Radboud University Nijmegen

Institute for Computing and Information Sciences

Federated Regression Analysis on Personal Data Stores

Improving the Personal Health Train

Thesis MSc Data Science Specialization

Author:
C. van Aarle

Supervisor:
prof. dr. ir. Djoerd Hiemstra

Second reader:
prof. dr. Peter-Bram A.C. 't Hoen

October, 2021

Contents

Acknowledgement

I would like to express my deepest appreciation to D. Hiemstra for his guiding role in the making of this thesis. The effects of his week in week out presence may not be underestimated and affected the quality of the thesis and my well-being during this period. I would like to thank P. 't Hoen and N. Queralt-Rosinach for providing such a nice topic to work on. Their advice, feedback and introduction to the Personal Health Train are greatly appreciated. I was able to borrow some of the time of C. Sun and M. Kersloot, which is appreciated.

Abstract

Due to regulations and increased privacy awareness, patients may be reticent in sharing data with any institution. The Personal Health Train is an initiative to connect different data institutions for data analysis while maintaining full authority over their data. The Personal Health Train may not only connect larger institutions but also connect smaller, possibly on-device personal data stores, where data is safely and separately stored.

This thesis explores possible solutions in the literature that guarantee data-privacy and model-privacy, and it shows the practical feasibility when learning over a large number of personal data stores. We specifically regard the generation of linear regression and logistic regression models over personal data stores. We experiment with different design choices to optimise the convergence of our training architecture.

We discuss the *PrivFL* protocol [36] which takes into account both data-privacy and model-privacy when learning a regression model and is applicable to personal data stores.

We further propose a standardisation protocol, Secure Scaling Operation 2.0, that guarantees data-privacy for patients and experiments concluded that it improves convergence better than an adaptive gradient.

We implement an architecture that can learn over personal data stores and which preserves user privacy in *FedLinReg-v2* and *FedLogReg-v2*. While, in theory, no convergence is guaranteed, training over various datasets shows a difference of 0 to 0.33% in loss differences over both training and test sets compared to models that are centrally optimised. No parameter optimisation was necessary. The coefficients however may deviate from centrally trained models.

We were able to train regression models while preserving data-privacy (15-private) over 150 personal data stores in minutes. An even higher level of data-privacy will cause a strong linear increase in computation-time in relation to the amount of personal data stores included.

Chapter 1

Introduction

In recent decades, the quantity of digital information has grown to immeasurable numbers. This acquired data is distributed over data centres and personal devices across the world. It has pushed the development of machine learning solutions in healthcare that make use of data quantities like these.

More recently, privacy law enforcement puts extra requirements on the usage and storing of personal and processed data. Participating patients could also be hesitant to share their sensitive data that is stored somewhere unknown and shared to researchers to create useful models. Machine learning models will decrease in quality when they can only use part of the data or no data at all.

The Personal Health Train (PHT) is a Dutch architecture and solution to this problem, as it can apply database queries (analytics) [3] and generate machine learning models [16] when including multiple data sources. Current PHT architecture is still tested on a few large data collections with limited cooperating organisations. Limited research has been done on learning over personal data stores and this thesis explores the following gap.

One use case that is not yet considered is the generation of machine learning models on personal data stores. We explore the possibility to create informative models while shielding user data from other parties, and shielding model details from all users. Such a solution will result in users contributing to medical research while being able to maintain full authority over their data.

We build an implementation that can train logistic and linear regression models over isolated patient data, and that should decrease participant reticence in sharing data, and improve data authority. We extend the research to other machine learning (ML) models and summarise how they could be implemented under the same conditions.

1.1 Research Approach

This research proposes possible solutions to privately learn over personal data stores and develops a product to apply regression on personal data store in a privacy-preserving way. Therefore, it attempts to answer the following research questions regarding regression models. The research questions are all in the context of personal data stores.

RQ1: How do we ensure data-privacy and model-privacy when creating regression models by applying federated learning?

RQ2: How much do adaptive gradient and data standardisation improve convergence?

RQ3: How well does our architecture approximate linear regression and logistic regression while applying federated learning and preserving data-privacy?

> RQ3a: How do coefficients of our linear regression model, trained via federated learning, compare to the coefficients of classic linear regression?
>
> RQ3b: How do the odds ratios of our logistic regression model, trained via federated learning, compare to the odds ratios of classic logistic regression?
>
> RQ3c: How well can a linear regression model, trained via federated learning, fit the data compared to classic linear regression?
>
> RQ3d: How well can a logistic regression model, trained via federated learning, fit the data compared to classic logistic regression?
>
> RQ3e: How well can a linear regression model, trained via federated learning, predict compared to classic linear regression?
>
> RQ3f: How well can a logistic regression model, trained via federated learning, predict compared to classic logistic regression?

RQ4: How does a logistic regression model, trained via federated learning, converge when using mini-batch updates?

RQ5: How does the amount of personal data stores influence computation time?

> RQ5a: What is the influence of the amount of personal data stores in the network on the computation time for the central handler?
>
> RQ5b: What is the influence of the amount of personal data stores in the network on the computation time for every personal data store?

RQ6: How can we privately optimise other types of models in a federated way while preserving privacy?

> RQ6a: How do we apply federated learning to train regularised regression models, support vector machines and artificial neural networks?
>
> RQ6b: How do we apply privacy-preserving techniques to train decision trees and random forests?

Numerous papers have been written on the subject of federated learning protocols which decrease privacy concerns and cryptographic methods that can reduce data leakage. It is used to conclude RQ1 on how data and model-privacy can be guaranteed. RQ2 gives us insight in methods that should help convergence to an optimum. RQ3 is answered by practically implementing methods that abide some of the privacy measures posed in the background literature. RQ4 explores how mini-batch updating influences the convergence of our current setup (further explained in section 6.2.6). For RQ5 we analyse how the size of the network with personal data stores influences the run-time. In RQ6, we explore some other models that can be optimised with federated learning. It also explains an optimisation method for decision trees and random forests that do not fit the federated learning technique. All RQ's are answered in the context of the Personal Health train, further explained in section 2.2.

In summary, this research first tries to establish a literature research on the privacy concerns in federated learning. Second, it shows the practical feasibility of learning while preserving data-privacy, concerning convergence rate (performance metrics) and computation-time (speed).

This thesis starts by providing required background literature on the subject in section 2. Section 3 explains the possibility to standardise the distributed data. With 4 we dive deeper in specific solutions regarding the design of an architecture that ensures data-privacy and model-privacy. Any literature research is non-exhaustive and is mainly focused on finding one complete solution and realising an implementation. We describe how we built the architecture in section 5, while we test the resulting regression model quality against our baseline in section 6.Solutions for the optimisation of other models optimisation besides regression are discussed in section 7.

Chapter 2

Background

2.1 Personal Data Stores

In existing research, among 603 interviewed, 56% of researchers demand control over or knowledge in the usage of their personal data [21]. Personal Data Stores (PDS) are patient-managed platforms that store data to be used by institutions with the correct permissions. This stands in contrast to cloud storage that holds data meant for personal use. PDSs stimulate user-centred software solutions and the data is accessed via a personal API. When a PDS is meant for storing personal health data, the data in it is referred to as Personal Health Records (PHR). This stands in contrast to Electronic Health Records (EHR) which are governed by institutions themselves and patients have less control over collection and processing. The store connects to a personalised database where users or institutions can add information to, or the information is updated via pointers from an external EHR [52]. This PHR should not be confused with a personal portal, where all the data is a reference to the original data from an EHR and is not a platform providing ownership and governance.

