, described in *Tramèr et al. Stealing Machine Learning Models via Prediction APIs. USENIX Security 2016.*

- (https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_tramer.pdf)
    - Implementation: https://github.com/ftramer/Steal-ML
- Pick one of the following types
    - Equation-solving attack for (M) LR or MLP
    - Path-finding attack for DT
    - Lowd-Meek for Linear Binary Models
    - Retraining attacks for non-linear SVM or NN

B. Type 2: Substitute training attacks

- Knockoff nets for CNN proposed by Orekondy et al. in Knockoff Nets: Stealing Functionality of Black-Box Models (https://arxiv.org/pdf/1812.02766.pdf)
    - Implementation: https://github.com/tribhuvanesh/knockoffnets

- Copycat networks for CNN proposed by Correia-Silva et al. paper Copycat CNN: Stealing Knowledge by PersuadingConfession with Random Non-Labeled Data (https://arxiv.org/pdf/1806.05476.pdf)
    - Implementation: https://github.com/jeiks/Stealing_DL_Models
- Active Thief for CNN or RNN proposed by Pal et al. in ACTIVETHIEF: Model Extraction UsingActive Learning and Unannotated Public Data (https://ojs.aaai.org//index.php/AAAI/article/view/5432)
    - Implementation: https://bitbucket.org/iiscseal/activethief/src/master/

3. Type 3: Cryptanalytic attack on ReLU NN

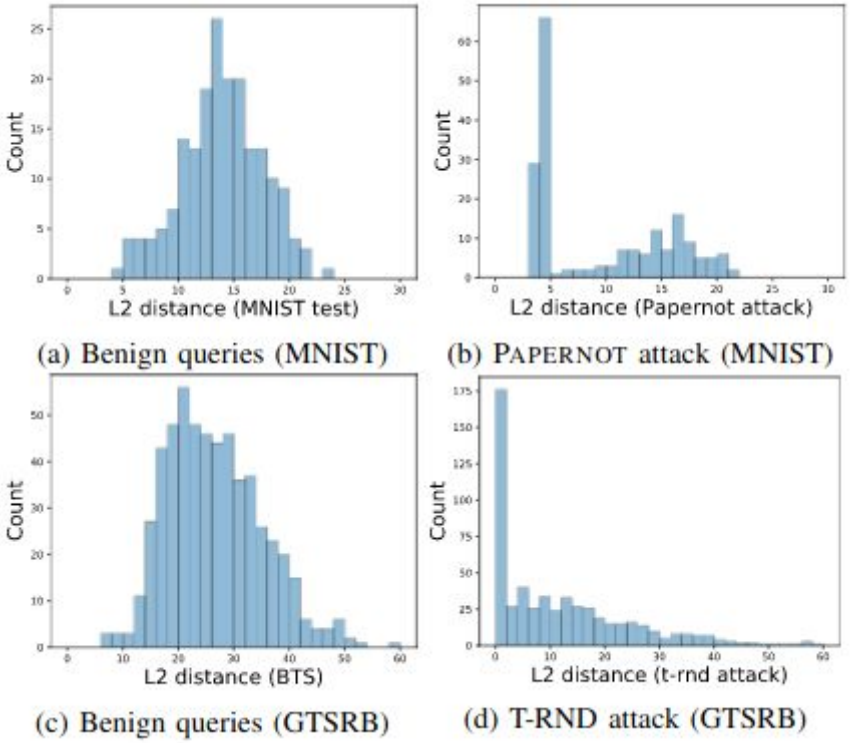
- Cryptanalytic Extraction of Neural Network Models by Carlini et al. (https://arxiv.org/pdf/2003.04884.pdf)
    - Implementation: https://github.com/google-research/cryptanalytic-model-extraction

You can also use the implementation(s) available by the IBM Adversarial Robustness Toolkit: https://github.com/Trusted-AI/adversarial-robustness-toolbox/wiki/ART-Attacks#3-extraction-attacks

Topic 3.1.1.5: Detection of Model Extraction

The idea of this task is to analyze the difference between distributions of data from the initial problem domain, and data generated by the adversary, e.g. synthetic/adversarial samples. The following steps should be done:

- Pick a dataset and explore it's distribution. We propose to pick one of the following datasets: MNIST, FashionMNIST, PubFig83, GTRSB, CIFAR-10
- Use different techniques, such as
    - Adversarial image generation: FGSM, IGS, FGV, Deepfool, C&W, or JSMA (as e.g. in https://arxiv.org/pdf/1809.04913.pdf)
    - Data composition (https://elidavid.com/pubs/stealing-knowledge.pdf) or
    - Random noise inputs (https://arxiv.org/pdf/1912.08987.pdf)
- For each technique, explore the distribution of generated samples and compare it to the original data distribution. Also data from a non-problem-domain dataset can be analyzed
- Implement a detection (or use (or modify) an existing one such as described in https://github.com/SSGAalto/prada-protecting-against-dnn-model-stealing-attacks), to evaluate which adversarial/synthetic samples can be recognized by distribution analysis



(a) Benign queries (MNIST) (b) PAPERNOT attack (MNIST)

(c) Benign queries (GTSRB) (d) T-RND attack (GTSRB)

Topic 3.1.1.6: Watermarking ML/DL models

Sharing trained models has brought many benefits to development of ML and DL systems. However, training models is usually a task that requires vast resources, from data to time and computing power. **Watermarking** can help the owners of such models mark their property in order to trace unauthorised usage or redistribution. **The task of this exercise is to work with one watermarking technique**, either a white-box or a black-box watermarking technique, e.g. from the following techniques:

- White-box Approaches
  - Uchida et al.: Embedding Watermarks into Deep Neural Networks (https://dl.acm.org/doi/pdf/10.1145/3078971.3078974)
    - Implementation: https://github.com/yu4u/dnn-watermark)
  - Rouhani et al.: DeepSigns: An End-to-End Watermarking Framework for Ownership Protection of Deep Neural Networks (https://dl.acm.org/doi/10.1145/3297858.3304051)
    - Implementation: https://github.com/Bitadr/DeepSigns)
  - Feng et al.: Watermarking Neural Network with Compensation Mechanism (http://link.springer.com/10.1007/978-3-030-55393-7_33)
  - Wang et al.: RIGA: Covert and Robust White-Box Watermarking of Deep Neural Networks (http://arxiv.org/abs/1910.14268)
- Black-Box Approaches
  - Merrer et al.: Adversarial Frontier Stitching for Remote Neural Network Watermarking (https://arxiv.org/pdf/1711.01894.pdf)
    - Implementation: https://github.com/dunky11/adversarial-frontier-stitching)
  - Adi et al.: Turning Your Weakness into Strength: Watermarking Deep Neural Networks by Backdooring (https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-adi.pdf)
    - Implementation: https://github.com/adiyoss/WatermarkNN)
  - Zhang et al.: Protecting IP of Deep Neural Networks with Watermarking (http://dl.acm.org/citation.cfm?doid=3196494.3196550)
    Either one of the proposed trigger images (unrelated, content or noise).
  - Guo et al.: Watermarking deep neural networks for embedded systems (https://dl.acm.org/doi/10.1145/3240765.3240862)

Either: implement the chosen technique or utilize an open source implementation.

