UCLouvain

epl

École polytechnique de Louvain

Homomorphic Encryption for Privacy-Friendly Augmented Democracy

Author: **Matthieu Brabant**
Supervisors: **Olivier Pereira, Pierrick Méaux**
Readers: **Marc Joye, Michel Verleysen**
Academic year 2020–2021
Master [120] in Computer Science and Engineering

Abstract

"We could have a senate with as many senators as we have citizens."

We are not in early Athens, but well and truly in the 21st century, and even more. Augmented Democracy is a proposal to assist citizens in their democratic duty through a digital twin. People do not have the time, interest or expertise to be experts in every field. Open-data government portals are barely used and initiatives gathering citizens' opinions reach only limited levels of participation. Following these two findings, Artificial Intelligence could be used to take over some of the tasks citizens are overloaded with.

This is a broad topic that raises issues in many different areas. From an engineering perspective, we focus our study on the discussion whether or not privacy-preserving constructions of such avatars are possible. More technically, we formulate the problem as a Collaborative Filtering recommendation system and solve it with Matrix Factorisation. Finally, we use Homomorphic Encryption and propose two privacy-preserving protocols to address the cold-start problems.

Acknowledgements

First of all, I would like to thank my supervisors Pr. Olivier Pereira and Dr. Pierrick Méaux for giving me the opportunity to study a subject I am passionate about. I also wish to thank them for all the long discussions and exchanges that gave a shape to this work. The setbacks were many, but necessary. My thanks to Pr. Pereira for all his wise and pertinent remarks. Special thanks to Dr. Méaux for his help, advice and patience in explaining the theoretical foundations of cryptography to me.

I thank Politools Asbl for supplying me with a dataset. My thanks also go to Pr. Vincent Jacquet, Pr. Min Reuchamps and Mr. Julian Clarenne for their opinions on the social and political implications of such a system.

Happiness is only real when shared. Many thanks to my friends for all the inspiring discussions and the many wow moments. And this is just the beginning.

A special thanks to my scout staff for the lasting relationships and all the crazy stuff they took me in. In the same way, I would like to thank all my housemates of ULYC Asbl and Kap Expe for the 4 unforgettable years spent in Louvain-la-Neuve.

I thank my parents and brothers for the evenings of laughter and unwinding. Special attention to Zazu, and everyone knows why. A winning team.

Finally, among many others, I would like to thank Stack Overflow and Google who helped me throughout my studies and without whom, this work would still be stuck in chapter 2.

Contents

Acronyms

AD Augmented Democracy. 1

AI Artificial Intelligence. 1

CF Collaborative Filtering. 15

DP Differential Privacy. 31

FL Federated Learning. 14

GD Gradient Descent. 6

HE Homomorphic Encryption. 2

HEAAN Homomorphic Encryption for Arithmetic of Approximate Numbers. 34

MF Matrix Factorization. 13

MLE Machine Learning Engine. 32

RLWE Ring Learning With Error. 34

SMC Secure Multiparty Computation. 31

VAA Voting Assistant Application. 8

Introduction

Even though the number of democracies in the world has increased a lot over the last decades, during the last few years, a deterioration of the democratic principles has been observed in the light of the citizens' expectations. This has been called the *democratic backsliding* [1]. The Pew research centre [2] and Dalia research centre [3] reported the main feelings people have about our actual democracies. An interesting result is that 78% of the respondents around the world believe that democracy is essential and 40% of persons living in a democracy feel that their country is not actually democratic. Moreover, 64% of the surveyed think that representatives do not care about what people like them think. In representative democracies (i.e. most EU democracies), this is understandable since an elected politician needs to represents thousands, or even millions, of individuals. In a way, politicians represent a unique average individual (centroid) of their electorate (cluster).

The idea of Augmented Democracy (AD) breaks this one-over-a-million ratio and aims to have one representative for each citizen. This 1:1 ratio is only possible if we accept that the representative is not a human, but an Artificial Intelligence (AI) powered avatar. We would then have a mix between a direct and liquid democracy. In direct democracies, all citizens are invited to participate in the political decisions. The best, although not completely democratic[1], example was Athenian Democracy in 500 BC. On the contrary, in liquid democracies, a citizen delegates his or her democratic ability to someone else; in this case, an avatar.

Such systems raises many legal, ethical and privacy interrelated issues. These are multidisciplinary problems and experts of several domains should work together. Last but not least, one should ask oneself whether the transfer of voting ability to an AI-powered avatar is desirable or not. While the answer to this question is left to the reader, we will discuss here the privacy issues of such AD systems.

Data science and machine learning techniques rely on data. Privacy-preserving training on confidential data is a difficult task that is intensively studied for its various domains of applications (e.g. medical, financial sector). While online-voting schemes go to the great lengths to protect the vote confidentiality of their users, it would be nonsense to leak this data during the training of the model [4]. Therefore, among the several existing techniques, we will set aside some techniques that insufficiently guarantee the confidentiality of user's data and instead bring out Homomorphic Encryption, the heavy artillery.

[1] since only free, male, adult Athenians could participate. However this should be put in the context of the time.

Homomorphic Encryption (HE) is a cryptographic primitive that allows computations through the encryption envelope (i.e. computations on encrypted data). In classical client-server architectures, data is securely stored on both user and server side. During communication exchanges over an untrusted network, data is encrypted through various schemes. But when the server has to *do the job* (i.e. the service the client subscribed to), it has to decrypt clients' private data and process it in clear. This is where the shoe pinches. With HE, data remains encrypted end-to-end and the server can continue *doing the job* while the data remains encrypted.

This brings us to the following research question : **Can homomorphic encryption enable privacy-friendly augmented democracy?** While HE ensures a good security standard, this comes at a price. HE schemes are computationally heavy and current supported homomorphic operations are still -by several orders of magnitude- slower than plaintext operations. Therefore, we will consider the benefits and drawbacks of HE and put these in perspective with other solutions.

To be able to study the privacy issues of AD, we first had to give it a shape. While some primarily *legal* discussions regarding the introduction of a, augmented direct democracy have been published; to the best of our knowledge, no *technical* proposals have been made. The question of how to model such an avatar led us to consider numerous different problem modelling approaches before reaching the final one discussed here. Each modelling approach has different privacy requirements. As the two problems are intrinsically linked, a change in the first one, could bring a fundamental difference in the second one. This significantly slowed down the process.

The aim of this master thesis is to prove that it is technically possible to achieve privacy friendly AD and that it should not become a stillborn idea. I hope this work will inspire other people to study this field and eventually see some practical applications of it in small communities, schools, universities or even cooperative banks [5]. We make no claim that the solutions we propose here are the most appropriate; on the contrary we would like to put the technical part of the topic on the table and initiate the debate.

This work covers two different technical fields: Machine Learning and Cryptography. To make this work accessible to everyone, we have grouped the technical aspects of machine learning and cryptography in Chapters 3 and 4 respectively. And so, Chapter 2 discusses the AD problem and presents the general ideas of our proposed solution with a reduced technical overhead.

Chapter 1

Background

1.1 Augmented Democracy

1.1.1 Original Proposition

Professor Cesar Hidalgo presented in his Ted Talk "*A bold idea to replace politicians*" [6] the concept of "*augmented democracy*". The main idea behind augmented democracy is the use of digital twins to enlarge the direct participation of citizens in democratic decisions. These AI-powered digital twins would be able to predict citizens' vote on a certain bill. To do so, it would train itself by having access to the citizen's active data (questionnaires, surveys and citizens feedback) and passive data (reading habits, social media behaviour, etc). With this democratic construction, there would be a 1/1 ratio between the number of citizens and the number of politicians. Each citizen would have a twin that is representing his ideas and no other.

AD allows for the three prisms of *e-Participation* [7]. More specifically, e-Participation aims at augmenting the participation and involve the citizens in the democratic processes. Three levels of participation can be defined.

1. *Information provision* which is the one-way communication from the representatives to the citizens,
2. *Citizen consultation* that allows a bidirectional communication between the government and the people that can express their opinion
3. *Citizen active participation* requires active citizens involved in the decision-making and the shaping of policies.

Cognitive Bandwith Prof. Hidalgo highlights the actual cognitive bandwidth problem with actual direct democracies. Most of them achieve the first level of participation by publishing large numbers of open data (e.g. Parliaments, etc). As a result, citizens receive more data than they are able to process. Moreover, people don't have the time and motivation to read all these reports and to be an expert in every subject. Therefore a digital twin could be a solution for direct democracy and play the role of a personal expert taking over some part of citizens duty.

Autopilot He also introduced the idea of an autopilot mode. Depending on his interest and knowledge on the topic, a citizen could choose from different autopilot modes. For example, these modes could cover the three participation levels presented above. A citizen defining a high autopilot mode, would delegate parts of his voting ability over to his avatar. The opposite, an active citizen could define a low autopilot mode and his avatar would then act as a personal expert and nothing more. Of course, different levels in between could be considered.

The disappearance of corruption and the loss of utility in political fuss are other augmented democracy arguments put forward in the provocative Ted Talk. However, a lot of new questions raises with this new concept. Who will control the data? Who will develop these sensitive AIs? What about the biases? Privacy? How could we add guardrails?[1]

1.1.2 Existing Initiatives

We will see that while some initiatives already use AI techniques for democracy, others could greatly benefit from it to (a) increase participation in referendums or public consultations, (b) improve communication, feedback and transparency or even (c) summary and simplify bills to make them readable for all (e.g. GPT-3 with promising text summaries results [8]).

Pol.is

Pol.is [9] is an open source technology developed by the non-profit *Math & Democracy* that wants to bring data science to democratic participation. Organisations can create polls and broadcast invitation links across their participants. These can agree or disagree with the existing propositions and add new ones to which their fellow citizens will give their opinions and so on. At the end of the process, Pol.is uses data science techniques to cluster the participants in different groups and find consensus among the proposed statements. Pol.is has been used across the world with projects like *vTaiwan* [2] or *Engage Britain* for diverging purposes.

Switzerland

Switzerland direct democracy tradition dates back to 1848. For every new bill published by the federal (i.e. national) government, citizens have 100 days to collect 50 000 signatures. If that number is reached, a referendum (called a votation) is held and the law comes into force only if the referendum outcome is favourable. These referendums are mostly optional but becomes mandatory when it comes to the constitution. On a smaller scale, some cantons or even cities have their own referendum processes.

However the whole population is not equally represented during these votations and the participation level of some parts of the population is quite low (e.g. young or low-income people). [10]

[1] He proposed some answers to these questions on his website : `https://www.peopledemocracy.com/`

[2] `https://www.nytimes.com/2019/10/15/opinion/taiwan-digital-democracy.html`

Democratic Participation for Smart Cities

In response to a steady decrease in public confidence in the government due to corruption and other scandals, Madrid decided in 2015 to launch a pioneering initiative in the field of e-Democracy. Decide Madrid has been built to engage citizens in the decision-making process. Five years later, the results are positive but there are still several ways to improve it, especially regarding the participation level. Transparency and communication seemed to be the most important problems of Decide Madrid. The lack of feedback and information about the result of their contribution demotivated more than one citizen to take part actively in the project. [11]

1.2 Preliminaries

The aim of this preliminary section is to give a broad overview of the used techniques. The technical aspects will be discussed in more depth in Chapters 3 and 4.

1.2.1 Homomorphic Encryption

We develop here the main ideas of homomorphic encryption. The used notations will also be introduced. For a more in-depth discussion, we refer the reader to Chapter 4.

Let $\mathcal{P}$ and $\mathcal{C}$ be respectively the plaintext and ciphertext space (i.e. the unencrypted and encrypted space). A generic public probabilistic encryption scheme has the following three algorithms:

1. **KeyGen** The key generation method takes as argument the desired security level λ and generates a public key pk and a secret decryption key sk.

2. **Encryption** With public key pk one can encrypt a message $m \in \mathcal{P}$ into a ciphertext $[[m]] \in \mathcal{C}$ using the $\texttt{Enc}(m, pk) = [[m]]_{pk}$ algorithm. As we are using only one public key throughout this work, we will omit the second parameter and use $\texttt{Enc}(m) = [[m]]$.

3. **Decryption** The secret key sk is necessary to decrypt the ciphertext $[[m]] \in \mathcal{C}$ and obtain the original message: $\texttt{Dec}([[m]], sk) = m \in \mathcal{P}$. Again, as we are only using one (pk, sk) key pair, we will omit the second parameter.

This encryption scheme becomes a *Partially Homomorphic Encryption* scheme if it satisfies the following property, indefinitely:

4. **Homomorphic Operation (1/2)** An operation $\star$ is homomorphic in the ciphertext space if there exist another plaintext operation $\square$ such that:

$$\texttt{Dec}\Big([[m_1]]_{pk} \star [[m_2]]_{pk}, sk\Big) = \texttt{Dec}\Big([[m_1 \square m_2]]_{pk}, sk\Big)$$

   For example, the RSA [12] and ElGamal [13] schemes support the multiplicative homomorphism `Mult`, while Paillier [14] supports the additive homomorphism `Add`.

A *Somewhat Homomorphic Encryption* (SWHE) scheme supports the methods described above together with a second homomorphism.

5. **Homomorphic Operation (2/2)** A second homomorphic operation is supported. However, only a predetermined number of those operations is supported before the ciphertext collapses under its weight[3] and becomes indecipherable.

If we allow both homomorphic operations for an unlimited number of times, we obtain a *Fully Homomorphic Encryption* (FHE) scheme. Most FHE schemes achieve first a leveled-FHE scheme (similar to the SWHE presented above) and adds a ciphertext refreshing method once the maximum number of allowed operations is reached.

6. **Refresh** The `Refresh` method can be seen as encrypting a second time the ciphertext and -thanks to the supported operations- decrypt homomorphically the inner ciphertext while keeping the outer one intact. Once the predetermined number of operations is reached, we can thus replace the consumed inner ciphertext with a new freshly one.

Computation Cost To give an idea of the computational overhead of homomorphic operations, we report in Table 1.1 the latencies of homomorphic operations (`HOP`) on a high-performance commodity server. It includes ciphertext-ciphertext multiplications (`MultCC`), ciphertext-plaintext multiplications (`MultCP`), ciphertext-ciphertext additions and table lookups (e.g. for evaluating non-linear functions). The notations `MultCC` and `MultCP` will frequently be used. Ciphertext-plaintext additions come almost for free. Not reported here, one must also take into account latencies of `Keygen, Encryption` and `Decryption`.

Operation	BFV (s)	BGV (s)	TFHE (s)
`MultCC`	0.043	0.012	2.121
`MultCP`	0.006	0.001	0.092
`AddCC`	0.0001	0.002	0.312
`Table Lookup`	/	307.9	3.328

Table 1.1: Computational *amortised* cost of 3 different HE schemes. Taken from [15]. These results vary *significantly* from one implementation to another and should be taken with a grain of salt.

1.2.2 Gradient Descent

Gradient Descent (GD) is a first order iterative optimisation algorithm that aims at minimising an objective function $\mathcal{L}$ over a set of parameters θ. For each epoch t, the gradient $\nabla_\theta^{(t)}\mathcal{L}(\theta)$ is computed and the parameter set is updated.

$$\theta^{(t+1)} = \theta^{(t)} + \eta \, \nabla_\theta^{(t)}\mathcal{L}(\theta^{(t)}), \tag{1.1}$$

where η is the learning rate. Slight variations exist and they differ by adding one or more terms in the update rule (e.g. momentum), but are based on the same principles. If the

[3]Why and how is explained in Chapter 4

objective function is convex and differentiable, the gradient descent method converges to a local minimum. Several other iterative optimisation methods exist [16].

1.2.3 Performance Metrics

We will use the following metrics to evaluate our different models:

MAE The mean absolute error (MAE) gives rapidly an intuitive idea of the *average* (absolute) distance between the prediction $\hat{y}$ and the true value y: $\text{MAE} = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i|$

RMSE We will use the root mean squared error (RMSE) instead of the mean squared error (MSE) to penalise more large errors as they are particularly undesirable in our system: $\text{RMSE} = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}$

R2 R-squared (R2) gives an idea of the variance explained by the model over the total variance. In other words, R2 is the fraction of the total sum of squares explained by the model and is defined as follows:

$$\text{R2} = \frac{n \times MSE}{\sum_{i=0}^{n}(y_i - \bar{y}_i)^2}$$

We will use the r-squared score for the parameter tuning steps since it is more nuanced and intuitive (the higher, the better). RMSE and MAE will be used for the final evaluations and comparison of our protocols.

1.3 Related Work

We discuss here the related work regarding e-Democracy and the prediction of voter outcomes. Related work of the technical constructions we used, is discussed in Section 5.2.2.

1.3.1 Election Result Forecast

Driven by political interest and their resulting economic issues, many machine learning studies tackle the forecast of election results [17]. They use (one or) a combination of historical data, different types of polls, expert surveys, statistical models and sentiment analysis. Unfortunately, this active and appealing research domain varies fundamentally from the less active one discussed here.

Predicting a vote outcome over a bunch of citizens (e.g. a region or a country) is far away from predicting individual citizens votes. First of all, the available data sources for both problems are diametrically opposed by their quantity, their size, their features and their quality (see Section 3.1). This problem has already been highlighted when predicting vote outcome on regional level [18]. As stated in [19], election forecasting models uses measures of aggregated explanatory variables. Disaggregate these variables over a region is often impossible. Last but not least, during individual predictions, prediction errors sum up instead of cancelling each other out.

Sentiment Analysis Sentiment analysis (aka opinion mining) uses Natural Language Processing to identify user opinions and sentiment occurrences in texts. When sentiment analysis meets social network data, various economical and socio-political applications emerge. The authors of [20] were the first to argue that Twitter data and sentiment analysis could replace traditional polls. Since then, researchers worked on various sentiment analysis models to use Twitter data for *a posteriori* election outcome prediction [21]. On the other hand, the authors of [22] swim against the AI-tide and, motivated by studies contradicting each other, they provide a balanced view of the real predictive possibilities of sentiment analysis. Their conclusion is negative and suggests that the reported results are biased by enthusiast researchers that only report positive results (aka the "file-drawer effect"). They also warn against the bias of Twitter data as it represents only a fraction of the population that decides to (publicly) tweet a lot. Even if sentiment analysis can be used in recommendation systems to address the cold-start problem [23], we decided not to use it for the above-mentioned reasons.

Online Prediction of Vote Results [18] is probably the work that comes closest to our case study. On referendum days, this Swiss laboratory provides predictions on the outcome of the national vote based on the first *regional* results published at noon. They used interesting techniques and we will often refer to their work.

Voting Assistants Nowadays almost every country has its own Voting Assistant Application (VAA). On these platforms, politicians and citizens expresses their preferences on political aspects and based on these answers, candidates and users are matched. Similarity scores [24] or fuzzy clustering [25] are used.

Socio-demographic indexes For widely debated political issues (e.g. legalisation of cannabis, abortion, etc), sociological studies identifies indexes and determinants that will impact individuals opinion [26]. However there don't exist any *golden rule* that defines indexes and determinants for each publicly debated issue[4].

1.3.2 Data Science for Democracy

Still in Switzerland, authors of [27] made a data-driven study of the political landscape of the country. They discovered interesting patterns thanks to data science techniques, without the need of time consuming manual analysis. They crossed multiple datasets and empirically confirmed many ideological findings.