While the advantage from the perspective of the user is clear, institutions do not widely adopt this method of recording data [27]. Also, data sharing through PHRs is protected by the european General Data Protection Regulation (GDPR) as being one of the most stringent data protection laws[22]. PDSs do not guarantee data governance as, in its current form, a copy of the data is still shared with organisations.

PDSs can be manifested in two ways. The patient has the data stored on its personal device, or companies can manage a patient's data storage (called Personal Online Data Storage, or PODS). The latter has access to the sensory information of the device, possibly with peripheral equipment such as a heart-rate monitor. A large difference is the accessibility of the data. While a server can be assumed to be always available, a personal device can be disconnected due to various reasons.

Currently, almost all data is centralised due to the value that such data collections have. Owners of such collections can apply numerous machine learning and analytics tasks. While PDSs are not widely adopted, a shift to a more secure environment for personal records could cause a drop in available data for

researchers and/or institutions. In this case, we would need a mechanism that can utilise the data without it leaving the device so that governance is maintained. On the other hand, if such a mechanism could be applied, this may set in motion a shift towards a society that governs its own medical data, and more privacy is maintained. The Personal Health Train is such an initiative to bridge this bipartite gap.

2.2 Personal Health Train

The Personal Health Train (PHT) can be considered a framework for applying analytics and machine learning tasks [3] over multiple data sources. The PHT was developed under the FAIR conditions. The **FAIR** principles act as a guideline for our data-rich society, where information should be Find-able, Accessible, Interoperable, and Reusable [57]. While FAIR data considers the communication with the data source, the PHT is concerned with connecting different information sources to automate accessing multiple data sources at once or in parallel. Numerous machine learning algorithms can be applied to centralised data. These algorithms however must be altered, as data is not stored centrally. When using the train metaphor, the PHT can be separated in three distinct components.

Trains contain the explicit code of the analytic or learning task. This action may be a simple query on a database. As an analytics example: how many patients have a blood level x higher than value y. But it may also return updated model parameters when learned over the data. A user of the PHT may send a request to a central PHT server. This server sends a request over to the data stores, which can, in turn, retrieve the right Train from storage and execute it over their data. The result is sent back to the central server and the aggregated result (summation or another specified aggregation method) is returned to the client that requested it. A client may be some authorised user within this PHT framework. Machine Learning trains can implement two different learning strategies [42]. (1) When applied in parallel, we implement a central aggregator Train that communicates with the data-rich side of the train at the source. (2) The task can also be executed sequentially, which it is then referred to as Weight Transfer [11] and Institutional Incremental Learning [48].

The **Rails** is a metaphor for the central server that can receive a Train directive from an authorised client, dispatch them over relevant data locations and aggregate results. It receives a reference train object and metadata. The latter could specify some required information, such as the model architecture, learning rate, data used, etc. It also deploys the server-side part of a Train locally, providing the correct aggregation tool.

The **Station** has access to a private data collection and can execute the algorithm of a Train. The Station trusts the Train, as it is located in a Train Registry. This active Train on the Station communicates with the part of the Train on the Rails (server).

The PHT is not only limited to Machine Learning or Analytics over multiple

data sources, it tries to containerise algorithms to reproduce or repeat specific results to promote usability. The PHT also implements privacy-by-design by letting the data owner control how its data can be shared, processed, or is not available at all. These settings may be applied for specific data or specific applications.

Over the past few years, some research is done on the application of analytics tasks within the PHT [15]. Choudhury et al. described a methodology to apply analytic tasks in a privacy-preserving manner [15]. Some advancements have been made on training ML models regarding multiple data sources, which will be described in chapter 2.4.

To this point, research has focused on testing the PHT on a very limited set of institutions with big data storage, and tasks from simple data analytics to deep learning have been applied. If we indeed tend towards separating our personal data from institutions, which generates more ownership and security, a decline in the application of data analytics and machine learning is unavoidable. This may cause a decline in the discovery of new medicine and valuable insights for treatment. The PHT is a platform for creating solutions to increase knowledge even after the data has been separated. Vice-versa, the PHT platform itself may also give users incentive to separate their personal health records in PDSs as the PHT guarantees user governance over the personal data. This research may also give incentive to institutions to use the PHT because of a larger collection of data that can be accessed for research purposes while decreasing the need to deal with privacy regulations.

2.3 Regression Algorithms

In the situation of isolated PDSs, research on this data must not be limited because it is separated. This thesis' main concern is limited to constructing two prominent algorithms to optimise linear regression and logistic regression models over PDSs.

Those algorithms can be classified as belonging to the statistical domain or the machine learning domain. Both areas overlap, thus regression can fit into both. When taking a statistical perspective, our algorithm applies a descriptive analysis over a dataset and minimises a certain loss function to represent the data optimally. But when the machine learning perspective is taken, the predictive analysis (i.e. performance) over an unobserved test set must be optimised. In our research, we take into account a possible train-test set distribution so that the model can be tested over unobserved data, but a classical inference model (goodness-of-fit) should not be excluded.

2.3.1 Linear Regression

A linear regression model models the relationship of a scalar output with one or more explanatory variables: $\hat{y}_i = \beta_0 + \mathbf{X}_i\boldsymbol{\beta}$. Linear regression is commonly optimised with the Ordinary Least Squares (OLS) estimator method $\boldsymbol{\beta} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$.

(Intercept β_0 can be calculated with the estimated coefficient $\beta_0 = \bar{y} - \bar{\mathbf{x}}\boldsymbol{\beta}$.) This method results in a vector of coefficients $\boldsymbol{\beta}$ that minimises the sum of the squares of the residuals/error $SSE = \sum_{i=1}^{n}(y_i - f(X_i, \boldsymbol{\beta}))^2$ [17]. In our case, the linear regression function is inserted for function $f()$.

Instead of using OLS to optimise the regression model, it is possible to apply a gradient descent (GD) method that also optimises the loss function. In this thesis, we regard optimising linear regression with gradient descent, as this may open up ways to optimise the regression model and preserve data-privacy, see section 2.4. Gradient descent calculates an update step by calculating the derivative of the loss function with respect to the coefficients. For the loss function, we take the sum of squared residuals and derive one coefficient with $\frac{\mathrm{d}}{\mathrm{d}\beta_i}\sum_{i=1}^{n}(y_i - f(X_i, \boldsymbol{\beta}))^2$. After deriving, we can take the data (X,y) and current model coefficients $\boldsymbol{\beta}$ to calculate the gradient update step $\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_{i+1} + \eta\nabla g(\boldsymbol{\beta})$ where $\nabla g(\boldsymbol{\beta})$ is the vector of derivatives over all coefficients. Updating the coefficients with the negated derivatives of the coefficients decreases the sum of squared residuals. This process can be repeated.

But do the methods of Ordinary Least Squares and Gradient Descent (GD) yield equivalent estimates of the coefficients? As they both optimise the same function, they should theoretically yield equal results. Where OLS always finds a solution, gradient descent may potentially take some iteration steps when data is skewed and the learning rate is sub-optimal. An advantage of GD over OLS is when it is applied over an exceedingly large dataset. With OLS all data is put into one computation, while the mini-batch variant of GD may create an update over a subset of the data.