In both cases: in order to demonstrate the success of the chosen watermarking technique, evaluate experimentally the following three requirements. You can use the experiments from the paper of the chosen technique as a guidance.

- **effectiveness**: after embedding the watermark, it also can be successfully extracted and verified, thus the model owner can claim ownership in case of a threat event. Analyze if the chosen technique satisfies this requirement.
- **fidelity**: the watermark embedding has only minimal influence on the model's performance. Compare the model's accuracy before and after embedding the watermark.

- **robustness**: the watermark should be robust against various types of attacks, e.g. fine-tuning, pruning, transfer learning. In your experiments, perform **one** attack and analyse the robustness of the watermarking technique with respect to this specific attack.

Depending on the choice of either a black-box or white-box watermarking technique, consider the following settings for your task:

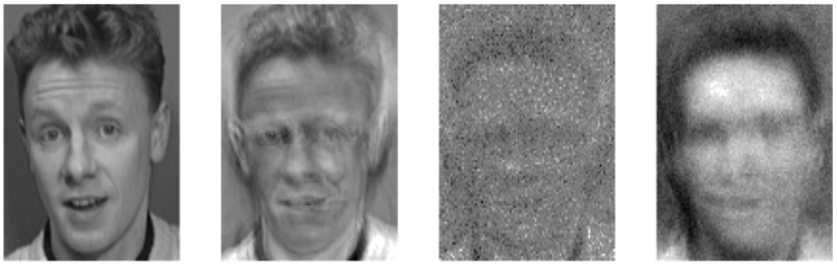
White-Box: implement the method on two different architectures: GoogleNet, SqueezeNet or Densenet, and one dataset (either MNIST, FashionMNIST, CIFAR-10, or ImageNet). Compare the results between those two architectures and conclude where the differences could arise from.

Black-Box: implement the method on one architecture (one of GoogLeNet, SqueezeNet, DenseNet), and on two different datasets (two out of MNIST, FashionMNIST, CIFAR-10, or ImageNet). Compare the results between those two datasets and conclude where the differences could arise from.

If you chose to implement a method for which no code is provided, you just need to show it only on one architecture and one dataset.

In TUWEL in the topic registration, specify the architecture(s) and dataset(s) used in the experiments. In the report describe your methodology in detail, explain your findings and provide the source code of your implementation.

Topic 3.1.1.7: Model Inversion Attack



Given access to a model, an ***inversion*** attack tries to extract the training data used for obtaining that model. A prominent example is for a face recognition system, where the model tries to identify to which of the known users a newly taken image (e.g. from an access control camera) corresponds to. This means that each class in the dataset corresponds to one specific individual, and the training data for each class is generally a number of images of that individual (taken with potentially different angles, zoom, light, background, and appearance such as hair style, ...).

For the attack, in most cases, this means picking a specific class, and then trying to ***generate*** input patterns that return the highest confidence of being classified as that class. This normally means access to the model (also in white-box, i.e. being able to read the model parameters; potentially building on top of a model stealing/extraction attack, but you can assume to have white-box access).

You can chose two different versions of this task

A. Extending an existing implementation provided by us, based on the paper by Fredrikson et al: "Model inversion attacks that exploit confidence information and basic countermeasures" (http://www.cs.cmu.edu/~mfredrik/papers/fjr2015ccs.pdf), to other datasets. We will provide you with the code, and working examples on a number of datasets. Use two of the following datasets:
    - Yale Faces
    - A subset of Labeled Faces in the Wild
    - A subset from the PubFig dataset
    - A subset from PubFig83
    - Any other data set for face recognition (except the original AT&T / Olivetti Faces)
        - Your subset should contain around 20-30 of the most-frequent people, as the task is very difficult for a larger number of classes.

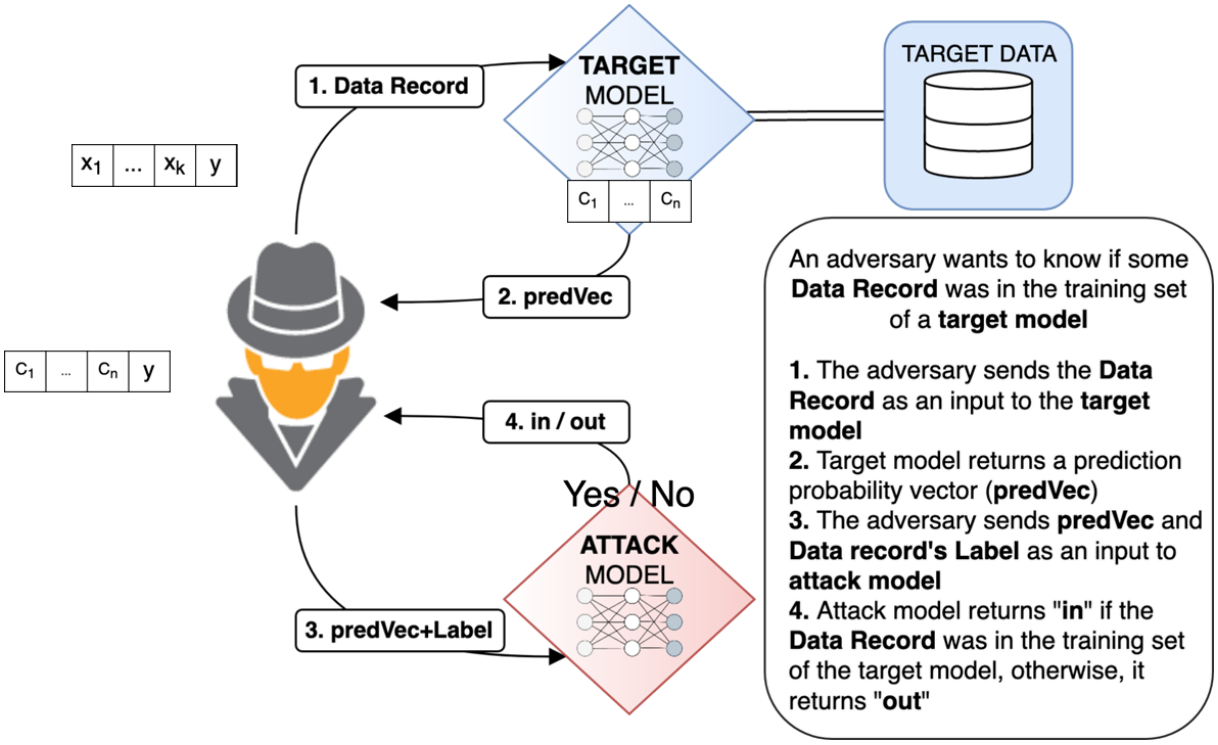
    and the following models:
    - A softmax layer
    - A simple, fully-connected network with 1 hidden layer
    - A very simply CNN of your choice