More interestingly, in a VAA setup, they crafted a *perfect* candidate profile with as aim to maximise the number of voting recommendations he gets. By identifying gaps in the PCA projections of the candidates and filling it with tailored responses, a perfect candidate could gather twice as many recommendations as the best real candidate. As good white hats, they also provide a mechanism to detect and illustrate opinion shifts between the VAA responses of a candidate and his daily votes in the Parliament.

[4]What came out of *my* interpretation of a inteview with Vincent Jacquet, a socio-political scientist.

Chapter 2

Problem Statement and Methodology

This section approaches the problem from a non-technical point of view. We will thus often refer to further discussions. Before being able to study any privacy issue, we had to give a shape to AD. As this is a vast subject, we were forced to make simplifications and hypotheses. Among the numerous technical challenges of AD, we identified the training of these avatars as a matter of privacy issues.

We report here some ins and outs related to the specifications and modelling of the problem. We formally define a collaborative filtering problem and give a quick overview of the proposed solution. Finally, we define criteria that will permit to assess and compare our proposed protocols.

Reluctantly, many simplifications had to be made to fit this vast subject into a master thesis. Concessions had to be made on the prediction part to maximise expressiveness and simplicity before a translation in the ciphertext space.

2.1 Problem Simplifications

Predicting citizens' preferences on a bill is a complex prediction task. We will make some simplifications that will allow us to formulate the problem in a more general form and establish links with existing problems.

From Bill to Item Because a bill contains a bunch of different nuanced subparts, we simplified the idea of voting directly on a bill towards emitting a preference on a specific question or subpart of it. This brings us to a kind of organisation comparable to the direct democracy in Switzerland where citizens vote several times a year on numerous different subjects. We won't deal with bills or acts anymore but with questions, votings or items.

From Citizen to *User* and Parliament to *System* In the original construction, citizens participate in an augmented democracy process which is administered by a parliament or government. Throughout this work, we will step away from this formulation

and rather consider USERs using a service hosted by a SYSTEM. For ease of writing, we will alternately use USER or *he* and SYSTEM or *it*.

Active and Passive Citizens We consider *active* and *passive* citizens who have to express some preferences on different issues. Active citizens are interested in this democratic process and want to give their opinion. Passive citizens are not interested in this process for personal reasons and set their avatar in a full-autopilot mode. These reasons could include a lack of time, knowledge or interest in the discussed issues.

No socio-demographic data As obtaining the best possible performances was not the goal here, we restricted our datasets and used only past emitted preferences to infer new ones. This simplified our models within sight to translate them in the cryptographic domain and train them in a privacy-preserving manner. We thus do not include any socio-demographic data (e.g. location, gender, revenue, education, etc.), although it has been proven that vote results are significantly correlated with spatiality [18], age [28] or socio-economic characteristics [29]. However, our system should easily generalise to the addition of these new data sources.

Sparsity of the responses As users could decide not to issue preferences for a while, delete their avatar, or rejoin the system later on (e.g. once the legal voting age is met), a sparse preference matrix is obtained.

Iterative methods For this study, we decided to stick to an iterative approach regarding the training of the regression model. Among linear programming approaches, other possibilities exists to train our proposed model. This choice is further discussed in section 3.3.1.

Hypotheses To model the problem we had to make strong hypotheses on sociological field that would simplify the prediction task. These hypotheses include:

1. **No temporal dynamics** The opinion of a USER did not evolve with time. If he voted `yes` on the question j at time t, he will continue to vote `yes` on the same question at time $t + 1$.

2. **Feature set** The preference of a USER on a new question can be predicted based on his previous responses. The previous questions are thus among the explanatory features of the prediction task.

3. **Low Dimensional Causes** It is possible to predict a preference using combinations of low-dimensional factors (causes) representing the question.

4. **No data poisoning** Users do not produce fuzzy data that would corrupt or decrease the performance of the model. See Section 4.4 for an in-depth analysis of the threat model.

2.2 Problem Definition

The simplified avatar discussed here assists a *passive* citizen and predicts its preference for him. For every new voting, the predicted votes of the *passive* citizens would be cast together with the votes of the *active* citizens. Each avatar would be able to predict the preference of its related citizen based on its personal profile and a representation (definition) of the voting issue.

Let $\mathcal{U}$ represent a set of USER(resp. citizen) profiles, $\mathcal{V}$ a set of ITEMs (resp. votes, questions) and $\mathcal{R}$ a set of preferences (resp. votes). We look for a model $\mathcal{M}$ that produces $\mathcal{R}$ in a privacy-friendly manner.

To remain as broad as possible, we define 4 problems. We believe these 4 problems form the core of an AD avatar.

1. **Initialisation**. Setup and initialisation of the model $\mathcal{M}$.
2. **New user problem**. A new USER i joins the system and seeks to get its personal profile $\mathcal{U}_i$.
3. **New question problem**. A new ITEM j is published and must obtain an encoding $\mathcal{V}_j$ that defines it.
4. **Prediction**. Using its USER-profile $\mathcal{U}_i$ and a ITEM encoding $\mathcal{V}_j$, the avatar issues a preference on the given ITEM.

In addition, based on the hypotheses and assumptions of previous sections, we look for a solution that fulfils the following high-level properties:

Property 1 (Transparency)**.** $\mathcal{M}$ *should adopt a simple approach to maximise transparency and interpretability of the issued preferences.*[1]

Property 2 (Generalisation)**.** $\mathcal{M}$ *should show a good generalisation ability to the addition of metadata (e.g. socio-demographic informations, reading habits, domain of interests, etc.)*

Property 3 (Sparsity)**.** $\mathcal{M}$ *should be able to deal with missing values and offline* USERs*.*

Property 4 (Complexity)**.** $\mathcal{M}$ *should use algorithms with an efficient translation in the homomorphic domain.*

Property 5 (AD)**.** $\mathcal{M}$ *should follow as best the core ideas of AD.*

2.3 Approaches

The above-mentioned requirements brought us to consider different approaches to model the problem.

[1] As a pledge of transparency, each USER should be able to interpret and understand his avatar predictions. So it comes back to the avatars to be able to explain the reasons and determinants used to make a choice. This can take several forms. This seems to be a given, but many actual machine learning algorithms are difficult to interpret (even by their designers).

2.3.1 Latent Space

Before arriving to the latent space approach, we considered two -initially opposite- approaches.

User-Based Approach

In this approach, the avatar is central and we are closest to the original idea of AD: having a *personal* avatar. Each USER has its own model and a prediction is obtained by feeding the model with a quantified profile (i.e. an encoding, a representation) of the question. We have thus as many models as we have USERS

$$\mathcal{M}_i : \mathcal{V} \to \mathcal{R} \quad \forall i \in \mathcal{U} \tag{2.1}$$

Item-Based Approach

Here, the question is central and we move away from the original idea of AD as all USERS share the same model. A user would obtain his preference by feeding his profile to the model. Each question has its model and for each new question, a new model must be retrained. The problem of finding a the question representation is here hidden into the model.

$$\mathcal{M}_j : \mathcal{U} \to \mathcal{R} \quad \forall j \in \mathcal{V} \tag{2.2}$$

Fair Encodings

The main issue of the user-based approach is to find the representations of the different questions. An intuitive representation would be obtained by dividing a question in a set of objective criteria (i.e causes, scores). However, the introduction of these representations raises several questions: Who defines them? Politicians? Experts? Which experts[2]? How can we obtain fair and objective representations without falling in a technocracy where experts are choosing for us? Rapidly we realise that this is the weak link of the chain. Interfering in the decision of these representations could be a form of corruption. Therefore these should be as neutral and unbiased as possible[3]. Basing our thoughts on *our* interpretation of the opinions of some political scientists we interviewed[4], we argue that the fairest way to proceed is to select a random subset of citizens and let them define together this sensitive question encoding.

In other words, in the first approach (2.1), each individual builds his model and the question encodings -defined by a random subset of the collective- serve as inputs of the model; while in the second approach (2.2), the collective builds the model and each individual feeds it.

[2]The covid pandemic was a good example of contradiction between experts and the challenges of choosing the expert(s) to listen to.

[3]One can argue that it is impossible to have unbiased models. As this is an important question, we make the hypothesis that by selecting a random group of USERS, bias can be minimised to a negligible value.

[4]Following discussions with Pr. Vincent Jacquet, Pr. Min Reuchamps and Mr. Julian Clarenne.

Latent Space Approach

Finally, we adopted a latent space approach which provides a compressed representation of the data by embedding it into an underlying vector space [30]. We define the latent space $\mathcal{S}$ and a projection $\pi_{\mathcal{S}}$ that project the users and items on it. This allows to represent both users and items with a latent vector in $\mathcal{S}$. Our model would then use these latent factors to produce the preference $\mathcal{R}$.

$$\mathcal{M} : \{\pi_{\mathcal{S}}(\mathcal{U}), \pi_{\mathcal{S}}(\mathcal{V})\} \rightarrow \mathcal{R} \tag{2.3}$$

We note that $\mathcal{U}$ and $\mathcal{V}$ are not constant over time (e.g. USERs join the system and vote on new ITEMs). Projecting these spaces on a fixed latent space $\mathcal{S}$ enables us to have a unique formulation, constant over time.

AD Property While the Item-Based approach causes us to move away from the original idea of having a 1:1 ratio between USERs and avatars (i.e. models), the latent space approach takes the best of both worlds.

Transparency Property Intuitively the d components of the latent vector represent the relative importance a USER places in the d possible aspects (causes) of the latent space. We reuse the same intuition as for classical movie recommendation systems where the latent factors represent movie genres or actors. Here these could be environmental, social or economic impacts of a vote. With this in mind, we could easily read and interpret user and item profiles (i.e. what a user thinks and what a item is about).

2.3.2 Matrix Factorisation

We noticed the close similarity of the problem with recommendation systems [31]. Among the many existing techniques, we will use Matrix Factorization (MF) for its simplicity and proven efficiency in numerous recommendation problems [32]. As shown in Fig. 2.1, MF enables us to factorise a preference matrix R in two separate matrices U and V such that $R \approx U \times V$. The user matrix U contains the user profiles as rows. Each row vector represents a USER and defines his avatar. The columns of the item matrix V contains the representations of the questions. The factorisation defines thus the projection $\pi_{\mathcal{S}}$ and solves the fair-encoding problem by letting the USERs themselves define the encoding (i.e. latent factor) of the question through their preferences and latent factors. As discussed in Section 3.2.1, other dimensionality reduction techniques exist for defining the projection $\pi_{\mathcal{S}}$.

Unfortunately, matrix factorisation techniques struggle to make recommendations on new ITEMs and new USERs. Every new citizen joining the system and every new question will be considered as a cold-start problem as the model is unable to gather sufficient information on the newcomers. While most cold-start problems are solved by either making first initial random guesses or through hybrid approaches combining multiple models and external features, this is not possible here. Instead, random USERs will be selected to answer and rate the new ITEM.

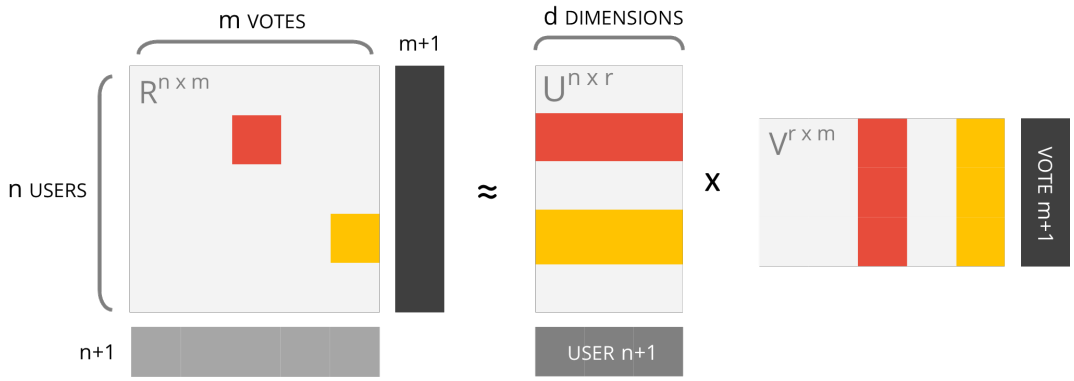


Figure 2.1: Factorisation of matrix R in a USER matrix U and ITEM matrix V. Adding a new USER to the system is represented in light gray and the addition of a new question in dark gray. Two predictions are highlighted in yellow and red.

Generalisation Property MF generalises well to the addition of metadata. The data could either be included directly into the factorisation or later on, as bias.[18] used such approaches to add region features and predict regional vote outcomes.

Sparsity Property MF allows the preference matrix R to contain missing values. These missing values are *absorbed* during factorisation and both U and V matrices do not contain any missing value.

Complexity Property Predicting a preference (multiplying U and V) is of limited complexity and uses HE-friendly operations.

2.3.3 Federated Learning

The factorisation of the preference matrix in two separate matrices makes it possible to apply Federated Learning (FL) principles. The core idea of FL is that each user pre-processes locally his private data. He then shares with the central server only the information necessary to train the model. So instead of storing the USER matrix U in a centralised manner, each USER can store locally his own part of the matrix (i.e. his own profile, his own avatar). As the item matrix V is publicly available, each USER can locally train his avatar and predict a new preference without having to share any information. This is schematised in Fig. 2.2a.

As USER data never leaves his device, we have a form of *privacy by design*. However to obtain the initial factorisation and to update matrix V with new items, USERS must share some informations (e.g. gradients in the case of FL). This brings confidentiality issues and make privacy attacks possible.

Symbol	Signification
$\mathcal{U} \equiv \{\mathcal{U}_0, \ldots, \mathcal{U}_{n-1}\}$	set of n USER profiles
$\mathcal{V} \equiv \{\mathcal{V}_0, \ldots, \mathcal{V}_{m-1}\}$	set of m ITEM encodings
$\mathcal{S}$	latent space (defined during the matrix factorization)
d	dimension of the latent space $\mathcal{S}$
$u_i \equiv (u_{i0}, \ldots, u_{id})$	USER i latent (row) vector
$v_j \equiv (v_{0j}, \ldots, v_{dj})$	ITEM j latent (column) vector
$R \equiv \{r_{ij} \vert \forall i \in \mathcal{U}, j \in \mathcal{V}\}$	vote matrix with the n users as rows and m items as columns. r_{ij} denotes the elements of this $n \times m$ matrix.
$U \equiv [u_0, \ldots, u_{n-1}]^T$	user matrix of size $n \times d$ with with the n latent factors u_i as rows.
$V \equiv [v_j, \ldots, v_{m-1}]$	question (or vote) matrix of size $d \times m$ with the m latent factors v_j as columns.
$S^k \subset \mathcal{U}$	set of k selected USERs for training a new ITEM

Table 2.1: Notations relative to the collaborative filtering problem. Other, more specific, notations are introduced in the next chapters.

2.4 Proposed Solutions

We formulated the problem as a Collaborative Filtering (CF) problem with as aim to automatically predict the preference r_{ij} of a USER i for a given ITEM j. More specifically, we will use a MF-based recommendation system with the notations defined in Table 2.1. Both the addition of new users and new questions will occur frequently and these two cold-start problems will form the angular stones of our model.

However, this is not a classical case of a collaborative filtering problem. This is mainly due to the unbalanced dimensions of the problem (i.e. the USER/ITEM ratio) and the high importance of the cold-start problems (as this is the main prediction we are looking for). Furthermore due to the nature of the problem, there is little scope for error.

We reformulate the 4 problems defined in section 2.2 to our distributed matrix factorisation approach and propose solutions for each of them. As the aim of this chapter is to give a broad overview of the problem and the proposed solution, we refer to some more detailed sections after each proposed solution.

Problem 1 (Initialisation)**.** *Given preference matrix $R^{n \times m}$, find profile matrices $U^{n \times r}$ and $V^{r \times m}$ which product approximates R in the d-dimensional latent space $\mathcal{S}$.*

Proposed Solution We define a classical Matrix Factorisation objective function that must be minimised over U and V. Gradient descent iterative methods are used to solve this minimisation problem. [Section 3.2.2].
Privacy concerns: For confidentiality reasons, each row of the preference matrix R is

encrypted before being outsourced to the SYSTEM. [Section 4.2]

Problem 2 (Prediction of a Preference)**.** *Given USER latent vector u_i and ITEM latent vector v_j, predict the preference r_{ij} of USER i to ITEM j.*

Proposed Solution In the latent space, a USER i can solve this problem by simply taking the dot product of his personal latent vectors u_i and the ITEM latent vector v_j. This is shown in Fig 2.2a. [Section 3.2.2]
Privacy concerns: As v_j is publicly available, the USER can make the prediction locally and never has to share or outsource his latent vector during the process. [Section 4.2]

Problem 3 (New User Cold Start)**.** *Given question latent vectors v_j and preferences r_{ij} for $j \in S'$, find the d-dimensional latent vector u_n representing USER n at best in the latent space $\mathcal{S}$.*

Proposed Solution The newcomer n answers a set S' of initial questions and infer his personal profile based on the encodings v_j of the answered questions. More precisely, he solves a linear regression on the answered ITEMs of S'. [Section 3.2.3]
Privacy concerns: Again, as the v_j representations are publicly available, the regression can be done on USER side and no information is leaked. Classical analytic methods as the least squares can be used to solve this linear programming problem. [Section 4.2]

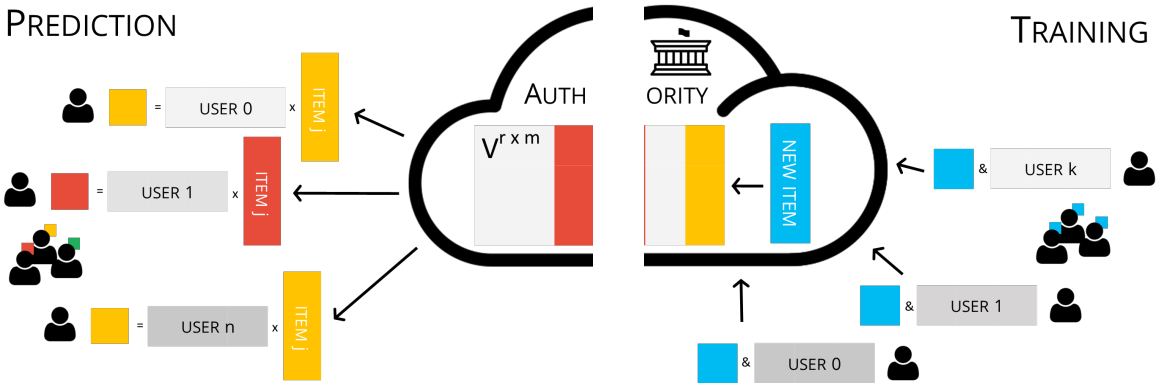
Problem 4 (New Item Cold Start)**.** *Given USER latent vectors u_i and preferences r_{ij} for $i \in S^k$, find the d-dimensional latent vector v_m representing question m at best in the latent space $\mathcal{S}$.*

Proposed Solution As previously discussed, this is the weak link of the process as a biased representation could have dramatic consequences. We estimated that using the collective and selecting randomly k USERS belongs to the fairest solutions to define a ITEM encoding. Similarly as above, a linear regression is fitted on the latent representations u_i of the k selected USERS. This sensitive process is further described in the next section and schematised in Fig. 2.2b. Technical considerations can be found in [Section 3.2.3].
Privacy concerns: Thanks to cryptographic techniques described in Section 4.3, a centralised server aggregates the local sensitive informations of k selected USERS to make a initial representation v_{m+1}^0 converge towards a final representation v_{m+1}^t. [Section 4.6]

Problem 1 defines the initial matrix factorisation used to define U, V and -at the same time- the latent space $\mathcal{S}$. Problem 2 must be solved in a privacy-preserving way each time a USER wants to get a preference r_{ij}. When solving problem 3, a new user seeks to get its own latent representation. In problem 4, a latent representation is searched for each new ITEM (i.e. question, bill, referendum) added to the system.