2.3.2 Logistic Regression

The formula for logistic regression with multiple independent variables is stated in equation 2.1.

$$\hat{y}_i = \frac{e^{\beta_0 + \mathbf{X}_i\boldsymbol{\beta}}}{1 + e^{\beta_0 + \mathbf{X}_i\boldsymbol{\beta}}} \tag{2.1}$$

As can be seen, the linear equation is included but is transformed by applying the sigmoid function, or inverse-logit function, over the linear result. Instead of describing its activation function as is standard in machine learning, literature sometimes also adds a link function common in the field of statistics. It maps the nonlinear result to a linear combination of variables. In this case, it is the inverse of the activation function, i.e. the normal logit function. The prior formula can be rewritten as equation 2.2.

$$\text{logit}(\hat{Y}) = \ln(\hat{Y}/1 - \hat{Y}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_i X_i \tag{2.2}$$

Coefficients in logistic regression do not represent the relationship accurately, as it is not linear. To be able to represent the dependency correctly, we use

the odds ratio (OR). The odds, $odds(Y) = \frac{p(Y=1)}{p(Y=0)}$, is the ratio of an event Y occurring versus not occurring. The odds ratio represents the constant effect of an independent variable X on the odds of Y occurring, $OR = \frac{p(Y=1|X=1)}{p(Y=0|X=1)} / \frac{p(Y=1|X=0)}{p(Y=0|X=0)}$. In the previous formula, X is a categorical variable. For continuous variables, the OR must be interpreted as the change in odds when X is increased with one unit. In both scenario's, OR can be deduced from the logistic regression model beta-coefficients, $OR = e^{\beta_i}$. In logistic regression, OR is constant for all possible values or categories of a certain independent variable X, but in reality, the OR may be inconsistent within X.

While the OR values can be interesting to compare, they are commonly accompanied by the Confidence Interval (CI), indicating a range with a chance of 95% that the real OR of the population is in that range. The confidence interval is not included in a logistic regression model and thus we limit ourselves to only considering the OR. We use OR instead of β_i as it has better interpretability. Future research on federated statistics could help fill this gap.

Logistic regression is mainly optimised via Maximum Likelihood Estimator (MLE) functions. MLE are the group of methods that estimate the most probable model parameters. Within logistic regression, the MLE always minimises the binary cross-entropy (i.e. log-loss). As there is no closed-form expression like linear regression which optimises via OLS, MLE is an iterative process. For our experiments, we will be using the quasi-Newton Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm for optimising logistic regression.

$$Loss = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i) \tag{2.3}$$

By inserting the logistic regression formula in the cross-entropy loss function, equation 2.3, we can take the derivative similar to linear regression to calculate an update step and apply gradient descent (GD). As both MLE and GD methods optimise the same loss function, they also theoretically converge to the same regression model.

To create linear and logistic regression variants in the PHT, it may be able to use techniques from the field of federated learning or decentralised learning.

2.4 Federated Learning

Rules, regulations and ethical aspects could prevent us from creating a centralised data storage and thus no machine learning can be applied. Some solutions within the field of federated learning (FL) are proposed that are suited as an applicable algorithm within the PHT and could provide us with the same insights as normal machine learning. We elicit their workings and global characteristics.

In 2015 and 2016, researchers coined the term Federated Learning (FL) as a way to distribute learning over separate devices [30, 31]. Such an FL architecture provides potential privacy, security, governance and economic benefits,

but also brings four new challenges [34], such as the impact of non-identical distributions of data [61], the communication bottleneck of FL implementations [5], devices having different capacities and capabilities [54], and possible information leakage from the devices [38].

The naming scheme can be somewhat confusing, as names are similar and misinterpretations are easily made. Figure 2.1 provides a graphical overview of the fields and subfields. Machine Learning is the overarching field of algorithms that implements self-optimisation based on certain goals or data. The biggest and most straightforward field is Centralised Machine Learning, where data is located on a central server. Decentralised Machine Learning was introduced as the computation power lacked behind the amounts of data that are collected nowadays. Under one single orchestrator and using multiple local devices as computation points, algorithms can use more calculation power to learn models. Available data is then distributed over those devices. Federated Machine Learning, or Federated Learning, is mostly applied to situations where computing power is limited, the number of data sources is very large, and privacy restrictions are present.

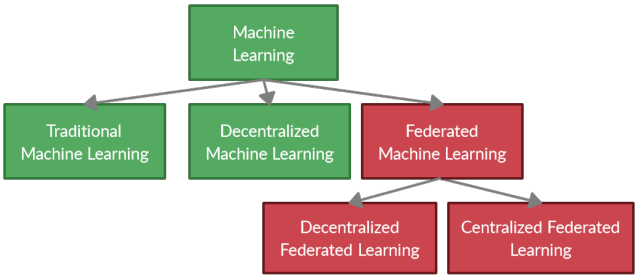


Figure 2.1: Hierarchical structure of terminology

Federated Decentralised Learning and Federated Centralised Learning differ in communication, as there is no central orchestrator needed within this field. Here the "Decentralised" part also refers to the learning algorithm, not only to the scattering of data. The PHT uses a central handler to dispatch algorithms or aggregate results, making it able to apply Federated Centralised Learning.

When one refers to FL, Horizontal FL is the most common technique. Vertical FL deviates from this by splitting the data on the feature axis instead of the sample axis. A concept was created that implemented Vertical FL in a PHT architecture [55]. Personal Data Stores however split the data, as the name suggests, on the sample axis. Some horizontal learning problems have been solved by applying horizontal FL to the PHT. A practical implementation was made to apply deep learning in a PHT infrastructure [42]. Two research papers regarding the analysis of data with a logistic regression model within the PHT were published in 2018 and 2019 [16, 49].

For the specific case of PDSs, and when looking at the above explanation of our terminology, we need to research the domain of Centralised Federated Learning. As (1) the PHT revolves around a central handler having access to the final result. (2) PDSs can be unavailable at some point in time. Research on the FL technique regards the PDSs as stateless, so that the algorithm still functions when there is a PDS dropout or other malfunction. (3) Computing power is very limited on edge devices, generating the need for a lightweight solution. (4) Data originates from different patients, making the data non-i.i.d. (explained in

the next section).

FL has an impact on all algorithms that can be optimised via gradient descent. This includes Neural Networks, Support Vector Machines, but also linear regression and logistic regression.

2.4.1 Independent and Identically Distributed

When choosing our solution, FL and DL techniques are clearly separated by some characteristics like the number of computing sources and data-privacy but also whether the data is independent and identically distributed (i.i.d.) [28].

Variables are mutually independent and identically distributed when they are drawn from the same probability distribution. This i.i.d.-ness refers to different samples of the same variable. In this thesis, a sample can be regarded as a measurement of multiple variables, including only one measurement per variable. The measurement of a patient's blood levels, will not influence the measurements of another patient. However, when the patient's measurements are taken again at an earlier or later moment in time, the outcome will be heavily dependent on the previous sample. In Federated Learning, non-i.i.d. variables are a significant problem [26].

One PDS will consist of data from a single patient. Multiple observations of patient features will have a high correlation. For us, this means that combining multiple patients and multiple observations per patient results in a non-i.i.d. dataset.

The fact is that there is only one observation per variable in a PDS. This is making the statement that "different observations are non-i.i.d." useless, as there are not multiple observations. When combining all the data from different PDSs, the data is i.i.d., as one sample does not give us any information over one of the other samples. When multiple measurements of one patient were to be included, this statement would be false.