    Try also on-purpose overfit / underfit models, to see how that affects the results of the inversion.
B. Try a method from a more recent paper, e.g. https://arxiv.org/abs/1911.07135, which combines generative adversarial examples to improve the inversion process, and try it on a different dataset. A first step is to recreate the original experiments. Then, extend the experiment to one of the additional datasets listed above for Variant A

In your evaluation, **compare how well the inversion works** for each of the individuals, and also evaluate what is the **"cost" of generating** the images – i.e. how many iterations of generating and asking the model are required, and how is the efficiency of this (in runtime). You can also evaluate the quality of the generated images by computing a numerical similarity to the target image (e.g. the most representative of the training images per class).

Topic 3.1.1.8: Membership Inference

Membership inference tries to infer information from a learned ML model on the data samples that were used for training the model. In terms of released information, it is one of the weakest forms of attack on the confidentiality of the data samples, with the goal of revealing whether a sample was used for training a model, or not. This can be disclosive on an individual, e.g. if one can then infer that the individual might suffer from a certain disease.



In general, an attacker obtains a data set similar (or even identical) to the one used to train the target model, and trains a number of ***shadow*** models on that data; these should behave very similar to the original target model, and are used to train the ***attack model*** to distinguish between instance from that were inside or outside the training data.

There are two variants of this task:

A. Detailed evaluation

This task should investigate the success of this attack, respectively what are the conditions that make this attack successful. To this end, try the attack on several datasets with different characteristics, e.g. with different number of classes, and with differently trained models, e.g. those that on purpose overfit, and some that rather underfit, as well as those that should be trained with the “optimal” fitting amount of training. You may target neural networks, or also other types of models. The likely easiest version is to evaluate your solution on artificially created data, where you can vary the properties (like number of classes, etc..).

B. Unsupervised approaches to membership inference

The above mentioned shadow models are used to have training data for the attack model - this needs to have a ground truth on whether a sample was in the training data (of the shadow model that shall simulate the target (real) model), or not. Another approach could be to perform an

unsupervised attack in a grey-box setting: given a number of responses (including the full prediction vector / activations from the last layer) from the target model, try to cluster these into two clusters, hoping that this corresponds to “in” and “out” clusters. Try multiple state-of-the-art clustering algorithms (don’t implement one on your own), and evaluate your clustering with actual ground-truth data (i.e. you know which data has been used to

We will provide you an implementation of the membership inference attack, thus please contact us.

- Main paper: Membership Inference Attacks Against Machine Learning Models, https://www.cs.cornell.edu/~shmat/shmat_oak17.pdf
- Other useful papers
    - Towards Demystifying Membership Inference Attacks, https://arxiv.org/abs/1807.09173
    - ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. https://arxiv.org/abs/1806.01246

Topic 3.1.1.9: Backdoor/poisoning Attacks & defence

Poisoning/backdoor attacks modify the training data to **embed a specific pattern** in some images (e.g. a yellow block on a STOP sign), and falsely label that image as a different class (e.g. as a speed limit 50 sign) to make the classifier learn this pattern for wrong predictions. These attacks generally work because a certain number of neurons in the network can be dedicated to learn these patterns, often because the number of neurons is larger than what is required to actually represent the pattern in the "clean" training data.

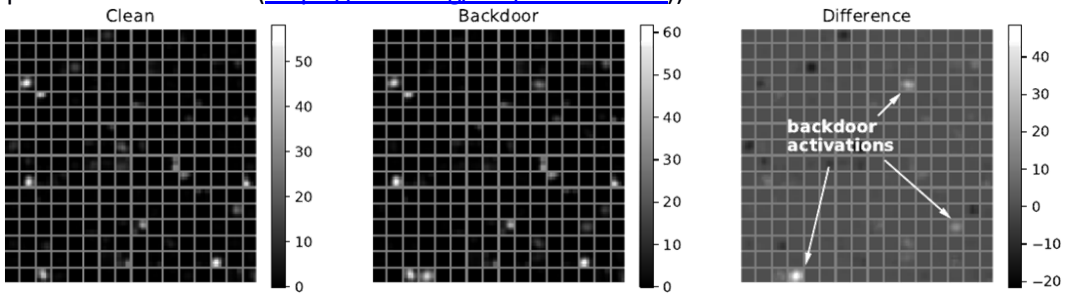
We will provide you with two datasets with manually prepared backdoor images, on the German Traffic Sign Recognition Dataset, and Yale Faces.