2.5 Studied Solution

The solutions to problems 2 and 3 are relatively straightforward and have no direct confidentiality leaks thanks to the distributed approach of FL. We will thus not elaborate these



(a) n USERs making a prediction (b) k USERs training a new ITEM profile

Figure 2.2: Federated approach to solve the problems 2 (left) and 4 (right). For the prediction problem (a), the authority publishes the latent vectors (yellow and red) and the USERs make their predictions on their own devices. The new ITEM cold-start problem (b) is solved by selecting k users that will share their predictions (blue) and latent vectors (gray) to help the authority infer the latent vectors of the new question.

any further in the following sections. On the other hand, the solutions and design choices of problems 1 and 4 have larger implications. Privacy-preserving MF is a topic extensively studied in literature (see Section 4.2), and we decided to use existing work as the basis for ours. We consider thus the matrices U and V initially factorised and will focus on problem 4.

The introduction of a new question to the SYSTEM is the weak link of the process. The process of obtaining a fair and neutral encoding of a vote is not an easy task. For privacy and fairness reasons, we came up with the following framework:

1. A new question m is published
2. k USERS are randomly chosen from a population and form the set S^k
3. Debates and discussions are organised with the k USERS around the new question
4. The USERS $i \in S^k$ emit their preference r_{im} on the new question
5. SYSTEM infer the representation of the new question v_m based on the k USERS emitted preferences r_{im} and their personal profiles u_i.
6. SYSTEM publishes the representation v_m of the new question
7. Avatars $n \notin k$ predicts their preference for question m
8. Depending on the autopilot mode (i.e. users level of implication in the decision process), each (user, avatar) pair will choose from the following:
   (a) **The user validates manually the predicted preference**
   The validated preference r_{im} and his profile u_i is added to the set of training batches (of the k selected users).

(b) **The user modifies the predicted preference**
The modified preference feedback r_{im} and his profile u_i is added to the set of training batches (of the k selected users).

(c) **The user does nothing or stays offline**
SYSTEM continues updating the representation of question m with the new batches arriving from the situations A and B presented above. Iteratively they publish new, more precise, representations for question m.

9. Shortly before the final deadline, all remaining avatars predict their preference and validate *de facto* the (unsupervised) prediction.

Again, this is a *possible* approach to the problem. Additional constraints can be included in the process. They may include: prevent more than a certain number of unsupervised predictions in a row, replacing unsupervised predictions by missing values, etc.

Contribution Now that we gave a (technical) shape to AD, we will further study how to perform steps 4-7 presented above in a privacy-preserving manner. In Chapter 3, we will study how Solution 4 performs in clear and explore the problem-specific parameters. In Chapter 4, we propose a interactive and non-interactive protocol allowing to efficiently solve the new ITEM cold-start problem on encrypted data.

Chapter 3

Prediction of the Response of a User

This chapter deals with the prediction part of the problem. Before translating our model in the cryptographic domain, we implemented the machine learning algorithms in the plaintext domain. During this stage, we stayed alert to the problems that would arise from a translation to the ciphertext space and especially the number of homomorphic operations (`HOP`).

The results obtained here are not universal and are not meant to be extrapolated to other cases.

3.1 Dataset

For this experiment, a heterodox dataset was required. In a classical problem, one searches for a large number of samples with a restricted feature set that maximises the expressiveness of the target feature. Here, instead, we looked for the dataset with the largest number of features regardless of the number of samples. The objective is twofold: (i) obtain the most precise user representation in the latent space and (ii) validate the obtained model by cross validating the *features* (and not only the samples).

While the number of datasets containing historical aggregated regional or national vote outcomes is abundant, vote results of individual citizens over several different votes do not exist. This is a direct result of the related confidentiality issues (see Section 4.1) and the lack of economic and politic interest in predicting a single vote outcome. Polls and surveys contain approximations of voting results or opinions, but as they are anonymous, it is impossible to gather multiple poll responses of an identical set of USERS over time. The few *online* data sources we found [33] were of low quality and had an insufficient number of ITEMs (less than 20).

We then turned to VAA's as, by their nature and function, they could be a possible source of data suited for this experiment. We obtained a anonymised Smartvote[1] dataset of user data collected during the 2021 Valais cantonal elections in Switzerland [34]. By answering a set of questions, the users of their application are able to obtain recommenda-

[1] A Swiss voting assistant application developed by Politools asbl.

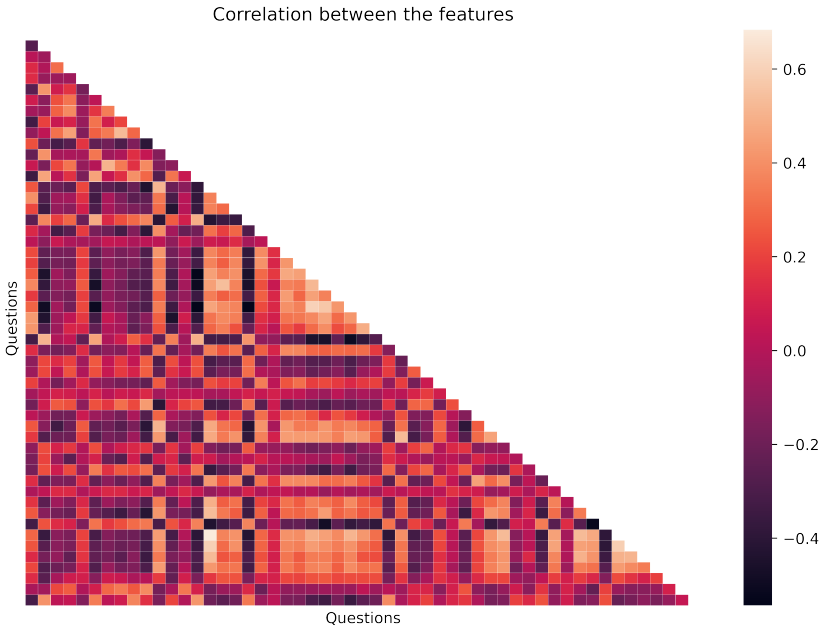


Figure 3.1: Pearson's correlation between the 53 features (i.e. questions, ITEMs) of our dataset.

tions on *candidates* close to their answered question.

Regional and national datasets also differ by the quality and reliability of their data. For example, a dataset containing the vote outcomes on the regional level is the result of a well-defined verifiable process (counting the number of "yes" and "no") and there is no doubt concerning the veracity of the features. The inherent noise originating from the behaviour of each individual is here averaged over several thousands of users. This is not the case in VAA datasets where we can assume this noise is even amplified by the dummy nature of the survey. Which users have correctly answered the survey? Which have not? When averaging vote results over several USERS (e.g. polls or surveys), this issue has a minor role.

Our dataset contains 48.552 samples with each 53 features. These features are user responses (between 0 and 100) to some general questions ranging from the ban of single-use plastic to free-trade agreements. We considered that respondents who have not answered all to all questions, did not complete the survey seriously. We removed thus 26.086 entries. To get a general idea of the dataset, we have reported the correlation between features on Fig. 3.1. We already notice that a series of questions are weakly correlated.

3.2 Matrix Factorisation

Matrix Factorisation (MF) regained interest since the creation of the Netflix Prize Competition [35]. This resurgence of interest stems from its simple design but proven efficiency. Furthermore, MF is very versatile and can easily be adapted to include personality information [36], adapt to temporal dynamics [37] or factorise into deep matrices for getting hierarchical nuances [38] and so forth. Its loss function and the gradient descent algorithm can easily be adapted to these extensions.

3.2.1 Latent Space

As previously explained, the factorisation of the preference matrix, defines inherently the latent space. Projection of user and item to a latent space has various benefits. Firstly, the observed preferences can now be expressed as functions of a number of possible causes. This is a non-negligible transparency and interpretability property. Secondly, this projection enables us to have a fixed low-dimensional space to work with. New ITEMs and new USERs will not unnecessarily expand the space.

While matrix factorisation is mainly used for initialising the problem, the latent space resulting from it will be used to solve all new USER and ITEM cold-start problems. Due to the stochastic character of the iterative methods used to solve the problem, the definition of the latent space is arbitrary. In a real application, additional constraints (e.g. non-negativity MF [39]) should be defined on this space to make it tend towards uniqueness.

Other unsupervised dimensionality reduction techniques could have been used (e.g. PCA [30], LDA [40] or even autoencoders [41]). But we preferred factor analysis for its simplicity and interpretability of the results. Furthermore, it is already successfully used for popular recommendation systems [42].

3.2.2 Factorisation

The matrix factorisation problem in its simplest form seeks to minimise the following objective function:

$$\min_{U,V} \|R - UV\|_F^2, \tag{3.1}$$

where $\|M\|_F^2 = \sum_{i,j} M_{i,j}^2$ is the squared Frobenius norm of matrix M [38]. For generalisation reasons, we add a regularisation term `Reg(.)` and USER and ITEM biases, respectively $b^{\mathcal{U}}$ and $b^{\mathcal{V}}$. A similar approach has been used in [18]. We obtain then our final objective function:

$$\min_{U,V} \mathcal{L}(U,V) = \sum_{i,j} (r_{ij} - \tilde{r}(u_i, v_j))^2 + \sum_i \texttt{Reg}(u_i) + \sum_j \texttt{Reg}(v_j) \tag{3.2}$$

where $\tilde{r}(u,v)$ is the predicted preference and is defined as follows:

$$\tilde{r}(u_i, v_j) = u_i^T v_j + b_i^{\mathcal{U}} + b_j^{\mathcal{V}} \tag{3.3}$$

We solve problem 1 and minimise the objective function (3.2) with first-order gradient-based optimisation methods. Although other widely used methods exist to solve the matrix

factorisation (e.g. the alternating least square method), we decided to stick to the gradient descent approach as we are already using it in the next steps. More details can be found in the next section.

Once the user and item matrices are factorised, a user can solve problem 2 and predict his preference $\tilde{r}_{i,j}$ by taking the dot product between his USER latent vector and the published ITEM latent vector and include the biases $b^{\mathcal{V}}$ and $b^{\mathcal{V}}$ as in (3.3).

3.2.3 Cold-Start Problems

Although we based our model on Matrix Factorisation, we will mainly use it to initialise the system. The new USER and new ITEM cold-start problems are the main problems of the proposed protocols. Both problems seeks to find the missing u_{n+1} (resp. v_{m+1}) latent vector.

Adding a New User

When a new USER i wants to join the system, it has to get its own latent vector representation u_i. The classical least squares problem notation $\min_x \|Ax - b\|_2^2$ adapted to ours becomes

$$\min_{u_i} \|Vu_i - r_i\|_2^2 \tag{3.4}$$

Due to the limited size of V and the centralised nature of the problem (i.e. exclusively on USER side), ordinary least squares methods can be used to solve this problem. As previously mentioned, we won't dig deeper into this problem as it is straightforward to solve and does not have further confidentiality issues.

Adding a New Item

Similarly to the previous cold start problem, a new ITEM m has to obtain its latent representation v_m. The classical least squares problem notation becomes

$$\min_{v_m} \|U_k v_m - r_k\|_2^2, \tag{3.5}$$

where U_k represents the k USER latent vectors and r_k their emitted preferences. Due to the large size of S^k and the distributed nature of the problem, we will use iterative methods to solve this linear regression problem (see next section).

3.3 Iterative Methods

3.3.1 Gradient Descent

Gradient descent optimisation algorithms are often overlooked due to their mediocre results and unreliable character. This is even more the case when solving linear regressions. However, when it comes to large datasets, gradient descent algorithms achieve unexpected good results. This is known as the trade-offs of large-scale learning and is covered in

detail in [43]. On very large datasets with regularity and convexity assumptions, second order stochastic gradients are asymptotically efficient after only a single iteration. While second order iterative methods are computationally costly, they can be approximated with averaged stochastic gradient which is only a step away from our implementation. [44]

Although the toy example presented here uses a dataset of reasonable size, a real implementation should support several hundreds of thousands (millions) of USERs. Other reasons that pushed us to use gradient descent algorithms include (i) their high generalisation capabilities to adapt to other, more complex, objective functions, (ii) their parallelisation ability well suited for the distributed nature of FL; and, finally, the iterative approach that (iii) ease the detection of fuzzy or harmful data and (iv) allows to continue updating the model once new data become available (e.g. a USER gives a prediction feedback).

3.3.2 Convergence

A gradient descent algorithm converges if the loss function is convex and differentiable and if the learning rate is carefully chosen (i.e. η is small). [16]

Classical least squares' problems have quadratic losses (3.4) and are thus convex and differentiable. However, as we are only taking a subpart U_k of the user matrix U, some uncertainty must be taken into account due to the possible variations in the training data matrix U_k: this is known as the *stochastic robust approximation* problem. If we assume U_k to be a random variable with mean $\bar{U}_m$, we can describe it as follows:

$$U_k = \bar{U}_m + \tilde{U} \tag{3.6}$$

Where $\tilde{U}$ is a random matrix with zero mean that describes the statistical variation of U_k from the average matrix $\bar{U}_m$. We can thus use the expected value of $\|U_k x - r\|_2^2$ as the objective value to minimise. With some simplifications we obtain:

$$\mathbb{E}\|U_k x - r\|_2^2 = \mathbb{E}(\bar{U}_m x - r + \tilde{U}x)^T(\bar{U}_m x - r + \tilde{U}x) \tag{3.7}$$

$$= (\bar{U}_m x - r)^T(\bar{U}_m x - r) + \mathbb{E}\ x^T\tilde{U}^T\tilde{U}x \tag{3.8}$$

$$= \|\bar{U}_m x - r\|_2^2 + x^T P x, \tag{3.9}$$

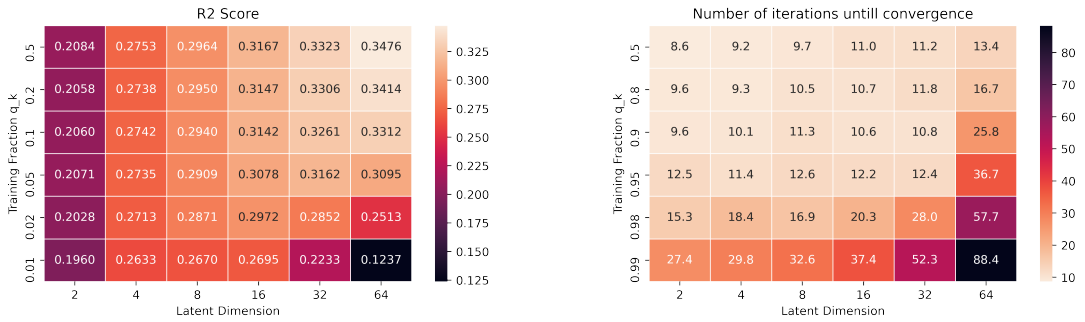
where $P = \mathbb{E}\ \tilde{U}^T\tilde{U}$. We notice here that the second term can be considered as a regularisation term and we obtain the Tikhonov regularised least square problem:

$$\min \|\bar{U}_m x - r\|_2^2 + \|\sqrt{P}x\|_2^2 \tag{3.10}$$

Which is a convex and tractable quadratic optimisation problem. A complete study of this problem and its worst cases can be found in [45].

3.4 Model Parameters

As discussed in Section 2.5, problems 1, 2, 3 are straightforward to solve or do not have privacy issues. We will thus focus the rest of our study to solving problem 4: the new



(a) Regression R-Squared scores (b) Number of iterations until convergence

Figure 3.2: Regression scores and number of iterations until convergence for different latent dimensions r and training fractions q_k. The lighter the colour, the better.

ITEM cold-start problem.

Even if obtaining the best performances was not the main goal of this study, keeping an eye on the model parameters is recommended to understand the learning process and avoid unwanted behaviours. Moreover, these parameter choices have a direct impact on the original problem.

Cross-Validation All our parameter choices have been cross-validated through a solid repeated k-fold with ($k = 10$, $n = 5$) on the whole dataset. However, we used a atypical form of validation since we are cross-validating the target values instead of the training samples. During each fold $^9/_{10}$ of the ITEMs are used to factorise the preference matrix and the remaining $^1/_{10}$ ITEMs are used for evaluating the cold-start problem. We shuffled the revealed splits between each repetition.

Weak Generalisation Due to the non-uniqueness of (stochastic) matrix factorisation, we could only train and evaluate our linear regression models on latent vectors of test-users that have been shaped together with ones of the training-users. This is a case of weak generalisation [46]. A case of strong generalisation would be here, evaluating the new item problem (4) on users that inferred their latent vectors with the new user problem (3). We let this as a future work.

We reported the choices made in decreasing order of importance. All design choices have been made within afterthought the aim to minimise the number of computations and ease the homomorphic evaluation of it.

3.4.1 Testing Fraction q_k

The first decision choice was to define the number of USERS that would form the group of « the k selected » for training the representation of a new item. So we defined the training fraction parameter q_k that represents the fraction of the dataset that will be used for training the model. R2-scores for different values of q_k are shown in Fig. 3.2a.

By taking a step back from this pure data science parameter, we notice that q_k has an important role to play in the implementation of such systems. A q_k of 0.1 means that 10% of the users are among « the k selected » and, on average, a USER will be selected every 10 new questions (i.e. $1/q_k$). So this fraction defines the level of USER implication required. While values above 0.75 questions the usefulness of such a system, values below 0.01 could induce a poor level of USER participation.

3.4.2 Latent Space Dimension d

The dimension d of the latent space has a direct impact on the complexity of the model. The higher d, the more nuance can be observed in the model. In return, a higher d will increase the number of computations as discussed in Section 5.2.1. Unsurprisingly, a higher latent space dimension goes hand in hand with a higher r-squared score. Although above a certain threshold, overfitting is no longer insignificant (see Section 3.5).

We chose a latent space dimension of $r = 16$ as (i) it is a power of 2 and will allow several samples to be easily included in a same ciphertext (see Section 4.6.1); (ii) it is reasonable for the number of available items (i.e. 53 ITEMs) and (iii) experimentally, this leads to sufficient results as shown on Fig. 3.2b. A latent space dimension larger than the amount of available data wouldn't be reasonable.

3.4.3 Learning Rate η

Iterative methods and especially gradient descent algorithms are very sensitive to the learning rate. The η values reported in this work are scaled over the number of samples in the training set. This avoids us to tune a parameter on a moving ladder.

With the aim of reducing the number of iterations (and thus the number of `HOP`), we tried an adaptive learning rate that decreases as the iterations progress. One can then oversize the learning rate with as aim that the GD algorithm converges faster. By defining carefully both batch size and learning rate, reasonably good results can be obtained in only a few iterations instead of over a hundred. This is shown in Fig. 3.3 and, in a sense, confirms experimentally the discussion in Section 3.3.1.

3.4.4 Mini-Batch Size b

The mini-batch size b defines the number of samples that are aggregated when computing the gradient. Mini-batch gradient descent corresponds to $1 < b < n$. The limit values $b = 1$ and $b = n$ corresponds to stochastic and full-batch gradient descent. The higher the batch size, the fewer weight updates one will make at each epoch. The lower the batch size, the higher the number of weight updates, and -in theory- the fewer epochs will be necessary. We chose among the power of two since this will offer us interesting optimisation opportunities during the translation in the ciphertext space.