Federated Learning focuses on optimisation over non-i.i.d. data, while decentralised learning assumes i.i.d data as input. While assuming decentralised learning would optimise the problem faster, it does not take into account our requirements: guaranteeing privacy and using numerous (possibly mobile) sources. Therefore we only explore methods that are labelled as Federated Learning.

2.5 Threat Model

A Threat Model tries to expose possible vulnerabilities in the system under certain limitations or indicates which threats the author may try to solve. As we are sending information through a network with different kinds of parties involved and our goal is to hide certain information from certain parties, it is good to analyse possible internal threats.

The PHT assumes the complete network to be trusted when dealing with institutions, however, Personal Data Stores owned by individuals can also be manipulated by patients themselves, which gives rise to a more prominent security

threat. Inaccurate data may influence the performance of the data significantly if the data were designed by an adversary, or access to the model may leak personal data contained within it. For this research, we ignore the possibility of data poisoning and the retention of individual's data in the final model, which we further elaborate in section 2.7.

We consider two types of deviating users. *Honest-but-curious* adversaries follow the protocol but receive a transcript of all the actions and values observed by the protocol. It cannot change communication contents by actively influencing the algorithm, which we would call a *Malicious* adversary. We can alleviate the privacy requirements by assuming the architecture interacts with honest-but-curious parties.

By having only honest-but-curious adversaries in the network, we can eliminate an extra trusted party that can verify parties. For example, without a verified network, the server could fake being a participating data locker and share critical cryptographic information (for communication) to extract privacy-sensitive updates from messages. For more information on cryptographic methods or to understand the example on why two patient parties would like to communicate, see the next section (2.6).

We do not imply that a malicious adversary is unrealistic, but we limit our research by assuming only honest-but-curious parties. We then arrive at the next threat model for our research:

- The central server may be an honest-but-curious adversary (potentially combining information from other sources)
- User devices may also be honest-but-curious adversaries (potentially combining information from other sources).

2.6 Cryptographic Tools

Cryptographic tools cannot be omitted when regarding federated learning, as FL itself does not guarantee any privacy per sé [4, 39]. The following sections explain two commonly used cryptographic tools important for private federated learning.

2.6.1 Homomorphic Encryption

Features can be homomorphically encrypted (HE), making them unreadable for parties without the corresponding private key. The advantage of Homomorphic Encryption is that the encrypted object is mathematically manipulable, i.e. summation and/or multiplication can be applied, changing the encrypted object so that when it is decrypted, the operations are immediately applied to the original object. In practice, this means that one party can manipulate an object, adding information without seeing the original information and without seeing the resulting information. In figure 2.6.1 the server S can manipulate an encrypted

value, while party C is the only party to observe the result of the manipulation.

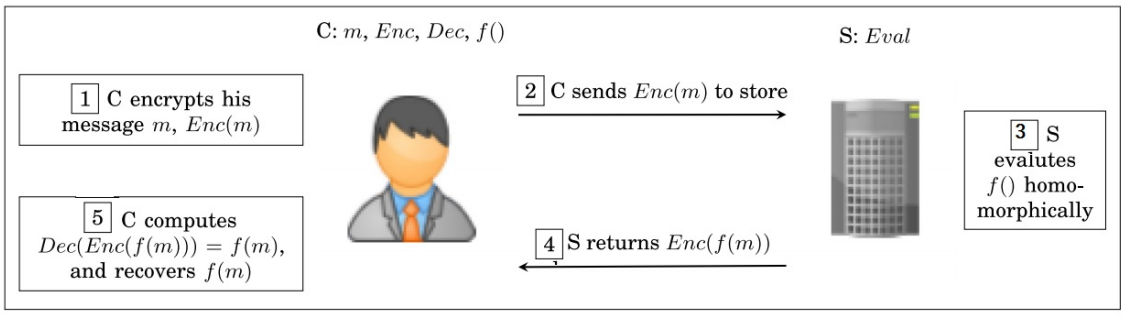


Figure 2.2: A simple client-server HE scenario, where C is Client and S is Server. [1]

Partially Homomorphic Encryption protocols are Additive HE and Multiplicative HE. Examples of additive HE are $E(x) + E(y) \rightarrow E(x + y)$ or $E(x) + y \rightarrow E(x + y)$. Examples of multiplicative HE are $E(x) * E(y) \rightarrow E(xy)$ or $E(x) * y \rightarrow E(xy)$. Fully Homomorphic Encryption methods can apply both (encrypted and decrypted) sums and products to an encrypted value. The use case for HE in federated learning is the central addition of information from numerous sources, while encrypted for the central aggregator.

An Additive HE algorithm can be described as a tuple of four items:

$$HE = (KeyGen, Enc, Eval, Dec)$$

- KeyGen: generates two distinct keys, private and public key, for asymmetric encryption. These keys are meant to be stored locally or distributed to trusted parties that must access encrypted information.
- Enc: can generate ciphertext with an encryption method when concerning plaintext and a public key.
- Eval: ciphertext can be altered in an encrypted state (sum, product).
- Dec: can recover the plaintext when applying a decryption method concerning ciphertext and the correct private key.

HE in Federated Learning

In practice, HE occurs in two forms, either (1) encrypting the model when sending it to the collaborators or (2) encrypting the data when sending it to the central handler, where all the encrypted values can be aggregated (summed). To ensure data-privacy, the latter is the most common technique. We discuss why naively applying HE in federated learning is not advised.

The PHT requires the final model to be accessible by the central handler. In the case of using HE to encrypt the raw data or the updates before aggregation,

the central handler will not have access to any model or will only have access to an encrypted version of the model. This means the final model must be decoded, and a fully trusted third party is required to do this as the central handler may not have the decoding key. Otherwise it is able to encode the raw data from the PDSs. In federated learning it is standard to not assume a trusted third party.

Most HE schemes have relatively high computation-time, but this is negligible due to the small shallow models we are training. We are dealing with a huge number of collaborators, which means there are many exposed encrypted values. The secret key should be renewed every round, as it will be prone to cyphertext attacks [12]. This makes the algorithm heavier from a the computational and communication point-of-view.

2.6.2 Diffie-Hellman Key Agreement for Data Aggregation

In a typical federated learning setting, the server receives the local updates and calculates the global update for the model. Local updates contain privacy-sensitive information and should not be leaked to the central handler. The Diffie-Hellman (DH) Key Agreement [18] is a method to share a secret through public communication. By sharing a secret through means of a Diffie-hellman key agreement, it is possible to hide individual updates while keeping access to the global aggregate.

As an example in figure 2.3, Alice and Bob are making a key agreement. Both parties can be considered two different PDSs where the central handler is omitted for simplification. Both parties agree to use numerical values g and p by communicating them over the network. Both Alice and Bob possess a secret key a and b respectively. By sending Bob $g^a \mod p$, Alice's secret a cannot be inferred easily. The same hold for Bob when sending $g^b \mod p$. Both values are communicated and are processed further by an exponentiation with its local secret key, $B^a \mod p$ and $A^b \mod p$ both equalling the same secret "AB". "AB" is not a multiplication but just the name for the shared key between Alice (A) and Bob (B).