You shall then first attack the model with this poisoned data, and evaluate the performance of the attack, i.e. the clean and poisoned samples, comparing the clean samples to a baseline of an unmodified network. The evaluation shall be in a similar manner as to "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. Tianyu Gu, Brendan Dolan-Gavitt, Siddharth Garg. https://arxiv.org/abs/1708.06733)

You shall then either:

- Compare three of the defences as implemented in the IBM Adversarial robustness toolbox (https://github.com/IBM/adversarial-robustness-toolbox). The toolbox currently provides (https://github.com/IBM/adversarial-robustness-toolbox/tree/master/art/poison_detection) :
    - Detection based on activations analysis (Chen et al., 2018)
    - Detection based on data provenance (Baracaldo et al., 2018, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8473440)
    - RONI (https://people.eecs.berkeley.edu/~adj/publications/paper-files/SecML-MLJ2010.pd)

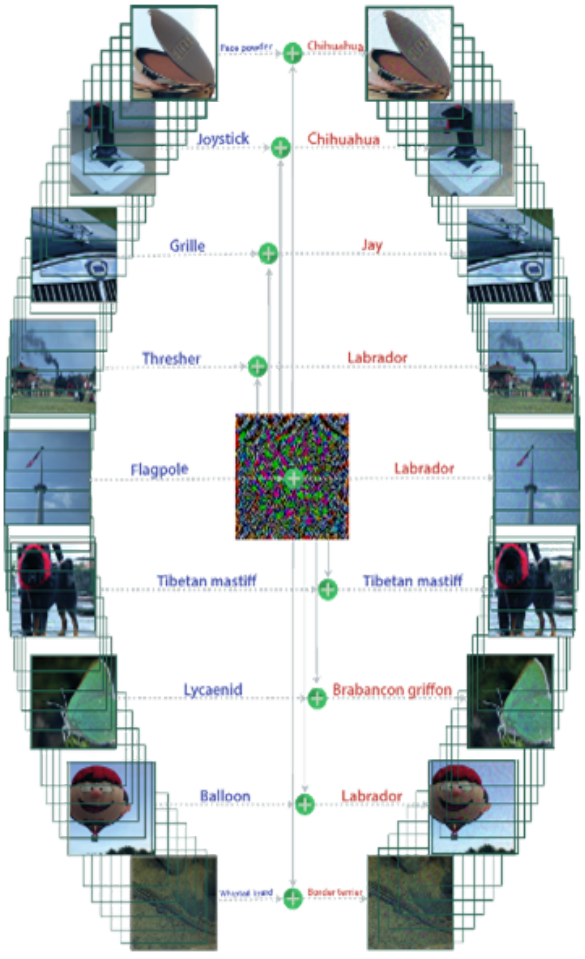
    - Further, it provides the ClusteringAnalyzer, which you might utilise for detection.
- Or implement another defence mechanism not yet provided in the toolbox. Your implementation should fit within the IBM Toolbox, i.e. use the existing base class etc, which will also help you in evaluating your approach. You can implement approaches such as
    - Fine-pruning (Liu et al: “Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks.” (https://arxiv.org/abs/1805.12185))



    - Outlier detection based, as in Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17, pages 3520–3532, USA, 2017. Curran Associates Inc.
    - Or another anomaly-detection based approach by Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. 2017 IEEE International Conference on Computer Design (ICCD), pages 45–48, 2017.

Topic 3.1.1.10: Model Evasion Attack

Evasion attacks are less targeted than backdoors, as they “just” try to avoid a specific class to be predicted.



In this task, you shall evaluate, e.g. on the AT&T Faces dataset or the Yale faces, or a subset of the PubFig/PubFig 83 dataset, how well different methods for generating adversarial images work, by measuring the distortion needed in a metric similarity, complemented with an occasional manual inspection of the perceived difference. Also measure the efficiency of the attacks. Specifically of interest are also methods that generate universal perturbations (i.e. not specific to one instance/image).

Then, try two simple methods for defending against these attacks (e.g. by adversarial training). You can utilise e.g. the IBM Adversarial Robustness Toolbox

(https://github.com/IBM/adversarial-robustness-toolbox), or also Google's Cleverhans (https://github.com/tensorflow/cleverhans) or Foobox (https://github.com/bethgelab/foolbox)

Privacy-Preserving Data Publishing

Topic 3.1.3.1: Generation and evaluation of structured synthetic datasets

Besides data sanitisation and secure computation, a further approach to work with data that contains sensitive information is to generate synthetic data. Synthetic data would be generated by obtaining some kind of model from existing data, and then letting this model generate new instances. The "model" can be very simple, e.g. it could estimate the distribution for each variable independently, and then generate synthetic data accordingly. Or, it could be more complex models that take correlations between variables into account – if the value of the n-th attribute is to be generated, it takes into account the already generated values for the attributes (1 .. n-1). More complex approaches use e.g. GANs to generate data. In general, the more information is obtained from the underlying dataset, the better synthetic data is generated.

ID	Birthdate	Gender	Balance
1	07/26/94	F	427.30
2	03/16/92		1001.42
3	01/17/87	M	33.33
4	08/16/94	M	54.33
5	09/04/82	M	5000.40
....	....	....	...

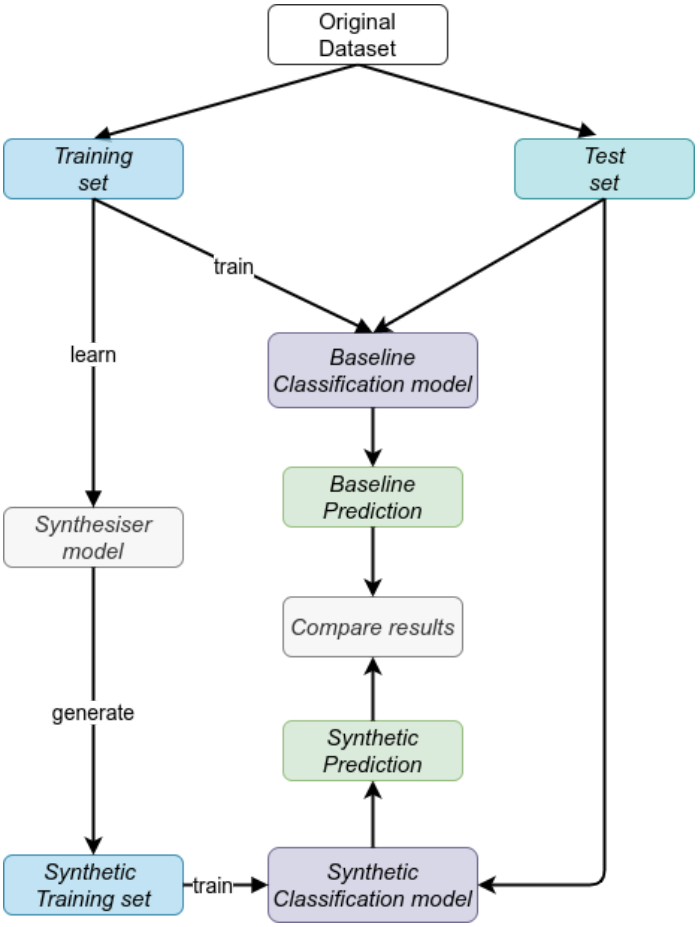
ID	Birthdate	Gender	Balance
1	11/26/88	M	4002.50
2	06/14/91	F	352.47
3	04/15/90	F	129.75
4	02/17/95		37.20
5	03/29/84	M	427.92
....	....	....	....