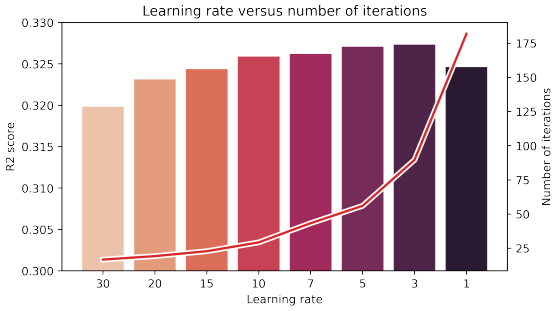


Figure 3.3: R2 scores and number of iterations before convergence for different learning rates.

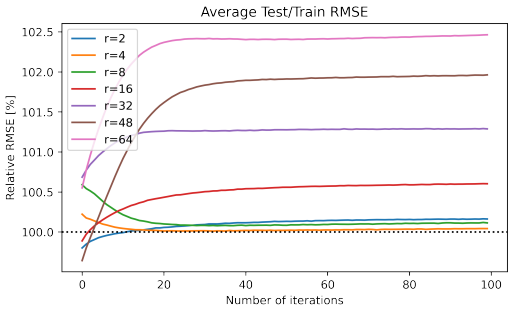


Figure 3.4: Overfitting of different latent space dimensions d.

Real-time Updates Important to be noted is that our two proposed protocols[2]have a different implementation of the mini-batch updates. These implementations stems directly from their design and whether there are interactive or not. The results presented here corresponds to the non-interactive implementation (i.e. the mini-batch gradients are computed with the latest available weight set $v_j^{(t)}$). We call this implementation: *real-time* weight updates. In the interactive implementation, the same weights are used during the whole iteration. While the end results are similar for both implementations, the interactive one logically requires more iterations to converge.

3.4.5 Regularisation Term L

Regularisation terms artificially add noise to force the model to be simpler. The aim here is mainly to prevent overfitting on the training data and keep good generalisation performances on unseen data. In regression tasks, this results in adding a L1 or L2 penalty term to the cost function defined in Eq. 3.2.

L1 (aka Lasso) regularisation tends to keep the most important latent factors and push the others towards zero. Intuitively, this is not what we want for our USER and ITEM matrices. Instead, we applied thus a light L2 penalty for the initial matrix factorisation. This intuition has been confirmed experimentally. Linear models are not prone to overfitting. However, we noticed that adding a L2 regularisation term, increased the convergence rate (but not the results). As the regularisation adds a new term in the loss function, the computational benefits from fewer iterations will be cancelled out by the extra cost of evaluating the new term in the loss function. This is why we won't use any regularisation when training the linear model.

3.5 Implementation Results

Since we decided to study the use of homomorphic encryption for AD, it is important to keep in mind the notion of computational complexity. Homomorphic operations are by several

[2]see Chapter 4

Parameter	Value
Latent Space Dimension r	16
Testing fraction q_k	0.9
Learning rate η	7^4
Mini-batch size b	256
Regularisation term $L2$	1e-4

Table 3.1: Final parameter set used.

orders of magnitude slower than plaintext operations, each addition and multiplication must be carefully taken into consideration.

Number of `HOP` The goal during these first experiments was to minimise the number of required `HOP` while achieving correct regression scores. To minimise the amount of these costly operations, we have five possibilities:

- either reduce the number of iterations by
  1. reducing the latent space dimension r (Fig. 3.2b)
  2. reducing the testing fraction q_k (Fig. 3.2b)
  3. increase[3] the learning rate η (Fig. 3.3)
- or reduce the number of multiplications per iteration by
  4. choosing a HE-friendly gradient update method
  5. augment the batch size b

Contributions (4) and (5) are discussed in more detail in the next chapter.

Overfitting Danger As for all data science problems, overfitting is the danger. In this case, if the trained model does not generalise well on unseen data, it could have dramatic consequences. One of the main sources of overfitting is here the dimension of the latent space. The higher the dimension, the more complex are our models, the higher the variance and the higher the danger of overfitting. This behaviour is experimentally confirmed in Fig. 3.4. Keeping a small q_k and limiting the amount of training data also tend to produce unwanted underfitting effects (i.e. too simple model).

Final Model Parameters Based on our plaintext analysis, we defined the model parameters summarised in Table 3.1. We defined these parameters to keep correct results whilst keeping an eye on the number of HE-unfriendly computations and the overfitting danger. We obtain final cross-validated RMSE and MAE scores of 0.2822 and 0.2226 respectively. For a complete discussion on these results, see Section 5.1.

[3]Increasing the learning rate should be done with care as the overshooting danger is important and the possibility of losing the convergence property exists.

Chapter 4

Privacy-Preserving Recommendation System

This chapter gets to the heart of the matter: how to address the inherent privacy problems of AD. We will first discuss some privacy attacks related to the training of machine learning models. We will outline the privacy problems that arise from these attacks and present our architecture and system model. We will describe and implement our two proposed protocols and, finally, evaluate them with toy parameters.

4.1 Existing Privacy Attacks

Many recent publications [47, 48, 49] point out the privacy breaches of modern machine learning models. Among the 3 big categories of ML attacks (i.e. confidentiality, integrity and availability attacks [50]), we will only discuss the confidentiality attacks as privacy preserving methods is the focus of this work. Confidentiality attacks can fall in either data extraction or model extraction attacks. Since the obtained model is here public, model extraction attacks lose their sense. We will discuss here de-anonymization, gradient leaks and inference attacks (see Fig. 4.1) as they had a direct impact on our design choices.

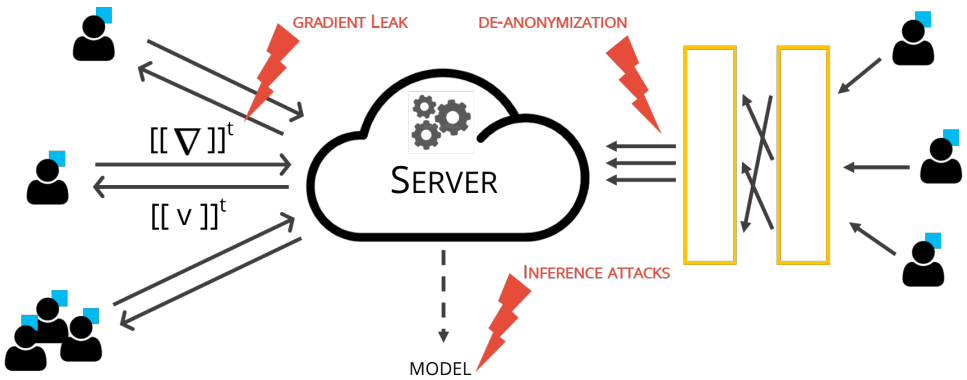


Figure 4.1: A diagram of the three studied privacy attacks. We find a federated learning approach on the left and a anonymization process (e.g. mixnet) on the right.

4.1.1 Deanonymisation

For a long time it was publicly accepted that removing key identification information (e.g. names or passport numbers), would yield an anonymized dataset. Over time, more and more studies highlighted the de-anonymization possibilities when the original dataset is crossed with external public data (e.g. zip code, gender, etc). The same open government principles introduced to promote transparency and accountability[1] [51], backfires against the citizens and publicly available (or sold) voter registration lists offer a dangerous de-anonymization catalyst. An experiment with a voting list of a small town in the Massachusetts was able to identify 30% of the voters by birth date and gender alone and a frightening 97% with birth date and zip code [52]. Similar approaches succeeded to de-anonymize the Netflix Prize dataset by crossing it with IMDb movie ratings [53].

4.1.2 Gradient Leak

Removing key identification information is thus not sufficient. Other leaks occur during training on private data.

As shown in several studies, intermediate gradients of an iterative optimisation algorithm, may leak confidential information [54, 55, 53, 56]. But, contrary to our intuition, these gradients leaks even if they are aggregated over a batch of local data [49]. This is a serious privacy breach for many FL models.

There are two possible stages of gradient leaks: (i) during the aggregation step (user side) and (ii) during processing (server side). The first situation happens when local gradients of b different USERS are aggregated to form one unique gradient. A gradient interception at this stage can directly compromise user data (e.g. see §3 of [54] for a simple example). Cryptographic techniques can be used to assure a secure aggregation with other users (e.g. oblivious aggregation [57]). The gradient leak at server side is often inevitable since the gradients must be processed to update the weights. Addressing this leak is thus more complex.

To the best of our knowledge, we did not find any work on the gradient leaks of *linear models*. The following assumptions are inspired from [49]. They achieve significant results of leaking gradients with a multi-layer perceptron (MLP) image classifier and not a linear regression (LR) model, while a MLP is more complex than a LR, one could argue that more leakage would occur with a LR.

In classical Federated Learning applications, a user i computes local gradients on his private training data of n samples. It aggregates the n gradients and sends ∇_i to the server. Classical gradient attacks use dummy variables, feed these into the model and get dummy gradients. When these dummy gradients are optimised to match the original gradient as closely, the dummy variables tend to approach the original data.

Here, instead of having 1 user aggregating n gradients, we have n users aggregating 1

[1] And supply us with a dataset

gradient. If we define a *super-user* (e.g. a trusted aggregator) who gathers and aggregates all gradients before transmitting the result to the server (i.e. the attacker), both problems are equivalent and the latter can be reduced to the first one. We conclude thus that single gradients aggregated over n USERS leak information.

Defense strategies exist to prevent gradient leakage. Three strategies are proposed in [49]:

1. **Adding noise to the gradients.** The idea behind this strategy is to hide the real values of the gradients to the server. Using the principles of Differential Privacy [58], one can add noise to each gradient of a dataset in a way to keep the same statistical properties of it. They showed that Gaussian (or Laplacian) noise with a variance of 10^{-2} is needed to avoid leakage.

2. **Architecture strategies**. The aggregation of b local gradients (i.e. a batch size of b) results in more difficulties to make the attacker converge as b increases. The higher b, the more possible permutations exist and choosing the right GD direction for the dummy gradients becomes hard. To solve this issue, the authors of [49] proposed to update only one of the dummy samples at each iteration. This seems promising, but still, the number of required iterations increase with the batch size. Going even further, [55] showed[2] that the distortion, resulting from the batching process, is not uniform over all samples. Even when aggregating over one hundred local gradients, while most of the images are unrecoverable, around 1/10 of them are approximately recognisable.

3. **Gradient compression**. The idea here is to set an arbitrary number of gradients, with small magnitudes, to zero. [59] shows that at least 20% of the gradients have to be set to zero in order to leak a limited amount of information. Theoretical works are able to hold accuracy constant by compressing more than 300 times the gradients [60]. This results in a sparsity of 99% (far above the required 20%) and results in an efficient way to prevent gradient leakage. Following our research question, this strategy has not been studied. However it is an interesting area of research.

Indirectly we will use architecture strategies by aggregating over a (large) batch of different USERS. But as this is the aim of this work, we will especially address the gradient leak with homomorphic encryption. This falls thus under the first category.
Differential Privacy could have been used, but it often reduces accuracy [61] and it is proved that linear regressions and SGD experience high sensitivity to differential noise append [62].

4.1.3 Membership Inference Attacks

The last privacy issue covered here concerns inference attacks. Authors of [63] have shown that attackers can find out whether or not a sample has been used for training. For example, in the case of a ML hospital model trained on medical data, the attackers were able to know if a certain individual was a patient at the hospital or not.

[2] still in the image domain with a MLP

The attacker could be either an inside user participating in the process or a malicious outsider. In the first case, the user tries to infer other participants confidential data using the data exchanged during the training phase. In the latter, everyone having access to the model (even a black-box access), can make inferences about the training samples.

Among the possible defense strategies, we have Secure Multiparty Computation (SMC) and Differential Privacy (DP) [64]. HE adds more overhead but also prevent inside inference attacks. As data remains encrypted during the training process, a user that would infer other users' data would break the scheme. Outsider attacks are more difficult to address and our proposed solution is described in Section 4.4.

4.2 Privacy Problems

We elaborate here on the privacy issues of each of the 4 problems reported in Section 2.4.

1. **Initial matrix Factorizasion** To make the initial factorisation of matrix R in U and V, all users must share their private information. Privacy-preserving HE-solutions for matrix factorizasion exists [65] and their work can easily be adapted to fit in our scheme. We consider the matrix as initially factorised and refer to their work to achieve a privacy-preserving matrix factorisation using homomorphic encryption and data masking. Furthermore, we based our 3-party structure (see next section) on their work to ensure continuity.

2. **Emission of a Preference by an user** The privacy issues of this problem are solved thanks to the FL approach. The USER receives the publicly available latent vector of the new question and locally computes his prediction. The users never share their latent vector during the process and this part is thus not a problem.

3. **New User Cold Start** The new user receives the item matrix and learning parameters. Once he has answered a set of initial questions, he trains a linear regression model on these question encodings and his answers (3.4). Again, the whole process happens on his device which prevents privacy issues.

4. **New Item Cold Start** In Chapter 2, we decided to let the USERs solve the ITEM cold-start problem. Several private informations (USERs profile and preference) must be shared during the training phase. This raises several privacy concerns.

We will thus focus our study on solving this last problem as the 3 others are either already efficiently solved, or does not have privacy concerns.

4.3 System Model

We use a 3-party model consisting of USERs, MLE and SYSTEM. A precise description and comparison of the three parties can be found in Table 4.1.

The **main idea behind this 3-party protocol** is that either SYSTEM has the private data *masked* or MLE has it *encrypted*.

User	MLE	System
who: a user looking for a recommendation	**who:** a unique intermediate with high computational power	**who:** parliament or decision policy
number: significant and distributed	**number:** unique, do not collide with SYSTEM	**number:** unique, do not collide with MLE
role: emit a personal preference when among the k selected	**role:** aggregate the user-specific data to obtain (train) new item representations; publish the additive masks	**role:** initializes the encryption process; share the public keys and publish the final item representations
have: their past emitted preferences r_{ij}, own latent vector u_i and the public ITEM representations V	**have:** training parameters (1/2) and additive masks for each item	**have:** decryption keys and the remaining training parameters (2/2)
have not: access to others' preferences or personal latent vectors	**have not:** access to USERs personal data and (clear) item representations	**have not:** access to USERs personal data, additive masks of MLE and (unmasked) item representations

Table 4.1: A description of the 3 parties in our protocols. MLE is unique, although an organizasion with multiple MLEs could relax the non-collusion hypothesis.

4.4 Threat Model

The SYSTEM wants to obtain the profile of a new item j by solving a linear regression on the latent vectors of a subset of USERS S^k. As the USERS do not want to share their profile for privacy reasons, SYSTEM must learn the weights of v_j in a privacy-preserving manner. To do so, we introduce between USER and SYSTEM, a intermediate MACHINE LEARNING ENGINE (MLE) that executes the training phase. SYSTEM becomes then a supervising authority that outsources the (costly) training phase.

The USERs are the owners of the private data and the (MLE, SYSTEM) pair forms the adversaries. These are considered as *honest-but-curious* adversaries. As summarized in [66], such adversaries always follow the defined protocols, but are curious and try to find out extra information about the data they deal with. SYSTEM initialises the cryptographic primitive and share the public keys. The USERs encrypt their personal data with it. Under the assumption that SYSTEM and MLE do not collide, they agree to share their personal profile and preference with MLE. Avoiding that MLE and SYSTEM obtain the final item

representation, adds a lot of flexibility to the overall system (e.g. evaluation of a stopping criterion, continue the training after a first publication of the item representation, prevent inference attacks, etc.). This brings us to a 3-party organisation with data-masking already used in several publications [65, 67]. In addition, this organisation is the same as in [65] which solves the first problem (see previous section), and so, we built our privacy-preserving cold-start protocols in the continuity of their work.

Using this 3-party structure, the goal is that SYSTEM learns and publish a correct and private profile v_j. Correct signifies here that the difference between the profiles computed in encrypted form and the one that would have been computed in clear, is negligible. Private means that no USER data is compromised (neither to the SYSTEM nor to the MLE) during the learning process and that its result remains unknown to both parties.

Based on the confidentiality attacks discussed in Section 4.1, we make the following observations:

- USER data must be aggregated in a privacy-preserving manner.
- To ensure *end-to-end confidentiality*, we want to prevent all forms of gradient leak during processing.
- The item representation v_j has to stay hidden throughout the learning phase. If MLE knows the tuple (η, v_j^k, v_j^{k+1}) of the parameter update step (1.1), he will be able to isolate the gradient between two iterations. This happens with a batch size of $b = 1$ or if MLE deviates from the defined protocol and decides not to aggregate all received gradients.
- As our model should generalizes well to the addition of meta-data (e.g. gender, zip code, etc.), data anonymization is not sufficient. Sharing this meta-data in clear, even after anonymization (e.g. through a mixnet), is holy bread for de-anonymization attacks.
- SYSTEM must initialise the first weights and send it to all USERs to be sure MLE does not initialise a second (factice) cryptographic primitive where he has the decryption key.

Although it is beyond the scope of this study[3], we note that if the attacker has access to the intermediate gradients in clear, so far, no existing mechanisms exist to detect integrity and availability attacks on the model [68]. The MLE could then build adversarial gradients and completely manipulate the model as he wishes. Therefore, outsourcing a federated learning process also raises many integrity concerns.

[3]We focus our study on confidentiality attacks and won't address integrity attacks.

4.5 Design Components

4.5.1 Homomorphic Encryption for Arithmetic of Approximate Numbers

The confidentiality reasons discussed in the previous section, led us to use homomorphic encryption schemes to protect USERs privacy during the training phase. We will use the Homomorphic Encryption for Arithmetic of Approximate Numbers (HEAAN) [69] scheme for our two end-to-end encrypted protocols. We used HEAAN[4] for its native support and efficiency with arithmetic on floating-point numbers. HEAAN supports both addition and multiplication operations together with a refreshing procedure: it is thus a FHE scheme.

RLWE Problem

Like many other FHE schemes, HEAAN bases its security assumption on the Ring Learning With Error (RLWE) problem. RLWE extends the classical Learning With Error problem which (basically) tries to efficiently distinguish encrypted vectors from uniformly random ones. The aim of the introduction of RLWE is to use the classical LWE formulation over other, more efficient, algebraic structures (i.e. rings of integer polynomials).

Here we define a simplified version of the RLWE problem [70] which has been adapted from the original Ring-LWE formulation of Lyubashevsky et Al. [71].

Let us fix an integer N and a prime q. Let $\mathcal{R} = \mathbb{Z}[x]/\left(x^N + 1\right)$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let χ be an error distribution over $\mathcal{R}_q$. For a given polynomial $s \in \mathcal{R}_q$, we define the Ring Learning With Error distribution RLWE $_{q,\chi,s}$ by:

$$\mathrm{RLWE}_{N,q,\chi,\boldsymbol{s}} = \left\{(b,a) \in \mathcal{R}_q^2 : a \leftarrow \mathcal{U}\left(\mathcal{R}_q\right), e \leftarrow \chi, b = a \cdot s + e.\right\}$$

Here $\mathcal{U}\left(\mathcal{R}_q\right)$ is the uniform distribution over $\mathcal{R}_q$. **The RLWE search problem is to distinguish the uniform distribution in $\mathcal{R}_q^2$ with the RLWE distribution**. Regev showed [72] that the LWE problem is at least as hard to solve as several worst-case lattice problems. Under the idea that lattice-based problems are hard, the LWE problem is also considered as hard (i.e. infeasible in probabilistic polynomial time). An analogue rationale can be made for the RLWE problem.