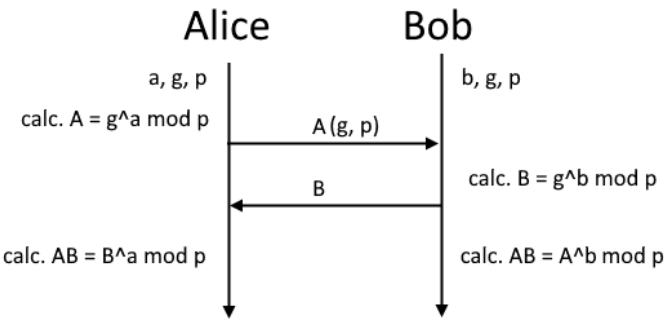


Figure 2.3: protocol flow of Diffie-Hellman key agreement

DH Key Agreement in Federated Learning

With this secret, both parties are able to generate a mask or negated mask by using the secret as a seed in a pseudo-random generator. When all those masks

are added to the local updates and sent over to the central server, the server is not able to see the individual updates. By summing the updates, the masks will cancel out and the aggregate (sum) of all the updates is shown to the central server, without showing individual updates. See figure 2.4 for a graphical representation of the method.

In our isolated PDS context, the exchange between all pairwise users may be time-consuming. The main second concern of such an aggregation method is that when one user drops out of the aggregation, the aggregate becomes unusable, as the masks do not cancel out anymore.

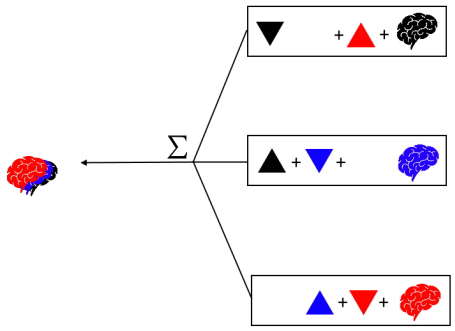


Figure 2.4: Protocol flow of aggregation with masks

2.6.3 Dropout-Robust Secure Aggregation

Secure Aggregation [6] was introduced specifically for application on Federated Learning for wireless devices, extremely big numbers of clients, to mitigate the risk of exposing private data and catch possible drop-outs. In the Personal Health Train setting regarding PDSs, this is a realistic scenario.

The previous method included masks so that it is only possible to aggregate when all clients are actively participating. To make a dropout robust protocol, Bonawitz et al. [6] used a method called **threshold secret sharing**, or Shamir t-out-of-n Secret Sharing. It is a protocol to split a secret into its shares, with the possibility to recover the secret by only obtaining t of the n shares, see figure 2.5. A polynomial can only be reconstructed when enough datapoints (shares) are recovered. The original secret is the value of a polynomial where $x = 0$. In federated learning, we split the secret key of a party into shares, but not the shared keys, nor the masks. That means that every party holds one share for every data party in the network.

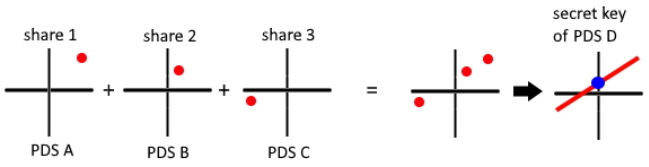


Figure 2.5: Rough visual overview of Threshold Secret Sharing protocol.

The algorithm can be seen as an expansion of the update masking from previous section 2.6.2. When a user D drops out, the mask of the dropped user must be recovered. To recover the mask, all the shares concerning the single secret key of D must be requested from all other parties (fig 2.5). Then the secret key of D can be calculated. The central handler can use this secret key of D to recover all the shared keys with other parties, as it can rerun part of the DH-key agreement. All the shared keys can in turn be turned into masks as we explained in the previous section. The central handler adds or subtracts the

masks from the final aggregated result to make it usable.

One problem with this technique is that we must assume an honest-but-curious server, as the server could fake that a user has dropped out to retrieve its mask and to recover its original update. If the server is indeed malicious, extra measures can be taken, as described in the paper of Bonawitz et al. [6]. As we do not assume a malicious server, any extra measures may be omitted. A good estimation on the risk of drop-outs during learning and possible processing bottlenecks would help to choose the best aggregation algorithm in a scenario. This is heavily dependent on the stability and type of the network. We also did not define whether PDSs are mobile or located on a wired server.

2.7 Differential Privacy

Besides the application of cryptographic methods, there is an alternative method to ensure dataset privacy. Differential Privacy (DP) is a measure of ensuring that individual data cannot be leaked. The DP originally refers to a situation where database information is leaked from a publication of a statistic, model, query or another medium. DP guarantees that no single dataset sample can be identified when such information is disclosed. In federated learning, this comes down to the information that the individual updates or the final model give us. An inference attack is an example of the extraction of samples from the original model, which DP tries to prevent. Differential Privacy is commonly achieved through the addition of noise.

In Federated Learning, DP refers to the prohibition of data leakage via two ways: model updates, and model parameters. Global Differential Privacy (GDP) only adds noise centrally after aggregation to prevent inference attacks. But there is a second way. Local Differential Privacy (LDP) adds noise to individual updates, masking the individual updates (just like the cryptography methods) to the central handler and also decreasing the risk of inference attacks on the final model.

We argue that our situation is different from most papers on DP applied to federated learning. Research has confirmed that the effectiveness of inference attacks is closely related to the complexity of the machine learning algorithm [50, 47]. We are training shallow regression models, which can be seen as one of the simplest neural network models. Using a shallow regression model is not a guarantee for user privacy. Having multiple independent variables when only including a few samples can still be a cause for over-fitting, but the number of samples has to be extremely small. If the number of samples is high enough, we hypothesise that no inference attacks will be effective. For simplification of this research and due to time-constraints, we do not tackle the problem of whether or not the regression models memorise samples. Therefore we also do not implement any solution to this problem.

We hypothesise that the risk of inference attacks on our model is small, which would deem GDP less useful. Local Differential Privacy can be applied to mask updates, but in our situation as there is only one patient involved per

PDS (instead of 100's or 1000's), we assume it would need such a high amount of local noise that the update becomes unusable. We conclude that applying both LDP and GDP comes with its drawbacks and can even damage the model quality. For this reason, we only use cryptographic methods for ensuring data-privacy. It is up to future research to determine whether differential privacy can replace cryptography. For now there is no theoretical basis to use DP.

Chapter 3

Federated Standardisation Protocol

Our research question tries to compare a federated learning architecture with its classical centrally optimised counterpart. To get the best performance from a federated learning architecture we explore the possibilities of standardisation of the data. As the data is distributed over different locations, we analyse the consequences of distributed data on standardisation and a possible solution.

Gradient descent methods are highly sensitive to feature scaling [56]. So why do we not just standardise or normalise our personal data stores? For standardisation, we need the sample mean $\bar{x}$ and sample standard deviation s for every input feature, and x_{min} and x_{max} for normalisation. Such a calculation is done centrally as all feature values must be taken into account. Our federated situation does not fulfil these requirements. We explore relevant literature and propose a different solution to improve the quality of our federated models on real-life datasets.

With this challenge, we also dive deeper into the realm of security and encryption. Some information has to be exposed somewhere, and for optimal security, the proposed mechanics have to be analysed. Both methods described in the next paragraphs should not influence the performance results, however, it is important to evaluate its security concerns. Only federated standardisation protocols are proposed, as we have not come across any method that can normalise data in a privacy-preserving way, nor have we created a method that is capable of doing this.