Your task is to utilise a synthetic data set generator, and evaluate how well that works as replacement data for training machine learning models. This task has two types you can choose from:

A. Use the Synthetic Data Vault with all three model types (copula, and the two GAN based ones, https://sdv.dev/SDV/user_guides/single_table/models.html), and compare the results on 3 of the following datasets:
   a. https://archive.ics.uci.edu/ml/datasets/Adult (https://archive.ics.uci.edu/ml/datasets/census+income)
   b. http://archive.ics.uci.edu/ml/datasets/banknote+authentication
   c. https://www.kaggle.com/rakeshrau/social-network-ads
   d. https://www.kaggle.com/c/titanic/data
   e. https://archive.ics.uci.edu/ml/datasets/iris

B. Use the synthpop and DataSynthesizer tool on 3 structured datasets that are **not** listed for the SDV topic above

From each dataset, take a part and use it as training data. On this training data, use the synthesizers to create a synthetic version of the model. Keep one part of the original data aside as test data, and use your model (trained on synthetic data) to predict the classes of the test data. Then compare these results with a model that you would train directly on the data used to estimate the synthetic data generator, to estimate the usefulness of the synthetic data, as shown in the Figure below.



Provide also an analysis e.g. of the confusion matrix, to test whether one can observe differences for certain classes or instances.

Links to tools:

https://github.com/DataResponsibly/DataSynthesizer
https://hdi-project.github.io/SDV/
https://cran.r-project.org/web/packages/synthpop/index.html

Topic 3.1.3.2.: Generation and evaluation of synthetic image datasets

This is a rather experimental topic, but maybe therefore even more interesting.

The idea is similar to the one above, but with the difference that you are not going to generate structured data, but image data. One approach could be similar to the one used in generative adversarial networks (GANs), though you can potentially also utilise a simpler approach.

Then in general, the approach shall be similar to the one for structured data – obtain a dataset of images, use a training set to learn the generator, generate data, learn a model from it, and predict the images on the test set. Compare this to the baseline of using the training set directly. You can test your approach e.g. on the traffic sign and fruits datasets linked above for the Deep Learning topic. Another evaluation would be on a medical dataset with a clear classification task.

As this topic is more experimental and novel than others, you can focus on the problem and implementation, and perform a less detailed evaluation than on the structured dataset. Also, we will honour your effort in the evaluation, and less your results.

For GANs in python, plenty of examples are available, e.g.
https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

One project that you can maybe draw inspiration from the medical domain is
https://www.fda.gov/MedicalDevices/ScienceandResearch/ResearchPrograms/ucm477418.htm

One data set to use (with a classification task) could be e.g.
https://www.kaggle.com/c/diabetic-retinopathy-detection/data

Topic 3.1.3.3.: Generation and evaluation of sequential synthetic datasets

This is a rather experimental topic.

The above discussed tools for synthetic data generation have been developed for processing structured relational data. One application scenario for synthetic data is however as well in sequential data, which DNA sequences are one specific instance of.

One such method has been described by Pratas et al and uses multiple competing Markov Models. First, a training DNA sequence is partitioned into non-overlapping blocks of fixed size. Each block is handled by a particular Markov Model. In the first phase, the statistics of the training data are loaded into the models. After this deterministic procedure has ended, the models are kept frozen. In a second stochastic phase, the synthetic sequence is then generated according to the learned statistics.

Your task is to recreate the implementation of the specified paper, and evaluate the quality of genetic data when comparing it to real sequences and evaluating it on predictive tasks.

Main paper: D. Pratas, C. A. C. Bastos, A. J. Pinho, A. J. R. Neves, L. M. O. Matos. DNA Synthetic Sequences Generation using Multiple Competing Markov Models. IEEE Statistical Signal Processing Workshop (SSP), pp. 133-136, 2011.

Topic 3.1.3.4.: k-anonymity and utility of anonymized datasets

k-anonymity is a property of a dataset that contains the same information for at least *k* individuals, for every distinct information. *k*-anonymous datasets can be obtained by a systematic generalization or suppression of the values of the dataset.

These methods reduce the amount of information contained; therefore *k*-anonymous datasets necessarily have lower utility compared to their originals. Utility can be measured in different ways, e.g. using metrics directly on data such as a loss measure, or by data performance on a particular machine learning task.

The aim on this task is to investigate the relationship between

- utility metrics (precision, sum of squared errors, entropy, granularity, ...) and
- classification performance (classification accuracy, F1 score, ...)

of anonymized datasets.

You should utilize at least 2 anonymization algorithms of your choice:

- Flash (within ARX tool; Java) – API (not GUI!): https://github.com/arx-deidentifier/arx
- SaNGreeA (Python): https://github.com/tanjascats/sangreea
- Mondrian (Python): https://github.com/qiyuangong/Mondrian
- implementations from UTD Anonymization Toolbox (Java): http://cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php?go=home
- Crowds (Python): https://pypi.org/project/crowds/
- sdcMicro (R): https://cloud.r-project.org/web/packages/sdcMicro/index.html
- some other open-source anonymization tool

The anonymization algorithms usually offer multiple anonymized datasets as a result. Having multiple options for anonymizing the dataset may have certain benefits, e.g. in a scenario where generalizing some attribute is preferred over the others. Therefore, besides calculating the utility for one optimally anonymized dataset, you shall also analyse the difference in utility between optimal and other solutions.

For your analysis you shall use 2 datasets that contain private or sensitive information (e.g. http://archive.ics.uci.edu/ml/datasets/Adult - don't use this example :-)). Choose datasets that differ according to some characteristics such as size and type of attributes. Some suggestions:

- Bone Marrow Transplant data
- Census Income (KDD)
- Credit Card Clients
- Drug Consumption Dataset
- ...

For most of the algorithms you may need to add your own implementation of utility metrics to evaluate the quality of anonymized data.