Algorithms of HEAAN

The RLWE problem adds thus some random noise e which makes the encryption scheme probabilistic and guarantees the hardness assumptions. However, this noise grows the more homomorphic operations are executed (roughly speaking, noises are added during homomorphic additions and multiplied together during multiplications).

HEAAN is different from other HE schemes by the way it perceives approximate numbers. Their main idea is to consider the approximate arithmetic errors inherent to

[4] Also known as *CKKS* following the name of its authors.

computers, as part of the encryption noise. By encoding the message m in the MSB and the noise e in the LSB, the significant figures are not likely to be destroyed by the noise if e remains small enough compared to m. This is opposite to BGV and BF-schemes where noise is stored in the MSBs and the message in the LSBs [73, 74, 75]. During a multiplication, the approximation errors are multiplied by each other and the bit size increases exponentially. This is also the case for plaintext operations. However, the traditional rounding ability of the plaintext space is not possible in the ciphertext space. Instead, HEAAN proposes a *rescaling* procedure that reduces the size of the ciphertext modulus and removes the error located in the LSBs of the message.

As this way of proceeding mimics the ordinary approximate arithmetic, the authors proved that their scheme has at most 1 bit more precision loss compared to unencrypted floating-point arithmetic [69].

The HEAAN scheme consists of the following algorithms: `KeyGen`, `Encode`, `Decode`, `Encrypt`, `Dec`, `Add`, `Mult` and `Rescale`; which are defined in [69]. We note here that the HEAAN scheme encodes (N/2)-dimensional vectors $\in \mathbb{C}$ into integer polynomials of the cyclotomic ring. We won't use the imaginary parts of these vectors although some works encoded a second real vector in it and achieved performance improvements [].

After each multiplication, the depth of the circuit, the bound on the noise B and the ciphertext level l increases while the number of bits of modulus decreases. Once l reaches the maximum level L, the modulus has become so small that it is almost not decipherable anymore. Proposed one year later, bootstrapping for HEAAN [76] allows refreshing the ciphertext and rewind the modulus. Their work was followed by several improvements of the original bootstrapping algorithm [77, 78].

HEAAN for Machine Learning

Among the many FHE schemes, we used the HEAAN scheme for its native support and efficiency for arithmetic on floating-point numbers (especially frequent in ML applications). Furthermore, the HEAAN scheme is based on the BGV-scheme which allows for efficient SIMD[5] operations [70] which are frequent in ML applications.

HEAAN supports both addition and multiplication operations together with a refreshing procedure. Like other FHE schemes, they allow to evaluate arbitrary polynomial functions. Non-linear functions are often used in ML applications and can't directly be evaluated with HEAAN. Although, one can approximate these with Taylor's approximation, this remains costly. On the other hand, TFHE [79] uses lookup tables and can efficiently evaluate these non-linear functions.

Actually, as we do not use any non-linear functions, our choice of using HEAAN is in adequation with the literature [80, 81, 82, 83].

[5]Single Instruction Multiple Data

4.5.2 Semantic Security

The scheme presented above is semantically secure. More formally, this means that it is impossible for a probabilistic polynomial time algorithm that has access to the tuple $([[x]]_{pk}, len(x), pk)$ to infer any extra information about the plaintext x with non negligible probability. [12]

We assume that the parameter set of the underlying RLWE problem (N, q, χ, s) is appropriately chosen to achieve a sufficient security level λ (e.g. 80 or 128). As discussed in [84], this means that the degree of the polynomial ring respects:

$$N_{min} \geq \frac{\lambda + 110}{7.2} \log(Q), \tag{4.1}$$

and is rounded to N, the smallest power of two greater than N_{min}. Modulus Q represents the largest modulus of the circuit. The use of (4.1) is advocated by the authors of HEAAN, however, the LWE estimator of Albrecht et al. [85] is also widely used to assess the security of a (R)LWE problem.

4.5.3 Additive Masking

We define the following additive masking scheme[6] (adapted from [12]). Let $a \oplus b$ define the bitwise exclusive-or (XOR) of two binary strings a and b and let integer l denote the size of our binary secret string x.

$\texttt{KeyGen}(l)$ generates a single-use mask m by choosing a string from $\{0,1\}^l$ according to the uniform distribution.
$\texttt{Mask}_k(m)$ encrypts secret x through: $c = m \oplus x$
$\texttt{Unmask}_k(c)$ decrypts ciphertext $c \in \{0,1\}^l$ with: $x = m \oplus c$.

The general idea behind this scheme is that with c fixed, for every possible x, there exists a mask m such that $c = \texttt{Mask}_m(x)$. Thanks to additive masking, neither MLE nor SYSTEM has the sensitive data in clear. For example, let x represent some data a USER need, but which must remain hidden to MLE and SYSTEM. MLE has $[[x]]_k$ and SYSTEM owns the decryption key k.

- Using the homomorphic properties and with $l = N$, MLE will: (i) generate a mask m with `KeyGen(l)`, (ii) compute $[[c]]_k = \texttt{Mask}_m([[x]]_k)$ and (iii) send $[[c]]_k$ to SYSTEM. This latter has the HE-decryption key k, but not m and can thus only obtain c.
- A USER needs this x. We have: (i) SYSTEM sends him c, (ii) MLE sends him m and (iii) the USER executes $x = \texttt{Unmask}_m(c)$.

[6] aka one-time-pad or Vernam's cipher.

4.6 Proposed Protocols

To tackle the new ITEM cold-start problem, we came up with two possible privacy-preserving solutions: either a (online) *interactive* or (offline) *non-interactive* training protocol. As the name suggests, in the first setup, the k selected users have to stay online during the whole process, while in the latter, users can go offline once all their data has securely been transferred to the MLE.

Homomorphic Operations We note that the ciphertext homomorphic operations (`HOP`) in HEAAN are executed slot by slot (e.g. as the Hadamard product does). To add or multiply a ciphertext with a scalar, one must thus duplicate this scalar in each slot of the ciphertext. For ease of writing, we will use `MultCC` (resp. `MultCP`) for ciphertext-ciphertext (resp. ciphertext-plaintext) multiplications.

Abuse of notations As we will use the same ITEM j throughout this section, we will omit the index j. The pair $(r_{ij}, v_j,)$ becomes thus (r_i, v).

General idea Both protocols share the same general structure, but differ in the way they handle step 3.

- **Step 1** SYSTEM initialises the protocol and shares the public parameters.
- **Step 2** Each USER encrypts his private data and sends it to MLE.
- **Step 3** MLE applies the Gradient Descent algorithm without (Π_A) or with (Π_B) USERs interaction.
- **Step 4** MLE masks the obtained item representation and sends the masked, encrypted vector to SYSTEM. He also sends the (unencrypted) mask to all USERs.
- **Step 5** SYSTEM decrypts the masked vector and sends it to all USERs.

Both protocols are schematized in Fig 4.2 and 4.3 and the related pseudocodes can be found in Appendix A2.

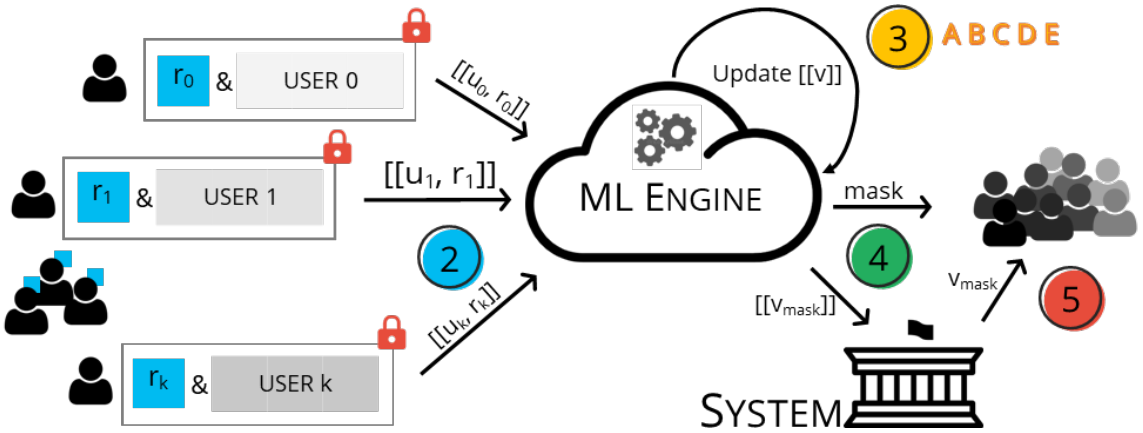


Figure 4.2: Non-interactive protocol A. The steps 2-5 correspond to the description of the protocol in Section 4.6.1. Step 1 (Initialisation) is not shown. After step 2, all USERs can go offline and MLE takes care of the rest of the job.

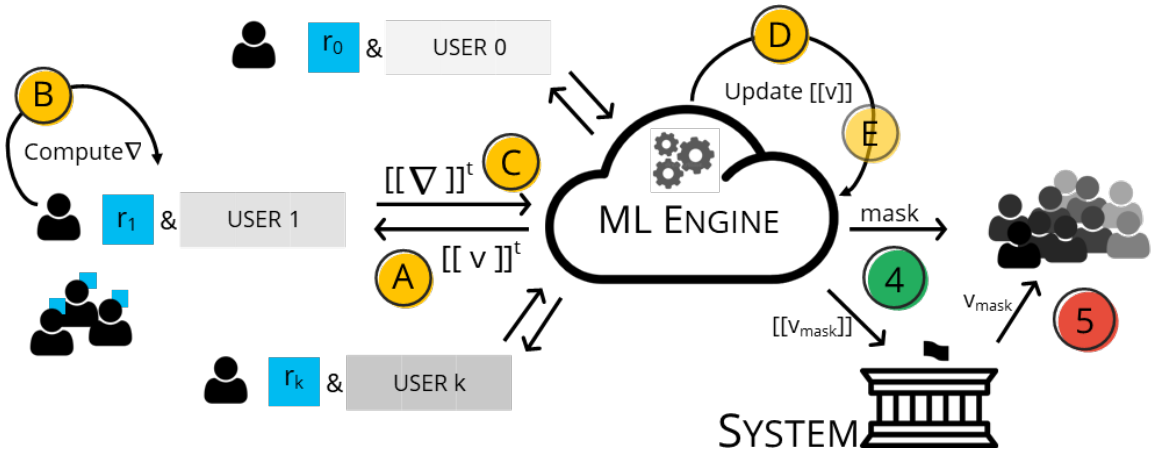


Figure 4.3: Interactive Protocol B. Based on the federated learning approach, USERs compute locally their gradient at each iteration. The yellow A-B-C-D-E steps constitute the third step. Step 1 (Initialisation) and 2 (Preparation) are not shown.

4.6.1 Π_A Non-Interactive Training Protocol

Main Idea

In this non-interactive protocol, once a USER has encrypted and transmitted his private data to the MLE, he can go offline. His job is then finished and we won't need him further. MLE applies the gradient descent algorithm in a fully-homomorphic way until the stopping criterion is met (see Section 4.6.3 for a complete discussion on the stopping criterion).

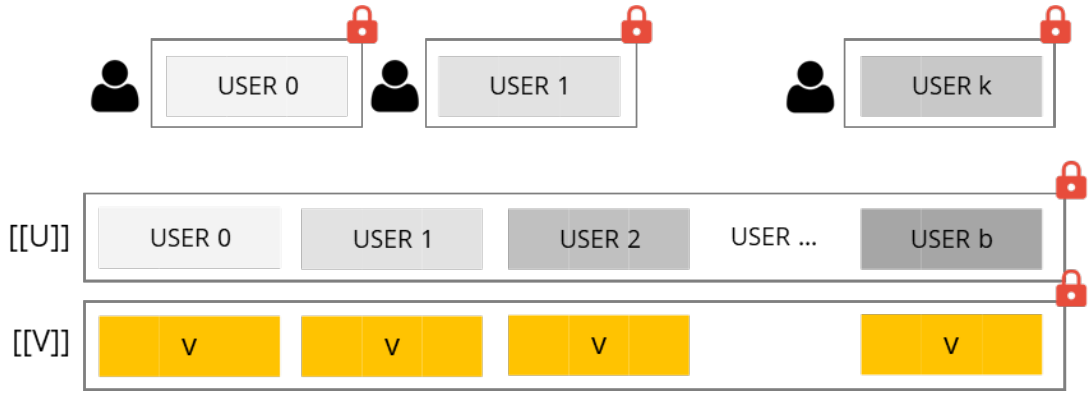


Figure 4.4: Packing b USERs in a single ciphertext [[U]]. We duplicate the initial latent vector v to fit all slots of [[V]] (yellow).

Description

To take advantage from the high number[7] of available ciphertext slots, we will implement packing techniques[8] and parallelise some computations. In particular, we will pack n_b USERs in a same ciphertext [[U]]. To ease ciphertext manipulation operations, all USER private data must fit in a vector p of length $|p|$ equal to a power of two. We will define the batch size b in a way that all USERs of a same batch fit in a unique ciphertext as in Fig. 4.4. We have N/2 slots in a ciphertext of degree N and can thus pack $b = \frac{N}{2|p|}$ USERs together. This defines the number of mini-batches (i.e. the number of ciphertexts):

$$n_b = \frac{k}{b} = \frac{2k|p|}{N} \tag{4.2}$$

The complete protocol is schematised in Fig 4.2 and the computational complexity of each step is reported in Table 4.2. More formally, we have:

Step 1. System initialisation The SYSTEM generates a (pk, sk) key pair and distributes pk to the MLE and all USERs. Both MLE and USERs save the pk. The MLE defines the learning parameter set (η, b). Specific to the HEAAN scheme, the SYSTEM generates rotation and relinearization keys (respectively noted as $k_{rot(i)}$ and k_{relin}) and send them to the MLE. The SYSTEM also generates the first random weights:

$$v^{(0)} = \begin{pmatrix} a_0 & a_1 & \dots & a_d & -1 \end{pmatrix} \tag{4.3}$$

where a is a random vector sampled from a normal distribution of mean 0. Finally, he duplicates it b times to fill all the slots, encrypts it and sends $[[v^{(0)}]]$ to the MLE.

[7] Needed to ensure the hardness of the RLWE problem, see Section 4.8.2

[8] Sometimes called batching, but we will stick to *packing* as we already have mini-batches in the gradient descent method.

Step 2. Users encrypt and outsource With the idea of minimizisg the number of `HOP`, we fit the preference, USER and ITEM bias in a same personal vector p of size $|p| = d + 2$.[9] Each USER i forms his private vector

$$p_i = \left(1, u_{i1}, \ldots, u_{id}, r'_i\right),$$

by prepending his latent vector u_i with 1 (for the ITEM bias) and adding $r'_i = r_i - b_i^{\mathcal{U}}$ at the end. The USER bias is directly subtracted from his preference r_i. He encrypts p_i with SYSTEM pk and sends $[[p_i]]$ to the MLE.

Step 3. Gradient Descent by the mle

- **Step 3a. Aggregation** The MLE shuffles and packs the k received personal vectors $[[p_i]]$ into n_b ciphertexts $[[U]]$ containing each b personal vectors. $[[U]]$ is thus a $N/2$-dimensional vector packed in a single ciphertext, but for clarity reasons we will represent it here as a $b \times |p|$ matrix.

$$[[U]] = \begin{pmatrix} 1 & u_{11} & \ldots & u_{1d} & r'_1 \\ \vdots & \vdots & \ddots & & \vdots \\ 1 & u_{b1} & & u_{bd} & r'_b \end{pmatrix} \tag{4.4}$$

  This packing process requires $(k - n_b)$ rotations and additions as described in Appendix A1. Shuffling the personal vectors while packing them is important to keep the stochastic character of the iterative method. The steps B-C-D are executed for each of the n_b ciphertexts.

- **Step 3b. Computing the error** E The packed personal ciphertext is `multiplied` with $[[v^{(t)}]]$ to obtain a raw version of E:

$$[[E_{raw}]] = [[U]] \cdot [[v]] = \begin{pmatrix} b & u_{11}w_1 & \ldots & u_{1r}w_r & -r'_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b & u_{k1}w_1 & \ldots & u_{kr}w_r & -r'_k \end{pmatrix}$$

  Each row contains all the terms of the inner product. To obtain a clean $[[E]]$ MLE has to sum the rows, extract this sum and duplicate it to all slots of the ciphertext. This process is described in Appendix A1 and consumes 1 level. We take advantage from this consumed level to directly incorporate the learning rate η and prevent 1 additional ciphertext-plaintext multiplication. This produces the final vector E:

$$[[E]] = \begin{pmatrix} \eta s_1 & \ldots & \eta s_1 & 0 \\ \vdots & \ddots & & \vdots \\ \eta s_k & & \eta s_k & 0 \end{pmatrix}, \tag{4.5}$$

  with s_i the sum of row i in $[[E_{raw}]]$. As the last column of [[U]] isn't needed anymore and would poison the next $\nabla = E \odot U$ computation, we set the last column of [[E]] to zero.

[9] As a remainder, d is the size of the latent space.

- **Step 3c. Computing the gradients** The MLE `multiplies` [[E]] with $[[U]]$ and sums the columns to obtain the final gradient $[[\nabla_v^{(t)}]]$. When all slots are used, summing the columns does not consume any level (see Appendix A1).
- **Step 3d. Parameter update** Last but not least, the MLE uses the gradient to update v_j.

$$[[v^{t+1}]] = [[v^t]] \boxminus [[\eta \nabla_v^{(t)}]] \tag{4.6}$$

- **Step 3e. Bootstrapping** At the end of the iteration, a bootstrapping step is executed to restore the ciphertext modulus to its original level.

Step 4. Publication of the result Once a stopping criteria is met (see Section 4.6.3 for a detailed discussion), a mask vector $mask_j$ is generated and added to $[[v_j]]$ to obtain $[[v_j + mask_j]]$. This last value is then sent to SYSTEM for decryption and distribution to the USERS.

Steps	**Where**	`HE.AddCC`	`HE.MultCP`	`HE.MultCC`
[3a] Aggregation	@MLE	$b-1$	-	-
[3b] Computation of E	@MLE	$2\log_2(d)+1$	1	1
[3c] Computation of ∇	@MLE	$\log_2(b)$	-	1
[3d] Weight update	@MLE	1	-	-

Table 4.2: Number of `HOP`s for protocol A. This corresponds to 1 iteration and 1 mini-batch. For n iterations over n_b mini-batches, these values must be multiplied with a factor nn_b. Missing steps do not require any `HOP`. Step 3e (not shown) is the bootstrapping step.

4.6.2 Π_B Interactive Training Protocol

Main Idea

The idea here is that the MLE outsources the costly `MultCC`s of the gradients to the online USERS. These have their personal latent vectors u_i in clear and the costly `MultCC` inner product of step 3 becomes a gentle one with equivalent `MultCP`.

As depicted in Fig. 4.3, we obtain the following exchange in step 3 (for each iteration):

a. The MLE sends the latest item representation v_j to all k USERS.

b. Each USER i performs locally the sensitive inner product and computes so his local gradient. Next, he sends his (encrypted) local gradient to MLE.

c. MLE aggregates all (encrypted) gradients,

d. and updates the latent vector v_j.

These 4 points represent steps 3a-b-c-d hereunder. By outsourcing the (outsourced) data back to the users (i.e. federated learning), we obtain a double gain: (i) we remove all costly `MultCC` and (ii) we delegate -and parallelise- the costly computations to the devices of the k USERS. This parallelization enables us to reduce the overall depth of the homomorphic evaluation circuit.