3.1 How Standardisation Influences Coefficients

As was just mentioned, standardisation of the continuous and categorical input and/or output variables influences the convergence rate of gradient descent regression methods [56]. While this is advantageous, models with standardised (beta) coefficients b^* must be interpreted differently than standardised (non-beta) coefficients b, but can be useful in different ways.

For logistic regression holds: “a one-unit difference in X is associated with a b-unit difference in $logit(Y)$" and “a one standard deviation difference in X is associated with a b^*-unit difference in $logit(Y)$". The predictor variable Y is not meant to be standardised.

For linear regression holds: "a one-unit difference in X is associated with a b-unit difference in Y" and when the input is standardised: "a one standard deviation difference in X is associated with a b-unit difference in Y". And a variant when also the output is standardised: "a one standard deviation difference in X is associated with a b^*-standard deviation difference in Y". It is good habit in data analytics to standardise all your data.

It is possible to convert a linear and logistic regression model with beta-coefficients to normal coefficients and vice-versa [40].

3.2 *Secure Scaling Operation* (by Mandal et al.) [36]

The Secure Scaling Operation (SSO) was introduced by Mandal et al. [36]. It uses Secure Aggregation and only one communication step to be able to calculate the mean and standard deviation. It does so in a privacy-preserving way. In our scenario, there is only one sample per PDS. Because of this, we explain the SSO in the context of isolated PDSs.

Here we summarize the original SSO. For every input feature, the PDS must return two values, i.e. it must calculate $2n$ dimensional values from n features. Due to the fact that sample size per PDS equals 1, the calculating formula is simplified to: $X^{(i)} = (x_1, ..., x_n, x_1^2, ..., x_n^2)$ where $\boldsymbol{x} = (x_1, ..., x_n)$, which is the original PDS data. Every PDS in the set of U calculates this vector. The central aggregator uses the proposed dropout-enabled secure aggregation protocol (from section 2.6.3), which we call π_{DEA}, to calculate the aggregate-sum of those values X along with a list of responding PDSs (U_a). $(U_a, X) \leftarrow \pi_{DEA}(U, X^{(i)}, |U|, t)$ which includes $|U| > t$ and where $X = (X_1, ..., X_n, X_{n+1}, ..., X_{2n})$ and t is the threshold for the number of PDSs that need to be used. Otherwise, the aggregation fails as the dropouts cannot be recovered from. The mean $\boldsymbol{\mu} = (\mu_1, ..., \mu_n)$ and standard deviation $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_n)$ can be calculated with respectively $\mu_p = \frac{X_p}{|U_a|}$ and $\sigma_p = \sqrt{\frac{1}{|U_a|}(X_{n+p} - (2|U_a| - 1)\mu_p^2)}$. The final step includes sending $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ to all users in U_a so to scale their data with $x_{standard} \leftarrow \frac{x - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$.

Participants may be honest-but-curious adversaries and this statistic sharing may be regarded as a model-privacy leak. With Homomorphic Encryption, a second communication round is able to prevent this. The calculation is done on the central handler on the homomorphically encrypted user data, sent from the PDS. $HE(x_{standard}) \leftarrow \frac{HE(x) - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$.

Because the sample size per PDS equals 1 in our scenario, the mean and standard deviation cannot be precisely estimated from the data and standardised data alone. Two unknown variables in in equation has infinite solutions. This means an adversary is unable to discover the original statistics based on its own data. It is a one-time calculation that does not increase the complexity of the complete protocol, it also clearly decreases privacy concerns compared to previous method and thus we argue that it is still beneficial to use this protocol.

However, the threat model includes a scenario where multiple PDSs are controlled by the same adversary. For this method, it is possible for the sample

mean and standard deviation to be discovered when any two PDSs are honest-but-curious adversaries. By having two samples of the input (raw data) and two samples of the output (standardised data), it is possible to retrieve both unknown variables μ and σ. This architecture is therefore 1-private, meaning that it is secure when only one PDS is controlled by an adversary. But when more than one PDS is controlled, the mean and standard deviation can be estimated by the adversary.

3.2.1 SSO 2.0

Preliminary results regarding the calculation of statistics in a federated way found that the standard deviation deviated from any centralised statistic by an approximate margin of 10-20%, meaning the given formula by Mandal et al. [36] is incorrect. By altering the given formula of σ_p to $\sigma'_p = \sqrt{\frac{1}{|U_a|}(X_{n+p} - |U_a|\mu_p^2)}$, various testing concluded that the decimal place accuracy is at least 7, and the difference is percentually negligible.

Chapter 4

Privacy-Preserving Protocols

We dive into aggregation methods that can potentially apply linear regression and/or logistic regression in a federated manner. This chapter provides solutions on how to learning in a federated fashion and provides solutions to preserve data-privacy and model-privacy, which answers RQ1. We can leverage this knowledge to create an implementation to answer RQ3, which we propose in section 5. Federated Learning methods alone do not guarantee any privacy, so this chapter dives into FL algorithms but also into other necessities for the creation of a privacy-preserving protocol.

Some papers describe learning a regression model by the sharing of data that is homomorphically encrypted (HE) [43, 2, 7], or by using Yao's garbled circuits in its protocols[23, 43]. HE can have some disadvantages in our scenario. The resulting model is encrypted and must be decrypted with one of the keys of the collaborative parties (PDSs). The central handler does not have access to the model coefficients/parameters [29] and possible sharing of the private key from the PDSs may increase chances of data leakage. Therefore we exclude discussing HE in its simple form. We also exclude any implementation regarding Yao's garbled circuits due to time-constraints.

4.1 Aggregate Methods

Most recently proposed federated learning algorithms are all based on the communication of gradients or consequent model parameters.

Alternative Direction Method of Multipliers is a method introduced by Boyd et al. [9] before the Federated Learning era but can be viewed as a FL protocol. Every iterative local calculation is based on residual compression errors from previous rounds [9], meaning that a dropout of a node halts the training process. Such a 'stateful' algorithm is impractical when potentially training over thousands of PDSs. Although the long existence of ADMM, FedAvg and FedSGD are more common in literature, including many work on the shoulders of those giants [28].

Federated Averaging (FedAvg) is the most basic form of FL and diverges the least from a centralised approach. Every location updates a model like

it was centralised, only using the initial model parameters received from the central handler. After learning, it communicates the new model weights to the central server, because the local model has been updated multiple iterations with different gradients. When clients end multiple iterations of local updates, the updated model parameters are then sent back to the central server, where they are averaged per weight. This new model is then used to validate and to start a new training cycle. The model is not guaranteed to converge as the local models potentially update local parameters differently when the data is non-i.i.d., which we discussed in the previous chapter.

Federated Stochastic Gradient Descent (FedSGD) is a Federated Learning algorithm [44]. It calculates the gradient of the loss function of the model on each separated instance. As the loss function is to be minimised, updating the parameters with the negated value of their corresponding gradient will decrease the loss function. A training cycle is computationally-light, as only one update step is calculated, but is communication-heavy, as each gradient descent step is to be communicated.

In contrast to FedSGD, FedAvg is known to have fewer communication rounds due to increased local training and that does not have to diminish the performance [37]. With FedAvg, we can increase the number of iterations of the local PDS update step, but as there is only one sample per local PDS, we argue that this is less effective. Iterating over the same sample does not add as much information as an iteration over a new sample. Optimising via ADMM seems especially impractical in our use case, as a communication error or changing access settings can cause an unrecoverable dropout. So FedSGD seems the optimal choice for our architecture.