1. Precision: one minus the sum of all call distortions normalized by the total number of cells, where a call distortion is the ratio of the domain of the value in the cell to the height of the attribute's hierarchy (Definition 5 from [1])

2. Loss/granularity: sum of the normalized loss for each column, where the loss of the (numerical) column is the ratio of value interval of that column to the interval of original values (see details in Section 3.1. of [2])
3. Entropy: Section 3.2. of [3]

We have prepared the experimental setup you shall utilize in your analysis (2 Jupyter notebooks). You can find them in a zip file at TUWEL.

Specify in TUWEL at the topic registration which datasets will be used for the experiments. Write a report including all your findings, explain your methodology in detail and elaborate the conclusions.

References:

[1] Sweeney, L., 2002. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(05), pp.571-588. (https://desfontain.es/PDFs/PhD/AchievingKAnonymityPrivacyProtectionUsingGeneralizationAndSuppression.pdf)

[2] Iyengar, V.S., 2002, July. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 279-288). (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.2993&rep=rep1&type=pdf)

[3] Gionis, A. and Tassa, T., 2008. k-Anonymization with minimal loss of information. *IEEE Transactions on Knowledge and Data Engineering*, *21*(2), pp.206-219. (https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.929.4919&rep=rep1&type=pdf)

Topic 3.1.3.5.: k-anonymity: generalization vs microaggregation

k-anonymity is a property of a dataset that contains the same information for at least *k* individuals, for every distinct information. *k*-anonymous datasets are obtained by either:

- a systematic generalization or suppression of the values of the dataset (global, local)
- microaggregation of the clustered values

Global and local transformation are types of generalization/suppression approach. Global transformation is achieved by generalizing the entire attribute (column) to the same generalization level from its hierarchy. Local transformation, however, aims to reduce the value to the least necessary generalization level. The locally anonymized dataset may contain values from different levels of generalization within one column (more information and examples at [1]).

Microaggregation is an approach that does not require defining generalization hierarchies for each quasi-identifier, but rather clusters the rows based on similarity and then generates the anonymized values as a result of some aggregation function (e.g. mean for numerical attributes, median for categorical, ...) [2].

These methods reduce the amount of information contained; therefore *k*-anonymous datasets necessarily have lower utility compared to their originals. One way to measure utility is via data performance on a particular machine learning task.

The aim of this exercise is to compare the utilities of the anonymized datasets obtained by different approaches:

1. Global transformation
2. Local transformation
3. Microaggregation

To that end, define classification tasks you want to perform and choose at least 3 different types of classifiers. Calculate the performance on the original dataset and then perform the same tasks using the anonymized datasets. Compare performance of ML classifiers trained on anonymized data to the ones trained on original data and compare how different anonymization approaches influence the performance.

The following anonymization tools provide different types of anonymizations:

- Generalization/suppression:
    a. Flash (within ARX tool; Java) – API: https://github.com/arx-deidentifier/arx (global/local)
    b. SaNGreeA (Python): https://github.com/tanjascats/sangreea (local)
    c. Mondrian (Python): https://github.com/qiyuangong/Mondrian (global)
    d. implementations from UTD Anonymization Toolbox (Java): http://cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php?go=home (global)
    e. Crowds (Python): https://pypi.org/project/crowds/
    f. sdcMicro (R): https://cloud.r-project.org/web/packages/sdcMicro/index.html
- Microaggregation:
    a. (use) sdcMicro-microaggregation (R): https://cloud.r-project.org/web/packages/sdcMicro/index.html
    b. (adapt) SaNGreeA (Python): https://github.com/tanjascats/sangreea
        - To this end, utilize SaNGreeA algorithm for clustering the dataset into partitions of size at least *k*. Then modify the algorithm so that the anonymized values are microaggregated instead of generalized (i.e. for each

cell calculate the mean value of the range if it is a numerical attribute and median (most frequent) value if it is a categorical attribute)

- You can also try to find other open-source anonymization tools

For your analysis choose 3 datasets with privacy implications and different characteristics (number of instances, number of attributes, types of attributes, etc.).

Specify in TUWEL at the topic registration which datasets will be used for the experiments. Write a report including all your findings, explain your methodology in detail and elaborate the conclusions.

References:

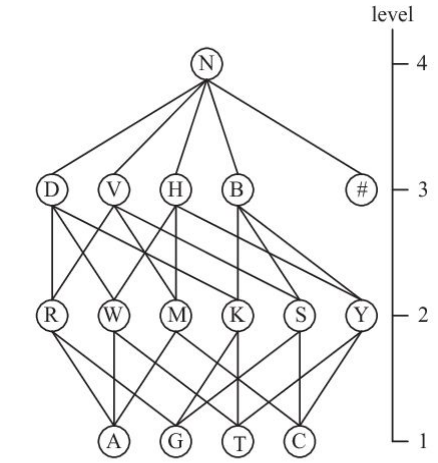
[1] https://sdcpractice.readthedocs.io/en/latest/anon_methods.htm

[2] J. Domingo-Ferrer and V. Torra, "Ordinal, Continuous and Heterogeneous k-Anonymity Through Microaggregation," *Data Min Knowl Disc*, vol. 11, no. 2, pp. 195–212, Sep. 2005, doi: 10.1007/s10618-005-0007-5.

Topic 3.1.3.6.: Genetic Data (DNA Lattice) Anonymization

The approach of DNA Lattice Anonymization (DNALA) has first been introduced in [1] and is applied to databases of DNA sequences. DNALA provides the possibility to prevent identification of DNA. For each DNA sequence, the technique finds **the most similar sequence** in the database and generalizes both to a common sequence. The distance between the sequences is computed according to the DNA generalization lattice in the figure below.

A = **A**denine	C = **C**ytosine
G = **G**uanine	T = **T**hymine
R = pu**R**ine	Y =p**Y**rimadine
S = **S**trong hydrogen	W = **W**eak hydrogen
M = a**M**ino group	K = **K**eto group
B = not A	D = not C
H = not G	V = not T
# = gap	N = i**N**determinate



For example, Adenine (A) and Cytosine (C) generalize to Amino Group (M), as can be seen by following the only arrows from A and C pointing to the same letter in the next level of the lattice. The function *gen(x,y)* returns the distance in the lattice between *x* and *y* and is defined by

$$gen(x,y) = 2level(z) - level(x) - level(y),$$

where *z* is the value *x* and *y* generalize to. E.g., we have $gen(A,C) = 2*2 - 1 - 1 = 2$. Subsequently, this function is used to define distances between whole DNA sequences.