Description of the Protocol

The core of the protocol happens in step 3. It can be used in a levelled setting or with a bootstrapping step. As the distributed nature of FL does not permit to compute efficiently real-time weight updates, there is no longer any reason to compute mini-batches and we use a single batch ($b = k$ and $n_b = 1$).

Step 1. System initialisation The SYSTEM generates a (pk, sk) keypair and sends pk to the MLE[10]. The MLE defines the learning parameter set (η, b). Specific to the HEAAN scheme, the SYSTEM generates $k_{rot(i)}$ and k_{relin} and send these to MLE. USERs only need the rotation keys k_{rot}[11]. The SYSTEM also generates the first random weights:

$$v^{(0)} = \begin{pmatrix} a_0 & a_1 & \dots & a_d \end{pmatrix} \tag{4.7}$$

where a is a random vector sampled from a normal distribution of mean 0. It encrypts it and sends $[[v]]^{(t=0)}$ to all USERs.

Step 2. User gets ready Each USER issues his preference r_i and gets ready to participate in the learning process. He encodes his personal profile twice: once without modification and a second time after multiplying it with η.[12] This prevents one additional multiplication at MLE side. He also computes and encodes $r'_i = r_i - b_i^{\mathcal{U}}$ to include his USER bias in the learning process.

Step 3. Gradient descent We use here the federated learning approach and repeat the following steps for each epoch t,

- **Step 3a. Publication of the latest parameters** The MLE distributes his last version of the weights. As he only possesses these in encrypted form, he sends $[[v]]^{(t)}$ to all k USERS.
- **Step 3b. Local computation of the gradient** Each USER i receives $[[v]]^{(t)}$ and enriches it with his confidential data (u_i, r_i) to obtain his private gradient of epoch t:

$$[[\nabla_v]]_i^{(t)} = (u_i[[v]]^{(t)} - r_i)u_i \tag{4.8}$$

  We notice that the only `HOPs` required here are additions and plain `multiplications`. He then sends his encrypted gradient $[[\nabla_v]]_i^{(t)}$ back to the MLE.

[10]The USERs perform only plain additions/multiplication and don't need pk

[11]Technically, they only need $\log(d)$ rotation keys.

[12]We recall that we made the hypothesis that USERs won't act as malicious adversaries. Here, a malicouis USER could multiply with a disproportionate η

- **Step 3c. Aggregation of the gradients** The MLE aggregates the received gradients.

$$[[\nabla_v]]^{(t)} = \sum_i [[\nabla_v]]_i^{(t)} \tag{4.9}$$

- **Step 3d. Update of the parameters** Finally, the MLE uses the aggregated gradients to update v, again with a single homomorphic subtraction.

$$[[v]]^{(t+1)} = [[v]]^{(t)} \boxminus \eta[[\nabla_{v_j}]]^{(t)} \tag{4.10}$$

- **[Optional] Step 3e. Bootstrapping** When the number of iterations per bootstrapping (*ipb*, see Section 4.8.2) is reached, a bootstrapping step is executed to restore the ciphertext modulus to its original level.

Step 4. Publication of the result Once a stopping criterion is met, a mask vector $mask$ is generated and added to $[[v]]$ to obtain $[[v + mask]]$. This last value is then sent to SYSTEM for decryption and distribution to the USERS.

Steps	**Where**	`HE.AddCP`	`HE.AddCC`	`HE.MultCP`	`HE.MultCC`
[3b] Local ∇	@USER	$1 + \log_2(d)$	-	2	-
[3c] Aggregation	@MLE	-	$k - 1$	-	-
[3d] Weight update	@MLE	-	1	-	-

Table 4.3: Number of `HOP`s for 1 iteration of protocol B. Steps not represented, do not require any `HOP`. Step 3e (not shown) is the bootstrapping step.

4.6.3 Stopping Criterion

There are numerous ways to stop the iterative gradient descent process. This process is also called early stopping [86]. Among them we have:

i. the maximum number of iterations is reached.

ii. the difference in the objective function for held-out data (i.e. validation set) is smaller than a threshold.

iii. the total absolute difference in the weights $v^{(t+1)}$ and $v^{(t)}$ is smaller than a threshold.

iv. the gradient norm is smaller than a threshold.

Item (i) is often used and uses the convergence property of gradient descent. However, this does not tell whether or not the algorithm has converged yet nor if the model is overfitting the training samples. Item (ii) implies a significant number of homomorphic multiplications to evaluate the objective function on the validation set. However, it provides a guarantee that the model keeps improving and generalises well. The idea underlying item (iii) and (iv) is similar. Once the norm of the gradients (or the difference between $v^{(t)}$ and $v^{(t+1)}$) is

sufficiently small, one can assume the gradient descent algorithm has converged.

To bear in mind, is that all these values are homomorphically encrypted and the comparison with a threshold is not that easy. Two main approaches exists.[87] uses the Newton approximation and arithmetic schemes (e.g. the HEAAN scheme) to evaluate the `sign(.)` function. The second approach is to use non-arithmetic operations (e.g. table lookup) to perform the comparison. The TFHE scheme is among the most efficient schemes for these non-arithmetic operations. [88] showed that using efficient packing and optimisation algorithms, arithmetic and non-arithmetic circuits have comparable amortised time to evaluate the `sign(.)` function. However, they conclude that TFHE still performs better when evaluating a single comparison.

Among the 4 items proposed above, only the first one does not require this encrypted comparison circuit. For the 3 others, we adopt the general notation $\|\Delta^{(t)}\| \leq w$ where $\Delta^{(t)}$ is the value of the stopping function and w the defined threshold. We define the sign of $z^{(t)} \equiv \|\Delta^{(t)}\| - w$ as the stopping criterion; when $\texttt{sign}(z^{(t_{final})}) = -1$, the method stops.

We could directly include the sign function in the parameter update of (1.1):

$$[[v^{(t+1)}]] = [[v^{(t)}]] + [[\texttt{sign}(z^{(t)})]] \cdot \eta \; [[\nabla_v]]^{(t)}]] \tag{4.11}$$

This allows us to bound the homomorphic circuit, but $t_{max} - t_{final}$ superfluous epoch would be evaluated. To avoid these redundant epochs, we could involve the SYSTEM and let it decrypt $[[\texttt{sign}(z^{(t)})]]$. Evaluating the `sign(.)` function on MLE side prevent SYSTEM from discovering $z^{(t)}$ (and thus $\Delta^{(t)}$ if he knows w or $\Delta^{(t+1)} - \Delta^{(t)}$ if not). This process leaks only 1 bit among the $|z|$ bits. This single-bit leak is hardly unavoidable since it defines if the MLE must continue the iterative method or not. The first solution (4.11) prevents this leaks at the cost of $t_{max} - t_{final}$ extra epochs.

Intuitively we would say that computing the gradient norm would require fewer multiplications than computing the objective function over a held-out set. However due to efficient packing techniques, both criteria could require the same amount of `multiplications`. We also note that evaluating a single comparison in HEAAN is costly (evaluating a circuit of depth $\mathcal{O}(\log n)$ for n-bits numbers [88]).

To conclude this discussion, we recommend to use a combination of the first approach and one of the 3 others. For example, the MLE could evaluate the stopping criteria every *ipb* number of iterations. This process can then easily be parallelized with the bootstrapping step.

4.7 Privacy of the Protocols

We base the security of our protocols on the IND-CPA (i.e. semantic security) property of HE [12].

4.7.1 Privacy of user data

In protocol B, as USER data never leaves his device, his private data will never be directly compromised. In protocol A, ciphertexts containing private USER data are outsourced to the MLE. Under the hypothesis that MLE and SYSTEM do not collaborate, are honest-but-curious adversaries and the parameters of the RLWE problem are correctly defined, if MLE learns something more on the ciphertext than his size, he has broken the IND-CPA property of HEAAN. By doing so, he breaks the underlying RLWE problem and can thus break some worst-case lattice problems. As this is considered as infeasible, we can assume MLE will never learn anything more than what he is given.

SYSTEM will never receive any raw USER data and will never be able to learn anything. Since USER data is encrypted *end-to-end*, there is no danger of de-anonymisation.

4.7.2 Gradient Leaks

Even if USER data never leaves his device as in protocol 1, it can be indirectly compromised due to possible gradient leak attacks of the MLE. As gradients remain encrypted end-to-end, HE offers a solution to the inherent leak of gradient descent methods. The MLE can update the weights $v^{(t)}$ while all gradients remain totally encrypted. The intermediate weights $v^{(t)}$ remain encrypted during the exchange and there is no way to infer any information about the gradients.

4.7.3 Membership Inference Attacks

If the ITEM representation v is known by SYSTEM and MLE, they could generate dummy profiles u_d and make membership inference attacks to know whether or not u_d is present among the profiles of the k selected USERs. This is a minor attack as k can be large and the number of possible profiles u_d infinite. However the attacker could isolate some components of the latent space (e.g. a component contrary to the principles of the SYSTEM) and make inference attacks on this component only.

To prevent any form of membership inference attack, we kept all representations v_j hidden from SYSTEM and MLE thanks to an additive masking scheme.

4.8 Implementation

We implemented both protocols and evaluated them on our dataset with toy parameters. The results were obtained on a laptop with a Intel i7 CPU and 16GB of RAM. The codes of our 2 implemented protocols can be found on GitHub: `https://github.com/brabantm/AD-training`. All codes run on a single core.

4.8.1 Python Toy Example

To ease the comparison of the results between the models in clear and the ones encrypted in the ciphertext space, we kept the same language and implemented a Python toy example

for each of the 2 protocols. Python is definitely not the most efficient language for FHE, but again efficiency was not the main goal of this study.

To obtain results in reasonable time we used between 1/5 and 1/20 of the dataset for assessing the performances of our protocols. We rounded the number of training samples to a power of two to obtain fully packed ciphertexts. We kept the 9:1 ratio between the number of samples in the test and training set.

We used the Python `py-fhe` library [89]. We noticed that it has a non-conventional way of sampling the distributions during the setup and encryption steps. Instead of sampling the error e from a discrete Gaussian Distribution $\mathcal{DG}(\sigma)$, they sample it from a discrete triangular distribution[13] with probability $p = 0.5$. This gives a smaller σ and would potentially lead to security breaches. Therefore, we wouldn't recommend *py-fhe* for production purposes. However as a toy library, it is easy to get into and well-coded.

`py-fhe` implements the original bootstrapping algorithm proposed in [76], since then different improvements have been published [77, 78]. As shown in [78], faster bootstrapping reduces computing time by a factor of 30 while increasing utility by a factor of 50. This inefficient bootstrapping will stand out during execution.

4.8.2 Parameter Choice

We discuss here the parameters of the HEAAN scheme and explain the choices we made. All our used (toy) parameters are reported in Table 4.4.

Scaling Factor Δ We define the scaling factor $\Delta = 2^{25}$. These bits are not superfluous as bootstrapping drastically reduces precision [76]. Ideally we should have use two different scaling factors as in [70]: one for ciphertexts encodings and one for plaintexts. This enables us to consume only half a level during each `MultCP`. However, the library we used did not allowed for that, so plaintexts and ciphertexts are encoded using the 2^{25} scaling factor. We also define $\log \Delta_0 = 2^{10}$. This increase precision during bootstrapping, but all bootstrapping operations consume then $\log(\Delta\Delta_0)$ bits instead of $\log \Delta$.

[70] uses $\Delta = 2^{30}$ and $\Delta_0 = 2^5$, our choice keeps the same precision during bootstrapping while consuming fewer moduli bits during the overall `HOPs`. This comes at the price of a negligible precision loss during plain `HOPs` as the bootstrapping step is HEAAN's precision bottleneck.

Modulus $\mathbf{Q_0}$ The ciphertext modulus Q_0 depends on the depth of the homomorphic circuit that must be evaluated. We defined the initial modulus of all our ciphertexts as

$$\log(Q_0) = \log(\Delta_0\Delta) + ipb \cdot (L \cdot \log \Delta), \tag{4.12}$$

with ipb the number of iterations per bootstrapping and L the depth of the circuit.

[13] The discrete triangular distribution considered here samples values from $\{-1, 0, 1\}$ with respective probability $(1/p, p, 1/p)$.

Parameter	N	ipb_A	ipb_B	$\log Q_0$	$\log Q_1$	k	b_A	n_b
Set-I	64	1	12	635	1160	32	4	8
Set-II	256	1	12	635	1160	128	16	8
Set-III	1024	1	12	635	1160	128	64	2
Set-IV(ipb)	256	-	ipb	85	610	128	-	-
Set-V(k)	64	-	L	785	785	k	-	-

Table 4.4: Toy parameter sets used for evaluation. We do not claim any security level here. The 3 rightmost columns are problem-dimension parameters. The reported n_b are for protocol A. b_B is always equal to k and n_b (of protocol B) equal to 1. "L" stands for "levelled".

Modulus Q_1 We must also define a modulus Q_1. This second modulus is the highest possible modulus obtained during bootstrapping. However, a fraction of it is lost during the (homomorphic) evaluation of the bootstrapping circuit. We define Q_1 in a way to obtain Q_0 after the bootstrapping step. An analysis of our used library gives:

$$\log(Q_1) = \log(Q_0) + (8 + n_{taylor}) \cdot \log(\Delta\Delta_0) \tag{4.13}$$

with $n_{taylor} = 7$ the number of iterations of Taylor's degree 7 approximation of $\exp(x)$. We remind that the bootstrapping HOPs consume $\log(\Delta\Delta_0)$ bits instead of $\log(\Delta)$. This gives better precision during bootstrapping.

Depth L Both protocols have different depths L. Using the values of Table 4.2 and 4.3, we obtain the depths of both circuits $(L_A, L_B) = (3n_b, 2)$. For a same depth L, protocol B can make $3n_b/2$ times more iterations. We kept this ratio so both protocols have the same moduli and set thus $ipb_B = \frac{3n_b}{2}$ and $ipb_A = 1$.

Ring degree N For a same security level λ, the ring degree N is directly correlated with the highest modulus Q_1, which depends on the depth of the circuit L. In turn, L depends on the protocol and the dimensions of the problem. We can insert (4.13) in (4.1), solve for N^2_{min} and find the minimum degree ensuring λ bits of security. The final N is set to the smallest power of two larger than the obtained N_{min}.

4.8.3 Results

Performances

We report on the performance results of both protocols in Table 4.5. Regarding protocol B, step 2b requested the evaluation of k times two MultCPs. As this would normally be parallelised over the k USERS, this step was here the most time-consuming one. We noticed a strong decrease in computing time over the iterations (3 to 5x). We suppose this corresponds to the USERs being stored closer and closer to the CPU. We took the **maximum computing time** (i.e. first iteration) instead of the last (smallest) one. However, each USER could store his personal vector in a CPU-friendly manner and boost the performances.

Protocol A	**[3a] Aggr.**	**[3b] Err.**	**[3c] Grad.**	**[3e] Boots.**	**Total**
Set-I	4.109	1.946	0.803	51.12	57.978
Set-II	33.80	2.595	1.415	106.1	143.91
Set-III	195.7	14.82	10.13	3090.9	3311.5
Protocol B	**-**	**[3b] @User**	**[3c] Aggr.**	**[3e] Boots.**	**Total**
Set-I		0.0822	0.0119	54.5^a	3.9803
Set-II		0.9919	0.0323	102.12^a	7.9565
Set-III		4.046	0.0778	2639.5	2643.6
Set-IV(1)		0.6332	0.0240	58.79	59.472
Set-IV(L)		0.7852	0.0383	-	0.8235
Set-V(32)		0.1047	0.0041	-	0.1088
Set-V(128)		0.0949	0.0102	-	0.1051
Set-V(512)		0.0950	0.3962	-	0.4912

Table 4.5: Time performances (in seconds) of both protocols. The steps are noted between brackets. Step 3d has an average computing time less than 10^{-5} and is not shown. a exponent means this value is amortised over the number of iterations per bootstrapping ipb_B. Step 3b is very dependent of how memory is used. We noticed a strong decrease in computation time over the iterations. We took the **maximum** (i.e. first iteration).

Precision

This assesses the correctness of the model computed in a private manner w.r.t the one computed in clear. We didn't have the time and resources to make a complete cross-validated evaluation of the precision of both protocols. As an example, the dotted lines in Figure 4.5 and 4.6 show the *true* values and the solid one, the HE approximation of our protocols. With our choice of $\Delta = 2^{25}$, we observe a negligible difference between the encrypted and plaintext weights v_j.

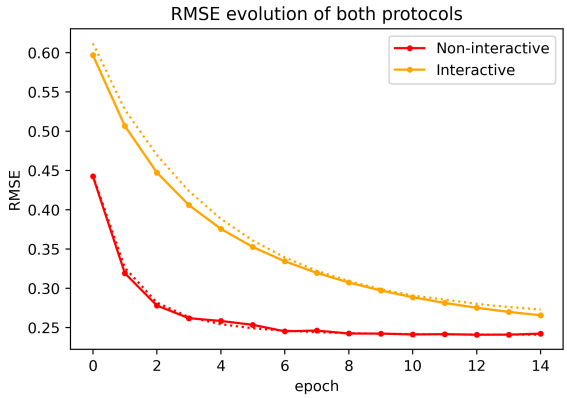


Figure 4.5: The RMSE evolution of both protocols for parameter Set-II. Dotted lines represent the *true* plaintext values.

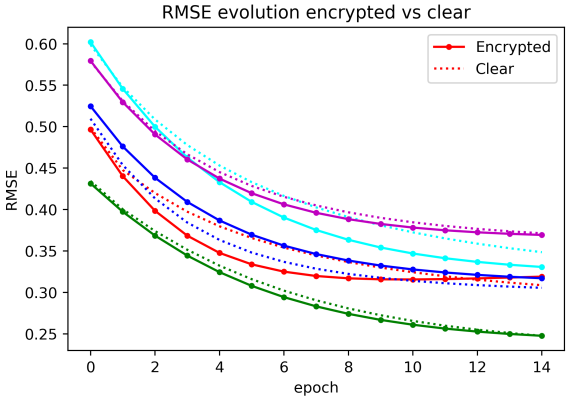


Figure 4.6: RMSE evolution of 5 ITEMs with their corresponding *true* plaintext values (dotted) for protocol B (Set-II).

Chapter 5

Discussion and Further Work

We discuss here the results obtained in both of the above chapters and give some directions of further work. Related work regarding the privacy-preserving protocols of chapter 4, will also be discussed.

5.1 Prediction Task

5.1.1 Performance

The final cross-validated regression scores of each of the 53 ITEMs are shown in Fig. 5.1[1]. With average RMSE values of 0.2822, the obtained results are encouraging but definitely not sufficient for such a demanding system. This is mainly due to the simplistic approach we kept with as after-thought the homomorphic evaluation of it. The limited amount of data we used also restrained our results.