4.2 Federated Learning Protocols

FL protocols define a complete architecture, while the prior aggregate methods only define the way of combining information from different data sources. Because of extra privacy issues that arise when regarding PDSs, a more extensive, complete protocol regarding Federated Learning is required. According to our threat model, we have to regard two main goals.

First, the protocol should **preserve user-privacy**. From the above described federated learning algorithm, two problems arise. (1) The gradient is visible to the central handler, which is a potential privacy leak, as the gradient may potentially leak some data that it was derived from. Robust-Dropout Secure Aggregation [6] was introduced to mitigate the risk of exposing separate gradients or model parameters to the central server. A more extensive protocol is *PrivFL* [36] and incorporates secure aggregation in a bigger protocol to hide individual gradients. Another protocol regarding this type of privacy is *VANE*. (2) The second risk that arises from this assumption is the risk of information leakage from the model that is available to the central handler. After hiding the individual gradients to the central handler, an incomplete or final model may still contain, thus leak, private information (inference attack). A common

technique used to prevent this leakage is the application of Differential Privacy (DP) [37]. For this thesis, we do not prioritise securing against inference attacks (see section 2.7).

Secondly, the model should have **model privacy-preserving properties**. Model-privacy is the act of hiding all model details from the data owners. With many federated learning papers focusing on data-privacy, there has been little focus on the privacy preservation of model parameters. With the standard application of FedSGD, the model parameters must be sent to the PDSs, thus no model-privacy is taken into account. *PrivFL* by Mandal et al. [36] uses HE to hide the model and its parameters from the user (regression models only). *VANE* does not include the model as a privacy-sensitive object and thus leaks the model parameters to the PDSs.

4.2.1 *PrivFL* (by Mandal et al.) [36]

PrivFL [36]is a solution for securing both the user input from the server and the model parameters from the users. It uses both Homomorphic Encryption and Secure Aggregation in its protocol. This protocol stands in contrast to only using an aggregation protocol, such as *Dropout-Robust Secure Aggregation* (section 2.6.2) or *DH-key Agreement with Data Aggregation* (section 2.6.3). Those methods are less complex but do not secure the model parameters against leaking the model details to a PDS adversary.

Figure 4.1 shows a high-level communication flow of the protocol. The model parameters are homomorphically encrypted by the server $E(\theta)$, with the server being the only party to have access to the private key, and are sent to all concerning data parties. For a PDS, the loss function is calculated over the encrypted model parameters $E(\theta)$ and the local data D_1. $E(loss) \leftarrow f_{loss}(E(\theta), D_1)$. The loss function has to be altered to deal with encrypted values as an input and is further described in the paper from Mandal et al.[36]. The application of logistic regression comes with an estimation of the sigmoid function to be applicable to encrypted values. The loss is used to calculate the encrypted gradient $E(\omega_i)$ for all weights. Random noise r array is generated and added to the gradient array to achieve s, resulting in $E(\omega_i) = E(s_i) - r_i$ These two shares are both sent to the central server, one in its original encrypted form $E(s_i)$ and one by applying the secure aggregation protocol π_{DEA} (from section 2.6.3) on r.

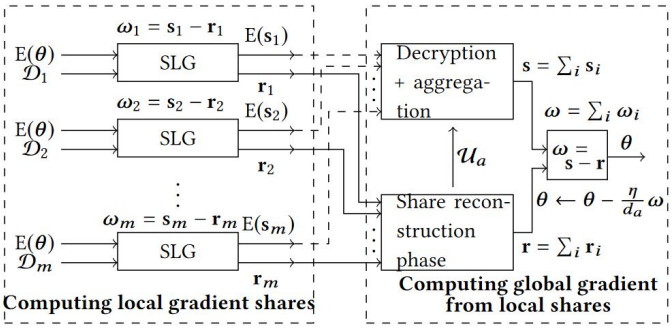


Figure 4.1: PrivFL's protocol flow of double privacy-preserving weight update for regression models [36].

The gradients cannot be extracted from the encrypted shares s_i without hav-

ing the corresponding random noise shares r_i. To access the aggregate of the random noise shares, Secure Aggregation requires at least a certain amount of random noise shares to make them accessible, as explained in section 2.6.3. If so, the shares s_i are decrypted and added, the corresponding random noise shares r_i are decrypted and added, and the total gradient is calculated with $\boldsymbol{\omega} = \boldsymbol{s} - \boldsymbol{r}$, where the individual gradients $\boldsymbol{\omega}_i$ are never seen. The final steps include dividing by the number of users involved d_a and applying a server-specified learning rate η. Because only the aggregate of the noise shares r is accessible, it is impossible to use a single noise share to subtract it from the share s to calculate a single gradient, which means it is unable to leak any user data to the server.

Chapter 5

Methods

Below we present the methods to test our research hypothesis. By not building a simulation but a real distributed architecture, it can be run on different devices. It is then also possible to get insight into the computation-time. The two federated learning architectures FedLinReg and FedLogReg, attempt to solve linear and logistic regression respectively when patient data is isolated with a guarantee of data-privacy.

We were not able to run any tests on a real distributed network which was the original idea for the design. This design can still be simulated on a single device and thus requires more computational resources than is necessary for edge devices in a real distributed setting.

The repository of an implementation of this architecture can be found here[1] here[2] and here[3].

5.1 Learning Algorithm

Federated Learning algorithms decrease privacy concerns but do not eliminate them altogether. We will be using one of the FL algorithms as research on the subject regarding privacy has delivered promising results. FedSGD is used to optimise our regression problems.

The name FedSGD is used throughout the thesis, however, a better name would be FedBGD, Federated Batch Gradient Descent, as our implementation does not select a subset of PDSs but rather uses all of them. FedBGD, however, is an unused term in literature and we will be referring to the optimization method with FedSGD. But why do we not need stochastic gradient descent?

Stochastic updating is used when there is a risk of optimising to a local error minimum instead of the global best error minimum. Any local minimum in a convex function is also a global minimum [8]. If we would minimise the error when the error function is a convex function, that would mean that it always converges to the global minimum which is the optimal result. The least-square

[1] https://github.com/CaspervanAarle/PHT_Node
[2] https://github.com/CaspervanAarle/PHT_Server
[3] https://github.com/CaspervanAarle/PHT_Data

loss function for linear regression is a convex function [8]. Idem for logistic regression where the cross-entropy function is convex [8, 46]. This means that no stochastic updating is required.

Every convex regression problem has an optimal batch size (selection of PDS instances for one update), where a larger batch size increases the computation-time but has higher gradient accuracy, while a smaller batch size causes a fast iterations but more iterations are necessary for convergence (slow convergence rate). Because we decentralise most of the calculations done, the computation-time will be less of a bottleneck. It is up to future research to find out at which number of PDSs stochastic updating becomes necessary. For this research, we include the maximum number of PDS instances possible, 100 or 160, and all samples in those PDS instances are used in every update step.

5.2 Cryptography

While this thesis also discusses model-privacy methods, we omit any implementation due to time constraints. An algorithm such as PrivFL uses extra encryption methods, which may be computationally expensive. While performance should be equal, computation-time may differ significantly.