Goal of this assignment: Read and study the original research paper by Malin. In particular, focus on understanding the DNALA method described in Section 3 and implement its core system (see Algorithm 2) in a programming language of your choice (preferred: python)

Primary reading:

B. A. Malin. *Protecting genomic sequence anonymity with generalization lattices.* Methods Inf Med., vol. 44, no. 5, pp. 687-692, 2005.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.2227&rep=rep1&type=pdf

Further readings (optional):

G. Li, Y. Wang, X. Su. *Improvements on a privacy-protection algorithm for DNA sequences with generalization lattices.* Computer Methods and Programs in Biomedicine, vol. 108, no.1, pp. 1-9, 2012. https://www.sciencedirect.com/science/article/abs/pii/S0169260711000459?via%3Dihub

S. Wan, M. Mak, S. Kung, *Protecting Genomic Privacy by a Sequence-Similarity based Obfuscation Method.* 2017. https://arxiv.org/abs/1708.02629

Topic 3.1.3.7.: Differential Privacy

Differential privacy is introduced by Cynthia Dwork [1] as a set of privacy-preserving mechanisms for publicly sharing information about the dataset by descriptive characteristics rather than sharing the information about the individuals in the dataset. Differential privacy is at first achieved with simple aggregate statistics and descriptive statistics such as mean function, maximum, histograms, etc., and recently the advances with differentially private machine learning models are achieved, as well.

Differential privacy is achieved by adding noise in the process. Depending on the phase where the noise is added we differentiate:

- Input perturbation - adding noise to the input before running the algorithm
- Internal perturbation - randomizing the internals of the algorithm
- Output perturbation - add noise to the output, after running the algorithm

The goal of this exercise is to compare these approaches for achieving differential privacy, from the privacy aspect and model performance aspect and analyse the tradeoff between privacy and performance. To that end, to achieve internal perturbation, utilize a differential privacy tool that provides differentially private versions of classification and clustering models (for Python: https://github.com/IBM/differential-privacy-library, for R: https://github.com/brubinstein/diffpriv). Familiarize yourself with the differentially private mechanisms used in the chosen tool. Secondly, apply an input perturbation and output perturbation based on the sensitivity method proposed by Dwork et al. in [1]. You may experiment with a model of your choice.

You shall analyse:

1. Input perturbation vs. internal perturbation vs. output perturbation approach on a chosen model; difference between the models' performances, and differences to the non-private model.
2. Trade-off between privacy and model performance for every approach.

Experiments from these papers [2, 3] may serve as an inspiration for the methodology of this exercise.

References:

[1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," in *Theory of Cryptography*, Berlin, Heidelberg, 2006, pp. 265–284. PDF: https://link.springer.com/content/pdf/10.1007/11681878_14.pdf

[2] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially Private Empirical Risk Minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011. PDF: https://arxiv.org/pdf/0912.0071.pdf

[3] K. Fukuchi, Q. K. Tran, and J. Sakuma, "Differentially Private Empirical Risk Minimization with Input Perturbation," in Discovery Science, Cham, 2017, pp. 82–90. PDF: https://arxiv.org/pdf/1710.07425.pdf

Further readings - differentially private ML:

[4] O. Sheffet, “Private Approximations of the 2nd-Moment Matrix Using Existing Techniques in Linear Regression,” arXiv:1507.00056 [cs], Nov. 2015.

[5] J. Vaidya, B. Shafiq, A. Basu, and Y. Hong, “Differentially Private Naive Bayes Classification,” in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Nov. 2013, vol. 1, pp. 571–576.

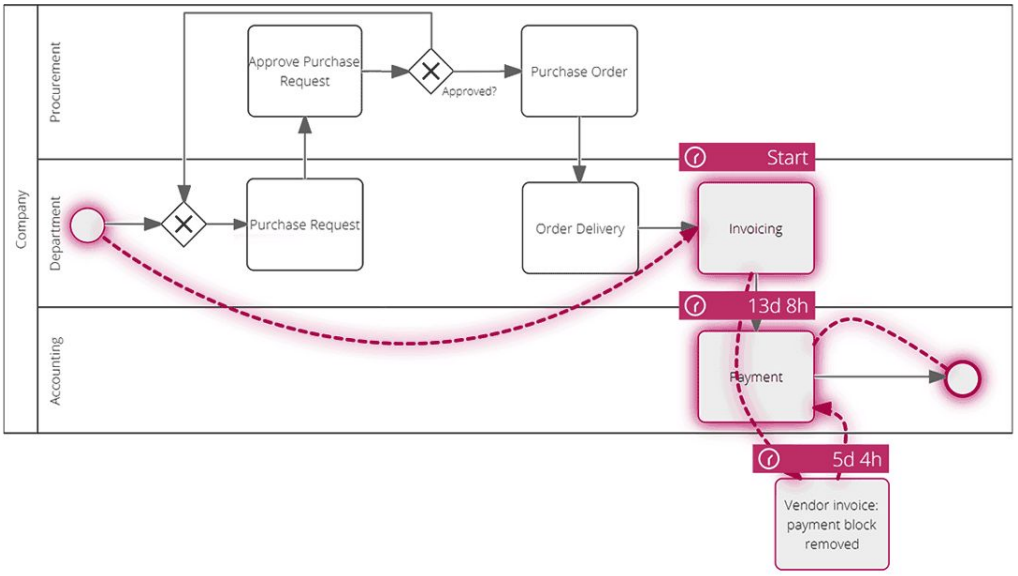
[6] H. Imtiaz and A. D. Sarwate, “Symmetric matrix perturbation for differentially-private principal component analysis,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 2339–2343.

Secure / Federated / Distributed Computation

Distributed computation in general addresses privacy / data protection issues by not centralising data that is initially already partitioned between several parties, and either computing functions securely w/o revealing the input (secure multi-party computation), or by exchanging only aggregated values, such as gradients or other model parameters (federated learning), or by computing on encrypted data (homomorphic encryption).

In this task, you shall evaluate the effectiveness and efficiency of these approaches. An evaluation on the quality of the models, as compared to having a model trained on all the different partitions of the data, shall be performed. - i.e. observe the effectiveness and efficiency on the centralised vs. the distributed approach.