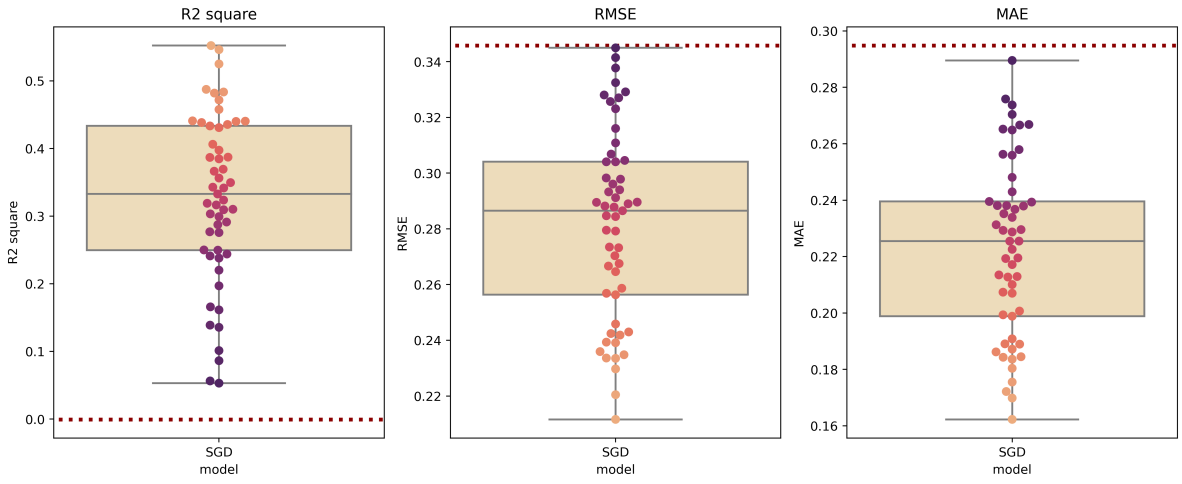


Figure 5.1: Boxplot and regression scores for each of the 53 ITEMs. The lighter, the better. We notice a series of ITEMs that pulls the results down. For comparison, the horizontal dotted line represents the constant (mean) model.

[1]We used the final parameter set as defined in Table 3.1

5.1.2 Further Work

We list hereunder a non-exhaustive list of further work related to our chosen approach for predicting single vote outcomes.

Matrix Factorisation

We believe that sticking to the matrix factorisation approach, has numerous benefits. Firstly, it allows for a certain degree of interpretability of the responses as the final result are a linear combination of different terms. This makes the avatars transparent in the way it achieves the prediction. Secondly, the factorised profiles can easily be supervised and guardrails can be added to avoid undesired actions. Next, MF reacts well to the addition of external metadata. Adding this metadata is essential to fine-tune users' profiles. Finally, the bi-matrix approach fit well into the federated learning setup.

[18] used a Bayesian approach and combined Gaussian Process with latent factor models to predict regional vote outcomes. They achieved better results with this combination than with each of the models individually. [90] combines MF with Generalised Linear Models and use high-dimensional historical data to factorise regional vote outcomes. Generalised linear models seem an interesting direction to continue the study at the level of individual users.

Latent Space

A thorough analysis of the latent space must be carried out to understand all the ins and outs of it. As it is meant to evolve with time, it should be supervised to avoid dangerous drifts. Efficient data-visualisation tools could be used to monitor and track this sensitive space.

In our eyes, the latent space is one of the cornerstones for raising awareness of the different issues at stake and make them comprehensible to all. For example, one of the 2021 Swiss votations concerned the *Economic Partnership Agreement with Indonesia*. The decomposition of the question in different causes helps to understand all the ins and outs of it. While one might have thought that economic issues would be the only predominant issue here, it was quite the opposite. The import of Indian palm oil (and its environmental and health consequences) was at the heart of the referendum.[2] Being able to explain the prediction results and their link with other domains is also a possible solution for the cognitive bandwidth problem described in Sec. 1.1.1.

More Data for Solving the Cold-Start Problems

Out of curiosity, we tried out some linear and non-linear models and found slight performance improvements. However as the literature suggests [18, 27], we believe that adding external data is essential for better cold-start solutions. Other emerging technologies as

[2]https://www.swissinfo.ch/eng/swiss-direct-democracy-could-undo-a-free-trade-deal-with-indonesia/46422650

natural language processing [91] could extract information from bills and votings and so help solve the cold-start problem.

Bias

As in every ML model, the bias-variance trade-off is sensitive matter [92]. Especially in this system, bias could have dramatic consequences. A (too) high variance in the models would result in a possible overfitting on the preferences of the k selected USERs. Being among the k selected would then give your preference a greater importance. The selection process of the k USERs is thus a highly sensitive task and must best represent all participants. Another source of bias is the design bias. An interesting discussion regarding models inherently biased by their designers can be found in [93].

5.2 Privacy-Preserving Cold-Start Protocols

5.2.1 Comparison of Both Protocols

Both protocols are different by the way they handle the problem. The interactive one requires USERs implication and availability, while the non-interactive one gets the job done on his own.

We are aware HE is often used to reduce the USER-side communication and computation overhead of other techniques (e.g. secure multi-party computation). Our interactive protocol Π_B goes in the opposite direction and requires user participation. However only `MultCP` are performed on USER side and this allows for low overhead and high flexibility. By keeping all gradients encrypted end-to-end, even aggregated ones, we prevent confidentiality breaches and limit the possibilities of integrity attacks.

Computational Complexity

	Party	`AddCP`	`AddCC`	`MultCP`	`MultCC`
Protocol A	MLE	-	$n_b\ (b + 2\log_2 d + \log_2 b + 1)$	n_b	$2n_b$
	USER	-	-	-	-
Protocol B	MLE	-	k	-	-
	1 USER	$1 + \log_2 d$	-	2	-

Table 5.1: Comparison of the complexity and depth of both protocols for 1 iteration. n_b is the number of mini-batches. As a remainder, $n_b = 1$ and $b = 1$ for protocol B.

Table 5.1 compares the complexity of protocols. It clearly highlights the differences between both protocols. In the non-interactive protocol Π_A, a USER shouldn't make any computation at all and it is the MLE that makes them all. Due to the centralised character of the protocol, MLE can make *real-time gradient updates* (see discussion in Section 3.4.4). Thanks to packing techniques, the first protocol has a depth of $3n_b$ levels per iteration.

	Public Key	Rotation Key	Relinearization Key
Protocol A	✓	✗	✗
Protocol B	✗	$\log_2(d)$ keys	✓

Table 5.2: SYSTEM keys necessary for the USERs in both protocols. The reason behind the $\log_2(d)$ rotation keys is explained in Appendix A1

Although this is $N/6d$ times fewer levels than the naive way (4.2), the depth of the circuit remains consistent ($\mathcal{O}(n)$). If we give up the real-time updates and consider only a single batch, we obtain a depth in $\mathcal{O}(1)$ (see Appendix A3).

On the other hand, the interactive protocol Π_B delegates some computation to the USER. Interesting to point out is that by doing so, USER data never leaves his device and, at the same time, we avoid **all** costly `MultCC`. But to avoid a depth of the circuit linear in k, all multiplications are parallelised and real-time gradient updates are not possible anymore. Making mini-batches loses its sense and we form one unique batch. We obtain then a depth of 2 instead of $2k$.

Communication Costs

Protocol A has only a limited communication cost as only 1 ciphertext is exchanged between USER and MLE (step 2). Furthermore, the only key that must be exchanged is the public key. Protocol B has a significant communication overhead which is proportional to the number of iterations. At each epoch, $2k$ ciphertexts are exchanged between USER and MLE. Depending on the size of the underlying RLWE problem (and thus the size of the ciphertexts), this could become problematic. Furthermore $\log(d)$ rotation keys must also be transmitted to all USERs to allow ciphertext rotations. This adds an extra communication overhead. The keys exchanged in both protocols are reported in Table 5.2.

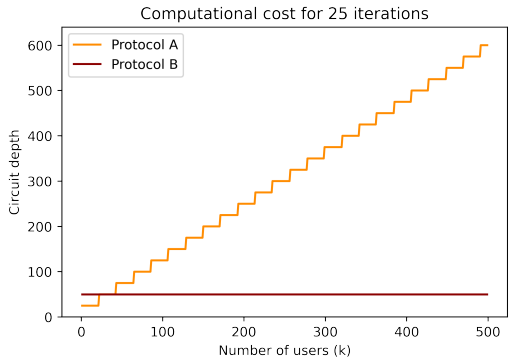
Leveled or Not

Protocol A can hardly be used in a levelled setting. The number of consumed levels would explode (see Fig. 5.2a) and the ciphertext size will follow this trend. For example, with parameter Set-II, protocol A consumes 48 levels per epoch while protocol B consumes only 2 of them[3]. Moreover these levels are `MultCP` and are executed way faster then `MultCC`.

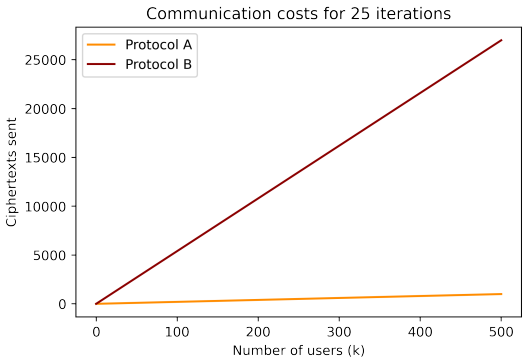
On the contrary, protocol B can be used in both settings as the number of consumed levels remains low. Using protocol B in a levelled setting is even more interesting since this drastically reduces the training time. For example, with parameter Set-II, MLE is able to update the parameters in 1s. If we do not take any communication latency into consideration and assume all USERs have the same computational power as used here, 15 iterations could be executed in 14.57s in the levelled setting against 14m46s with bootstrapping (and 36min for protocol A)[4].

[3]Or even one single level if we use a $\sqrt{\Delta}$ scaling factor as discussed in Section 4.8.2

[4]However this must be balanced by the fact that we do not use the latest bootstrapping improvements



(a) Circuit depth of both protocols versus the number of selected USERs k



(b) Communication cost (in ciphertext sent) of both protocols for 25 iterations.

Figure 5.2: We fixed $N = 2048$ and $d = 16$. Keys exchanged are considered as ciphertexts for (b).

Convergence Rate

As the non-interactive protocol can compute the gradients using the latest available weight set, evaluating mini-batches make sense and n_b weight updates are made at each iteration. The convergence rate of protocol A is thus higher than for protocol B. This is experimentally confirmed on Fig. 4.5.

Scalability

We discuss here how our protocols scale with an increase of a parameter, the other staying constant.

- **Increase of N** Protocol A takes advantage of the number of slots to pack a higher number of USER profiles in a same ciphertext. An increase in N, reduces n_b and so does the complexity (and depth) of the iteration circuit. However, by increasing N, we increase the size of all ciphertexts and make all computations slower.
- **Increase of d** If the size of the latent space d increases, the opposite as above happens and the depth of protocol A increases (4.2). Protocol B does not experience any significant change.
- **Increase of k** Protocol B scales well to an increase of the number of selected USERs. Its complexity remains unaltered. Procotol A will suffer from an increase of n_b (4.2) and so will the depth of the circuit. To absorb this increase in depth, the ring degree must be increased to keep λ constant and we observe $N \propto \sqrt{kd}$ for protocol A.

5.2.2 Related Work

Due to the recent GDPR regulations and the constantly increasing number of ML applications, a wide range of privacy-preserving training protocols and schemes have emerged.

as discussed in Section 4.8.1

Among this jungle of schemes, there is something for everyone. For curious readers, we refer to these interesting surveys [94, 95].

As mentioned in our research question and in section 2.4, we restricted our study to the iterative, homomorphic based, linear regression privacy-preserving training. We report here some works that could have been used in our use case and that, among many others, are worth reading. If we relax the homomorphic constraint, several schemes could be used with for each of them their strengths and drawbacks. Regarding secure distributed federated learning, we recommend [96]. For alternative privacy-preserving MF techniques, we suggest garbled circuits [97] and differential privacy [64, 98].

If we relax the iterative constraint, numerous schemes solve the regression problem with the least squares method [66]. If we relax both constraints, many other works are related to privacy preserving federated learning [96, 63]. Finally, for the case of hybrid recommendation systems using neural networks and complex activation functions, we recommend [15, 99].

5.2.3 Potential Use Cases

Our two proposed protocols are a mix of ideas from several different protocols [70, 65, 100]. They offer highly private ways to train gradient descent models on distributed data. While protocol A is a rather classical way to outsource the training using HE, protocol B is more unusual and has specific constraints (e.g. USERs must stay online and agree to do heavy computations). However we believe that protocol B could be used in specific applications (e.g. high value-added applications dealing with highly private data and a large number of users). And especially if the organiser has a limited computation power. By using accelerated gradient descent methods (e.g. Nesterov momentum [70]) one can increase the convergence rate, reduce the number of iterations and only require the users to stay online for a few minutes.

5.2.4 Further Work

First it would be interesting to study the way our protocols would react when implemented using the latest bootstrapping improvements and evaluated with 128 bits security parameters. We believe protocol B would keep its efficiency due to the shallow depth of its circuit and the absence of `MultCC`.

Recently [101] showed that approximate homomorphic encryption schemes (e.g. HEAAN) can verify the IND-CPA property and still be completely insecure. Their attacks on most implementations of HEAAN were successful. They define an extension IND-CPA^D which is secure against their attack at the cost of a decrease in precision. A study of an IND-CPA^D HEAAN scheme and its performance implications would be necessary to ensure the privacy of our protocols.

We believe that efficient HE-friendly packing techniques are crucial to make FHE

usable. We implemented a horizontal packing technique for gradient descent methods in $\mathcal{O}(n)$ per iteration, but suggest that vertical packing would be way more efficient. Real time update wouldn't be possible anymore. In a similar way as in [102] and as discussed in Appendix A4, one could already execute the outer product $u_i \otimes u_i$ and outsource a $d \times d$ matrix (wrapped in ciphertext) instead of a vector. This could have many advantages and reduce the depth of the circuit.

Although the use of HEAAN seems to fit well with linear models, more extensive models require complex non-linear functions. For such models, other schemes, such as TFHE, seem more appropriate [79].

Among the three musketeers of secure computation[5], we studied only (a fraction of) homomorphic encryption. Secure Multi-party Computation and Functional Encryption, the two other valiant musketeers, offer both a wide range of possibilities. Lightweight oblivious aggregation combined with gradient compression techniques seems to us a promising area of research to address the privacy problems related to gradient leaks in this particular use case.

[5]https://www.esat.kuleuven.be/cosic/blog/the-three-musketeers-of-secure-computation-mpc-fhe-and-fe/

Conclusion

When progress is driven by the promising results of AI and machine learning, privacy is often set aside. But as many recent publications suggest, machine learning models retain, leak and undermine personal data. We discussed the examples of de-anonymization, gradient leak and inference attacks from the perspective of a voting assistant avatar.

More technically, we formulated four general problems which, we believe, constitute the core problems a augmented democracy avatar must solve. We defined a collaborative filtering recommendation system and built two privacy-preserving protocols to address the new item cold-start problem. We showed their feasibility by implementing two toy examples and evaluated them with a Smartvote[6] dataset.

Our matrix factorisation approach is versatile and offers interesting generalisation possibilities (e.g. the addition of metadata or stopping criteria). However, it does provide relative accuracy. The addition of external data sources and the use of generalised linear models, seem a promising direction of further research. The distributed approach allows each user to keep control of its profile. The use of homomorphic encryption when the profile is to be outsourced ensures end-to-end privacy, but makes it computationally heavy and difficult to scale. This could be remedied by exploring alternative secure computation tools.

To the best of our knowledge, this is a first step towards a primary technical construction of an augmented democracy avatar. It should be seen as a proof-of-concept that augmented democracy -while preserving users' confidentiality- is possible. We do not claim to have *the* solution, but rather wish to highlight the path taken and the reflections we have had.

Augmented democracy comes with numerous dangers. However, it is based on some idealistic ideas that are interesting to be considered. Simply by its existence, it brings to light certain problems of current democracies and highlights the importance of bringing citizens closer to democracy.

In a near future, our problem formulation and privacy-preserving protocols could further evolve to build personalised voting assistants. As most public consultations (or referenda) often achieve limited participation levels, such systems could help and assist citizens in their duty and bring a wind of change to these public consultations.

[6] A Swiss Voting Assistant Application.

Bibliography

[1] Nancy Bermeo. On democratic backsliding. *Journal of Democracy*, 27(1):5–19, 2016.

[2] Richard Wike and Shannon Schumacher. Democratic rights popular globally but commitment to them not always strong. *Pew Research Center*, 2020.

[3] Fred Deveaux. Democracy perception index – 2020. *Dalia Research*, 2020.

[4] Susan Bell, Josh Benaloh, Michael D Byrne, Dana DeBeauvoir, Bryce Eakin, Philip Kortum, Neal McBurnett, Olivier Pereira, Philip B Stark, Dan S Wallach, et al. Star-vote: A secure, transparent, auditable, and reliable voting system. In *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, 2013.

[5] NewB. NewB, notre banque coopérative, belge, éthique et durable. `https://www.newb.coop/fr`.

[6] César Hidalgo. A bold idea to replace politicians. 2018. Available at `https://www.ted.com/talks/cesar_hidalgo_a_bold_idea_to_replace_politicians`.

[7] Raya Al-Dalou and Emad Abu-Shanab. E-participation levels and technologies. *The sixth International Conference on Information Technology*, 2013.

[8] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694, 2020.

[9] The Computational Democracy Project. Polis. `https://compdemocracy.org/Polis/`.

[10] Georg Lutz and Pascal Sciarini. Issue competence and its influence on voting behavior in the swiss 2015 elections. 22(1):5–14. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/spsr.12206.

[11] Sonia Royo, Vicente Pina, and Jaime Garcia-Rayado. Decide madrid: A critical analysis of an award-winning e-participation initiative. *Sustainability*, 12(4):1674, 2020.

[12] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press.

[13] Yiannis Tsiounis and Moti Yung. On the security of elgamal based encryption. In *International Workshop on Public Key Cryptography*, pages 117–134. Springer, 1998.

[14] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[15] Qian Lou, Bo Feng, Geoffrey Charles Fox, and Lei Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. *Advances in Neural Information Processing Systems*, 33, 2020.

[16] Charles L Byrne. Lecture notes on iterative optimization algorithms. *Department of Mathematical Sciences, University of Massachusetts Lowell*, 2014.

[17] Meng-Hsiu Tsai, Yingfeng Wang, Myungjae Kwak, and Neil Rigole. A machine learning based strategy for election result prediction. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1408–1410.

[18] Vincent Etter, Mohammad Emtiyaz Khan, Matthias Grossglauser, and Patrick Thiran. Online collaborative prediction of regional vote results. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, page 233. IEEE. Number: CONF.

[19] Michael S. Lewis-Beck. Election forecasting: Principles and practice. 7(2):145–164. Publisher: SAGE Publications.

[20] Brendan O'Connor, Ramnath Balasubramanyan, Bryan Routledge, and Noah Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 4, 2010.

[21] Priyavrat Chauhan, Nonita Sharma, and Geeta Sikka. The emergence of social media data and sentiment analysis in election prediction. *Journal of Ambient Intelligence and Humanized Computing*, 12(2):2601–2627, 2021.

[22] Daniel Gayo-Avello. I wanted to predict elections with twitter and all i got was this lousy paper. *A balanced survey on election prediction using Twitter data*, 1204, 2012.

[23] Felipe G Contratres, Solange N Alves-Souza, Lucia Vilela Leite Filgueiras, and Luiz S DeSouza. Sentiment analysis of social network data for cold-start relief in recommender systems. In *World Conference on Information Systems and Technologies*, pages 122–132. Springer, 2018.

[24] Smartvote. Description des méthodes – recommandation de vote smartvote. 2019.