Topic 3.1.2.1: Federated (Distributed) Learning for Process Mining



Process mining tries to learn the structure of a process from logs or other sources about previous executions of process. This is typically done on centralized data, i.e. with centralized learning. Collaborative learning of such a process model from different organisations can be an interesting setting in many cases. However, instead of centralising the data, which might not be an option due to confidentiality reasons, federated learning can become a viable approach.

In this task, you shall perform experiments for distributed process mining. To this end, take an existing benchmark dataset from e.g. https://data.4tu.nl/repository/collection:event_logs_real, and first perform process mining in a centralised way with an existing approach (e.g. using ProM, http://www.promtools.org/). Then, try to federate / distribute the process, by first distributing the data, and then the learning. Data distribution could be done in uniform (each subset is roughly similar) and non-uniform ways (some aspects of the data are present only at one node).

One approach for distributed learning you could follow is: “On the Feasibility of Distributed Process Mining in Healthcare”, 2019. https://link.springer.com/chapter/10.1007/978-3-030-22750-0_36

Topic 3.1.2.2: Federated learning for Image Data

Federated learning tries to learn one central model from data that is initially distributed, with the goal of achieving at least comparable predictive effectiveness (e.g. accuracy). In this task, you shall experiment with federated learning.

Utilising e.g. PySyft (https://github.com/OpenMined/PySyft), based on PyTorch, or TensorFlow federated (https://www.tensorflow.org/federated/), or simply your own setup, evaluate the effectiveness and efficiency of federated learning vs. the centralised learning on a small image database (e.g. AT&T faces, PubFig, CIFAR; if needed a subset of each) while learning neural networks. Specifically compare the difference in effectiveness when you vary some parameters, e.g. the number of nodes (from maybe 2-3, up to a higher number), or how often federation steps (sometimes called cycles) are performed.

Topic 3.1.2.3: Federated Learning on text data (or other sequential data)

In this task, you shall work with federated approaches for text data utilising RNNs (LSTM, GRU, …). Take a task of your choice (e.g. a text categorisation / classification task like the Reuters Dataset, or 20 newsgroups, etc..), and first establish a baseline with a state-of-the-art recurrent neural network, comparing your results to literature.

Then, simulate a federated setting by distributing the data on multiple sources, and compare your results to a centralised setting, similar to the setting in the topic above.

Topic 3.1.2.4: Federated Learning on relational data

In this task for federated learning, you shall work with "traditional" relational data (i.e. not sequential, not image, …). You shall pick two different datasets, which can be any dataset that you have e.g. used in the Machine Learning lecture in Exercise 1 or 2, and employ federated learning with two different types of algorithms on it. You are free to reuse existing implementations (like the ones mentioned above), or you can implement your own approach. For different algorithms, you can consider e.g. MLPs, but also SVMs, Naive Bayes, …

Simulate a federated setting by distributing the data on multiple sources, and compare your results to a centralised setting, similar to the setting in the topic above.

Topic 3.1.2.5: Secure Multi-party computation for Image Classification

Utilising e.g. the library mpyc for Python (https://github.com/lschoe/mpyc) and the implementation of a secure computation for a binarised multi-layer perceptron, first try to recreate the results reported in Abspoel et al, "Fast Secure Comparison for Medium-Sized Integers and Its Application in Binarized Neural Networks", i.e. train a baseline CNN to estimate a potential upper limit of achievable results, and then train the binarized network, as a simplified but still rather performant version, in a secure way. If needed, you can use a subset of the MNIST dataset.

Then, try to perform a similar evaluation on another small dataset, either already available in greyscale, or converted to greyscale, e.g. using (a subset of) the AT&T faces dataset.

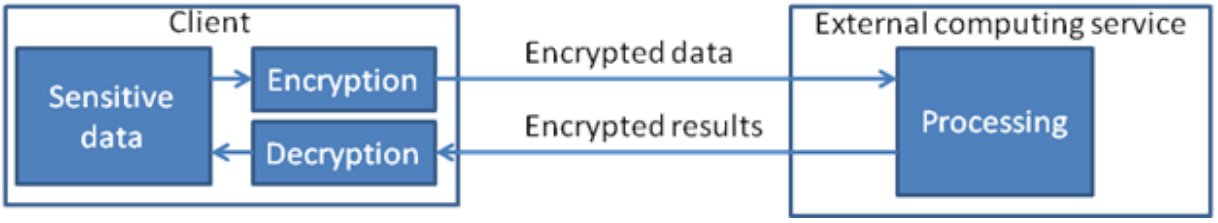
Specifically, evaluate the final result in terms of effectiveness, but also consider efficiency aspects, i.e. primarily runtime, but also other resource consumption.

Topic 3.1.2.6: Secure Multi-party computation for relational Data

In a similar setting as above (but with a machine learning algorithm of your choice), you shall perform secure multi-party computation on relational data. You shall pick two different datasets, which can be any dataset that you have e.g. used in the Machine Learning lecture in Exercise 1 or 2, and simulate a distribution of the data to multiple parties. Evaluate your approach as compared to a centralised setting, similar to as described in the topic above.

Topic 3.1.2.7.: Machine learning on encrypted data (Homomorphic Encryption)

One approach to conceal sensitive information in data is to only provide encrypted data to the machine learning algorithm, and directly compute on encrypted data. The returned model is also encrypted, but can be decrypted with the same key as the data. Thus, only the owner of the data can utilise the model, while the computation can be outsourced to an untrusted third party.



To achieve computation on encrypted data, homomorphic encryption schemes are utilised. These will achieve operations on the encrypted data without any loss, but are normally rather slow.

You can utilise one of the existing frameworks providing HE, such as the Intel HE-Transformer, or the Microsoft SEAL Library, to implement a simple HE-based machine learning model (logistic regression, Perceptron, k-NN, Naive Bayes, or similar). This means adapting existing implementations, depending on the case exchanging computations with approximations.

You shall then compare an implementation on unencrypted data with the encrypted version. The dataset can be a small toy dataset, e.g. the IRIS dataset or even smaller.

Tool support is available, e.g.

- Intel's https://github.com/NervanaSystems/he-transformer

- Microsoft's https://github.com/Microsoft/SEAL
- A list at https://github.com/jonaschn/awesome-he
- Some others, like https://github.com/shaih/HElib or https://github.com/tfhe/tfhe

If you chose this topic, register in TUWEL which algorithm you would like to implement.