[25] Luis Terán and Andreas Meier. A fuzzy recommender system for eelections. In *International Conference on Electronic Government and the Information Systems Perspective*, pages 62–76. Springer, 2010.

[26] José Miguel Cruz, Maria Fernanda Boidi, and Rosario Queirolo. Saying no to weed: Public opinion towards cannabis legalisation in uruguay. *Drugs: Education, Prevention and Policy*, 25(1):67–76, 2018.

[27] Vincent Etter, Julien Herzen, Matthias Grossglauser, and Patrick Thiran. Mining democracy. In *Proceedings of the second ACM conference on Online social networks*, pages 1–12, 2014.

[28] Nicola Maggini. The explanatory model: The determinants of youth voting choices. In Nicola Maggini, editor, *Young People's Voting Behaviour in Europe: A Comparative Perspective*, pages 77–115. Palgrave Macmillan UK.

[29] James W. Simmons. Voting behaviour and socio-economic characteristics: The middlesex east federal election, 1965. 33(3):389–400. Publisher: [Canadian Economics Association, Wiley].

[30] David Winant, Joachim Schreurs, and Johan A. K. Suykens. Latent space exploration using generative kernel PCA. In Bart Bogaerts, Gianluca Bontempi, Pierre Geurts, Nick Harley, Bertrand Lebichot, Tom Lenaerts, and Gilles Louppe, editors, *Artificial Intelligence and Machine Learning*, Communications in Computer and Information Science, pages 70–82. Springer International Publishing.

[31] Alper Bilge, Cihan Kaleli, Ibrahim Yakut, Ihsan Gunes, and Huseyin Polat. A survey of privacy-preserving collaborative filtering schemes. *International Journal of Software Engineering and Knowledge Engineering*, 23(08):1085–1108, 2013.

[32] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49:136–146, 2015.

[33] Data For Progress. What the hell happened. `https://wthh.dataforprogress.org/`.

[34] Politools. Smartvote project. 2021. `https://politools.net/`.

[35] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer, 2007.

[36] Rong Hu and Pearl Pu. Enhancing collaborative filtering systems with personality information. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 197–204. Association for Computing Machinery.

[37] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, 2009.

[38] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3203–3209. International Joint Conferences on Artificial Intelligence Organization.

[39] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

[40] Alan Julian Izenman. Linear discriminant analysis. In *Modern multivariate statistical techniques*, pages 237–280. Springer, 2013.

[41] Xiaopeng Li and James She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 305–314, 2017.

[42] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[43] Léon Bottou and Olivier Bousquet. *The Tradeoffs of Large Scale Learning.*, volume 20. Journal Abbreviation: Optimization for Machine Learning Publication Title: Optimization for Machine Learning.

[44] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[45] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[46] John Canny. Collaborative filtering with privacy. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 45–57. IEEE, 2002.

[47] Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: Model inversion attacks and data protection law. 376(2133):20180083.

[48] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, pages 587–601, 2017.

[49] Ligeng Zhu and Song Han. Deep leakage from gradients. In *Federated Learning*, pages 17–31. Springer, 2020.

[50] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[51] Debbie Rabina. Open government: Collaboration, transparency, and participation in practice, daniel lathrop, laurel ruma (eds.). o'reilly, sebastopol, CA (2010), 402 pp. $44.99 (paperback), ISBN: 978-0-596-80435-0. 28:129–130.

[52] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110, 1997.

[53] Tribhuvanesh Orekondy, Seong Joon Oh, Yang Zhang, Bernt Schiele, and Mario Fritz. Gradient-leaks: Understanding and controlling deanonymization in federated learning. *arXiv preprint arXiv:1805.05838*, 2018.

[54] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.

[55] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

[56] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706. IEEE, 2019.

[57] Elaine Shi, T.-H Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. *Privacy-Preserving Aggregation of Time-Series Data*, volume 2. Journal Abbreviation: NDSS Publication Title: NDSS.

[58] Praneeth Vepakomma, Tristan Swedish, Ramesh Raskar, Otkrist Gupta, and Abhimanyu Dubey. No peek: A survey of private distributed deep learning. *arXiv preprint arXiv:1812.03288*, 2018.

[59] J. Zhou, J. Li, E. Panaousis, and K. Liang. Deep binarized convolutional neural network inferences over encrypted data. In *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 160–167.

[60] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

[61] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, 2019.

[62] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[63] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 13–23, 2019.

[64] Arnaud Berlioz, Arik Friedman, Mohamed Ali Kaafar, Roksana Boreli, and Shlomo Berkovsky. Applying differential privacy to matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 107–114. Association for Computing Machinery.

[65] Jinsu Kim, Dongyoung Koo, Yuna Kim, Hyunsoo Yoon, Junbum Shin, and Sungwook Kim. Efficient privacy-preserving matrix factorization for recommendation via fully homomorphic encryption. 21(4):17:1–17:30.

[66] Irene Giacomelli, Somesh Jha, Marc Joye, C David Page, and Kyonghwan Yoon. Privacy-preserving ridge regression with only linearly-homomorphic encryption. In *International Conference on Applied Cryptography and Network Security*, pages 243–261. Springer, 2018.

[67] Guowei Qiu, Xiaolin Gui, and Yingliang Zhao. Privacy-preserving linear regression on distributed data by homomorphic encryption and data masking. 8:107601–107613. Conference Name: IEEE Access.

[68] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 3–14. Association for Computing Machinery.

[69] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437. Springer International Publishing.

[70] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Efficient logistic regression on large encrypted data. *IACR Cryptol. ePrint Arch.*, 2018:662, 2018.

[71] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, Lecture Notes in Computer Science, pages 1–23. Springer.

[72] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), September 2009.

[73] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[74] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.

[75] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[76] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 360–384. Springer, 2018.

[77] Kyoohyung Han and Dohyeong Ki. Better bootstrapping for approximate homomorphic encryption. In *Cryptographers' Track at the RSA Conference*, pages 364–390. Springer, 2020.

[78] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 34–54. Springer, 2019.

[79] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[80] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38. ISSN: 2375-1207.

[81] Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. Chimera: Combining ring-lwe-based fully homomorphic encryption schemes. *Journal of Mathematical Cryptology*, 14(1):316–338, 2020.

[82] Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. Simulating homomorphic evaluation of deep learning predictions. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 212–230. Springer, 2019.

[83] Qinju Liu, Xianhui Lu, Fucai Luo, Shuai Zhou, Jingnan He, and Kunpeng Wang. SecureBP from homomorphic encryption. ISSN: 1939-0114 Pages: e5328059 Publisher: Hindawi Volume: 2020.

[84] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Annual Cryptology Conference*, pages 850–867. Springer, 2012.

[85] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[86] Genevieve B Orr and Klaus-Robert Müller. *Neural networks: tricks of the trade*. Springer, 2003.

[87] Jean-Claude Bajard, Paulo Martins, Leonel Sousa, and Vincent Zucca. Improving the efficiency of svm classification with fhe. *IEEE Transactions on Information Forensics and Security*, 15:1709–1722, 2019.

[88] Ilia Iliashenko and Vincent Zucca. Faster homomorphic comparison operations for bgv and bfv. *Proc. Priv. Enhancing Technol.*, 2021(3):246–264, 2021.

[89] Saroja Erabelli. pyFHE - a python library for fully homomorphic encryption. Accepted: 2021-01-06T18:34:28Z Journal Abbreviation: Python library for fully homomorphic encryption.

[90] Alexander Immer, Victor Kristof, Matthias Grossglauser, and Patrick Thiran. Submatrix factorization for real-time vote prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2280–2290, 2020.

[91] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[92] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[93] Catherine D'Ignazio and Lauren F Klein. *Data feminism*. Mit Press, 2020.

[94] Marek Jawurek, Florian Kerschbaum, and George Danezis. Sok: Privacy technologies for smart grids–a survey of options. *Microsoft Res., Cambridge, UK*, 1:1–16, 2012.

[95] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim. Privacy-preserving deep learning on machine learning as a service—a comprehensive survey. 8:167425–167447. Conference Name: IEEE Access.

[96] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2019.

[97] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. Privacy-preserving matrix factorization. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 801–812, 2013.

[98] Jia-Yun Jiang, Cheng-Te Li, and Shou-De Lin. Towards a more reliable privacy-preserving recommender system. *Information Sciences*, 482:248–265, 2019.

[99] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. page 13.

[100] Guowei Qiu, Xiaolin Gui, and Yingliang Zhao. Privacy-preserving linear regression on distributed data by homomorphic encryption and data masking. *IEEE Access*, 8:107601–107613, 2020.

[101] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. *IACR Cryptol. ePrint Arch*, 2020:1533, 2020.

[102] Fengwei Wang, Hui Zhu, Rongxing Lu, Yandong Zheng, and Hui Li. A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent. *Information Sciences*, 552:183–200.

Appendix A

Appendices

1. Sum methods

We use a utility function `NaiveSum`($[[U]], d, scale, k_{rot}$). The length d defines the length of a USER vector. b of such USER vectors are packed in the ciphertext $[[U]]$. Factor *scale* defines if we sum the columns ($scale = 1$) or rows ($scale = d$).

Algorithm 1 Utility Sum Function

Input: A ciphertext vector [[U]], a packing length d, a scale factor *scale* and rotations keys k_{rot}

Output: The ciphertext [[U]] where all the *scale* slots are summed over a length d.

1: **function** NAIVESUM($[[U]], d, scale, k_{rot}$)
2: [[sum]] ← [[U]]
3: **for** i in `range`($\log_2 d$) **do**
4: step ← $scale \cdot 2^i$
5: [[tmp]] ← `HEAAN.Rotation`($[[sum]], step, k_{rot(step)}$)
6: [[sum]] ← `HEAAN.Add`($[[tmp]], [[sum]]$)
7: **end for**
8: **return** [[sum]]
9: **end function**

When summing the rows, only 1 slot per row has the right value after performing `NaiveSum`. This is because, rows are mixed together during the rotation steps. The remaining slots contain thus sum results over 2 different rows (i.e. USERs).

Algorithm 2 Sum the columns

Input: A ciphertext vector [[U]], a packing length d, rotations keys k_{rot} and the relinearization key k_{relin}

Output: The ciphertext [[U]] where the d columns are summed together.

1: **function** SumColumns([[U]], d, k_{rot}, k_{relin})
2: [[dirty]] ← NaiveSum([[U]], d, 1, k_{rot})
3: [[clean]] ← HEAAN.MultiplyPlain([[dirty]], mask, k_{relin})
4: **return** NaiveSum([[$clean$]], d, 1, k_{rot})
5: **end function**

Summing the rows, do not consume any level if all slots are used. This is because we use a *scale* factor and the whole row is rotated together.

Algorithm 3 Sum the rows

Input: A ciphertext vector [[U]], a packing length d, a number of rows c and rotations keys k_{rot}

Output: The ciphertext [[U]] where the c rows (of length d) are summed together.

1: **function** SumRows([[U]], c, d, k_{rot})
2: **return** NaiveSum([[U]], c, d, k_{rot})
3: **end function**

2. Pseudocodes

We show here some interesting pseudocode to help understanding the overal structure. The complete code with all 3 parties can be found on `https://github.com/brabantm/AD-training`.

Protocol A

We do not show the `Rescale` and `LowerModulus` steps required between each multiplication. We use a sequential circuit so their use is straightforward. The learning rate η is incorporated during the mask multiplication in Line 12.

Algorithm 4 Non-Interactive Protocol Π_A

Input: A ciphertext vector [[U]], a packing length d, a number of rows c and rotations keys k_{rot}

Output: The ciphertext [[U]] where the c rows (of length d) are summed together.

1: **function** GradientDescent$_A$($[[p_0]], \ldots, [[p_k]]$))
2: **while** StoppingCriterion not True **do**
3: $([[b_0]], \ldots, [[b_{n_b}]]) \leftarrow$ `MakeRandomBatches`($[[p_0]], \ldots, [[p_k]]$), b) ▷ Step 3a
4: [[v]] $\leftarrow$ `MakeEpoch`$_A$($[[b_0]], \ldots, [[b_{n_b}]]$), [[v]])
5: [[v]] $\leftarrow$ `HEAAN.Bootstrap`([[v]], keys) ▷ Step 3e
6: **end while**
7: **return** [[v]]
8: **end function**

9: **function** MakeEpoch$_A$(batches, c, d, k_{rot})
10: **for** [[U]] in batches **do**
11: $[[E_{raw}]] \leftarrow$ `HEAAN.Multiply`([[U]], [[v]], k_{relin}) ▷ Step 3b
12: [[E]] $\leftarrow$ `SumColumns`($[[E_{raw}]]$, d, k_{rot}, k_{relin})
13: $[[\nabla_{raw}]] \leftarrow$ `HEAAN.Multiply`([[E]], [[U]], k_{relin}) ▷ Step 3c
14: $[[\nabla]] \leftarrow$ `SumRows`($[[\nabla_{raw}]]$, $|U|/d$, d, k_{rot})
15: [[v]] $\leftarrow$ `HEAAN.Add`([[v]], $[[\nabla]]$) ▷ Step 3d
16: **end for**
17: **return** [[v]]
18: **end function**

Protocol B

We do not show the `Rescale` and `LowerModulus` steps required between each multiplication. We use a sequential circuit so their use is straightforward. The learning rate η is incorporated during the plain multiplication in Line 6.

Algorithm 5 Interactive Protocol Π_B

Input: A ciphertext vector [[U]], a packing length d, a number of rows c and rotations keys k_{rot}

Output: The ciphertext [[U]] where the c rows (of length d) are summed together.

function GradientDescent_User$_B$(([[p_0]], ..., [[p_k]]))
  while StoppingCriterion not True **do**
    Wait for [[v]] from MLE
    [[e_i]] ← `HEAAN.MultiplyPlain`([[v]], u_i, k_{relin}) ▷ Step 3b
    [[e_i]] ← `HEAAN.AddPlain`([[e_i]], $-r_i$)
    [[∇_i]] ← `HEAAN.MultiplyPlain`([[v]], $(lr \cdot u_i)$, k_{relin})
    Send [[∇_i]] to MLE
  end while
end function

function GradientDescent_MLE$_B$(batches, c, d, k_{rot})
  while StoppingCriterion not True **do**
    Send [[v]] to all k USERs ▷ Step 3a
    Wait for [[∇_i]] of all k USERs
    [[agg]] ← [[∇_0]]
    for i in `range`(1, k) **do** ▷ Step 3c
      [[agg]] ← `HEAAN.Add`([[agg]], [[∇_i]])
    end for
    [[v]] ← `HEAAN.Add`([[v]], [[∇]]) ▷ Step 3d
  end while
  return [[v]]
end function

3. User matrix

In a similar manner as in [102], we observe that the square of the loss function disappears when computing the gradient. We can then rearrange the gradient as:

$$\nabla_{v_j}\mathcal{L}(v_j, b_j) = \frac{1}{n}\sum_{i=1}^{n}(u_i v_j - r_{ij})u_i \tag{A.1}$$

$$= \frac{1}{n}\Big(\sum_{i=1}^{n}(u_i \otimes u_i)v_j - \sum_{i=1}^{n}(r_{ij}u_i)\Big) \tag{A.2}$$

$$= \frac{1}{n}\Big(Av_j - Y\Big), \tag{A.3}$$

Where A and Y can be completely pre-processed on USER side with their confidential (u_i, r_{ij}) pair. This halves the number of multiplications. However, each USER must now outsource a $(d+2) \times (d+1)$ matrix instead of a $(d+1)$ vector.

$$M_i = (A_i, Y_i) = \begin{pmatrix} 1^2 & u_{i1} & \dots & u_{id} & r_{ij} \\ u_{i1} & u_{i1}^2 & \dots & u_{i1}u_{id} & r_{ij}u_{i1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{id} & u_{id}u_{i1} & \dots & u_{id}^2 & r_{ij}u_{id} \end{pmatrix} \tag{A.4}$$

This is not problematic for protocol B since most slots are unused. Regarding protocol A, this means that we can pack $(d+2)$ times less USERs in a single ciphertext while halving the number of the multiplications. This is not cost-effective (as $d \geq 0$).

4. Real-Time for protocol A

We note that protocol A could also be used in a non-real-time setting (as protocol B). We would then update the weights once per iteration (instead of n_b times currently). In a similar way as in Appendix A, each USER packs his M_i matrix and outsources it. The MLE aggregates all matrices $M = \sum_i M_i$ and performs a **single** multiplication with v to obtain the gradient. After having sum the rows, we obtain then the gradient:

$$\nabla = \begin{pmatrix} \nabla_0 & - & - & - \\ \vdots & - & - & - \\ \nabla_d & - & - & - \end{pmatrix} \tag{A.5}$$

Where "-" represents dirty sums over 2 different rows. It remains for us to extract this sum (with a binary mask), and duplicate the first column to the remaining ones to obtain the final gradient.

In summary we can thus make non-real-time updates for Π_A and consume only 2 levels instead of the current $3n_b$ levels. This gives then for steps 2-3A-D of protocol A:

Step 1. System initialisation Identical as in 4.6.1. We remind that the v vector has the following form

$$v^{(0)} = \begin{pmatrix} a_0 & a_1 & \dots & a_d & -1 \end{pmatrix}, \tag{A.6}$$

duplicated to fill all slots.

Step 2. Local Preprocessing and Encryption Each USER i preprocess locally his data to form the A_i and Y_i matrices, respectively of dimension $(r' \times r')$ and $(r' \times 1)$ with $r' = d+1$ to include the bias term. He concatenates both matrices and obtains his personal data matrix M_i:

$$M_i = (A_i, Y_i) = \begin{pmatrix} 1^2 & u_{i1} & \dots & u_{id} & r_{ij} \\ u_{i1} & u_{i1}^2 & \dots & u_{i1}u_{id} & r_{ij}u_{i1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{id} & u_{id}u_{i1} & \dots & u_{id}^2 & r_{ij}u_{id} \end{pmatrix} \tag{A.7}$$

To ensure the privacy of his matrix M_i, each USER encrypts his matrix with the public key of SYSTEM

$$[[M_i]] = \textbf{HE.Encrypt}(M_i, pk_{\text{SYSTEM}}), \tag{A.8}$$

and sends it to the MLE. As soon as this is done, USER i can go offline. In the HEAAN scheme, we encoded the matrix row-wise (i.e. one ciphertext for each row of the M_i matrix).

Step 3. Gradient descent The MLE stores all received matrices M_i and executes for each iteration t,[1]

- **Step 3a. Aggregation by the mle** The MLE uses the additive homomorphic property of the scheme to aggregate the private matrices M_i.

$$[[M]] = \sum_{i=0}^{k} [[M_i]] \tag{A.9}$$

  He sets A as the first $(r+1)$ ciphertexts of [[M]] and Y as the last one.

- **Step 3b. Computation of the gradients** The MLE uses a single multiplication to compute the gradients:

$$[[\nabla_{v_j}]] = [[M]] \odot [[v]] \tag{A.10}$$

  However he must then sum the rows as in Appendix X and this consumes 1 level.

- **Step 3c. Parameter update** Last but not least, the MLE uses the gradients to update v_j with homomorphic additions.

$$[[v_j^{t+1}]] = [[v_j^t]] - \eta[[\nabla_{v_j}]] \tag{A.11}$$

- **Step 3d. Bootstrapping** At the end of the iteration, a bootstrapping step is executed to restore the ciphertext modulus to its original level.

[1]The general case with a (single) batch of size $b = k$ is here presented, the case of stochastic gradient descent or mini-batch gradient descent with $0 < b < k$ is a direct extension of it.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**