Our first implementation was tested in the preliminary experiments in chapter 6.1. It omits any secure aggregation protocol to safely share gradients with the central handler, i.e. individual PDS updates can be seen by the central handler. We did this due to the fact that implementing it should not influence performance in any way. Later this was added to test efficiency in section 6.2.7. We use the method from section 2.6.2, which we call Simple Secure Aggregation

5.3 Hardware & Tools

We use Python as our main programming language. We tried to use **Tensorflow** (TF) as a machine learning library to implement both our classifiers. By implementing linear and logistic regression in Tensorflow, we were able to extract the gradient over all the variables with its built-in tools. In practice, however, running an instance of the TF library caused a portion of the working memory to be reserved. With our current system setup (16GB RAM), we could run a maximum of 27 PDSs all running an instance of the TF library. We overcame this limitation by manual calculation of the gradient update step by taking into account corresponding formula and derivatives of linear and logistic regression without using the Tensorflow library, resulting in a maximum limit of 100 PDSs. During our research, an upgraded system with 32GB RAM was able to run 160 PDSs.

5.4 Communication

For communication within the local experiment, we use **Python Sockets**. This communication standard can communicate within the local network via router ports. The experiments will not be distributed but instead, all parts are run on one device, except for when run-time is measured (see section 6.2.7). The experiments are run in separate Python instances to simulate different PDSs. This makes the architecture easily transferable to a real decentralised setting.

The architecture is only an implementation of the Train part of the PHT, as a broader implementation does not regard the research question nor influence the resulting models. A broader implementation of the PHT architecture would include containerising Trains, implement authentication and encrypt all communication. We omit this, but the implementation can be regarded as the deployment of two kinds of trains. The Aggregator Train must run once and the PDS Learning Train can be deployed on every data device. A configuration file can be sent alongside the train to define its task in detail.

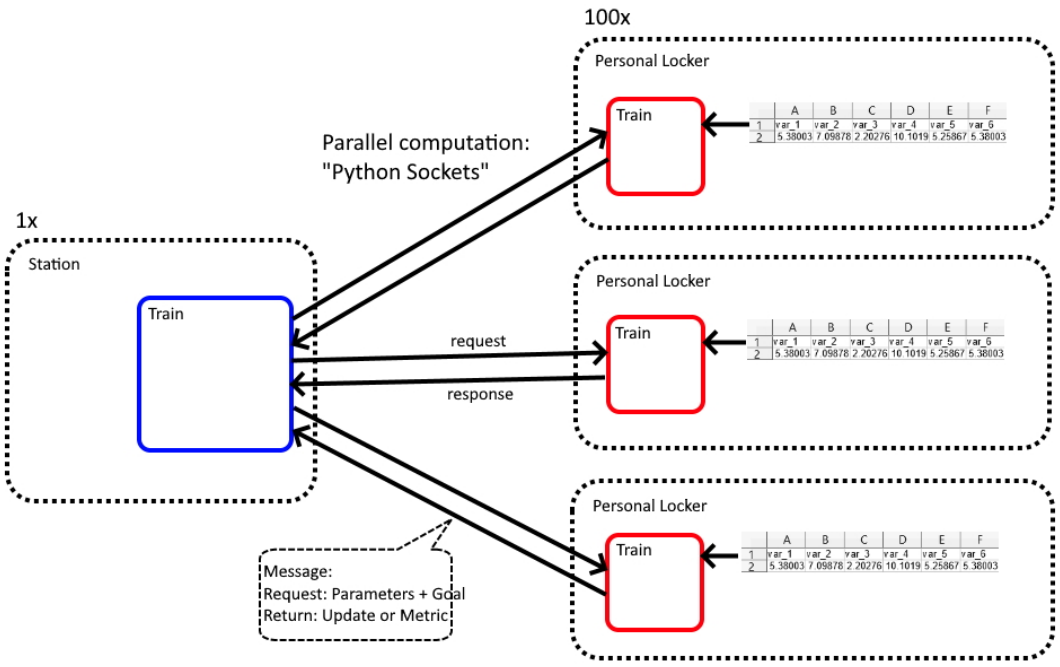


Figure 5.1: High-level structure of the design

Figure 5.1 shows a rough graphical overview of the communication architecture and the involved parts. The overview is simplified, as it only involves three PDSs. Communication is synchronous, i.e. updating is done whenever an update is received, instead of waiting for all other PDSs to finish. Different request types are discussed in section 5.5

We do not assume the existence of a trusted third party. That means that the sharing of secrets between PDSs is done through the network in figure 5.1.

5.5 PDS Request Types

As there is very limited data per PDS, the collection of PDSs must be split into a training set and a test set (instead of the data within the PDSs). While our first implementation included some limited options to create such sets, the final version omits this. For simplification of our experiments, models are tested centrally on a separate test set. As we only could run 100 or 160 PDSs, removing any test PDSs makes the number of training PDSs larger. Advantages are that the model becomes more robust as the size of the training set is larger, and there are various options to analyse a model centrally for our experiments. Otherwise, we would need to create extra functionality in our architecture. While for our experiments it is possible to test centrally, in practice this is not possible as there is no central database.

Preliminary implementation included some analytics requests, such as Accuracy or Loss, but they are not necessary to answer our research questions and thus were omitted in the final version. This would require extra analysis on possible privacy issues.

In general, there are three types of requests that can be made by the server:

- Secret Sharing: PDS can send/receive a key from/to a different PDS to generate a shared key for Simple Secure Aggregation.
- Gradient Calculation: PDS receives model parameters and returns a gradient descent step (with or without Simple Secure Aggregation)
- Encrypted Data Request: PDS can send/receive encrypted data to/from the server, where it is standardised.

Our implementation of the PDSs is a computation-loop, but it would be better to communicate via HTTP-requests.

5.6 Preliminary Architecture Optimisation

5.6.1 Learning Rate

The learning rate is the most common hyper-parameter that has to be tuned. Normal gradient descent uses a constant learning rate. This however may not be an optimal solution as the converging to the optimum is slower compared to more complex methods [25]. In our scenario, we have an incentive to decrease the communication required for training, as with every communication step, we still leak some information from the PDS. This hyper-parameter is the only parameter we optimise. Preliminary results showed that convergence to the optimum when using FedSGD can still take numerous iterations. To decrease communications costs and simultaneously increase the performance of the model, we propose to change the static learning rate η to an adaptive learning rate.

AdaGrad [20], changes the learning rate depending on previous updates of the weights. AdaGrad also considers weight updates separately from eachother, i.e. each weight has a separate learning rate. If we have large previous updates, that influences the next learning rate value to be high, while smaller updates cause the learning rate to decrease.

As a result of AdaGrad, the initial learning rate chosen becomes less relevant, though it may still have a big impact on more extreme η values. This in turn potentially decreases the time spent on optimising the learning rate, in turn decreasing communication costs. It is argued that a good optimiser for a convex optimisation problem is AdaGrad [25], see equation 5.1 and 5.2.

$$w_i^{t+1} = w_i^t - \eta B g_{t,i} \tag{5.1}$$

$$B = \frac{1}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} \tag{5.2}$$

An extra term B is added to the update step of a standard gradient update step. In formula every weight w_i in time-step t is updated based on the gradient g for the specific parameter i. Variable B needs access to all previous gradients g_τ.

5.6.2 Standardisation

As we have explained in chapter 3, standardisation could potentially help improve the convergence of our algorithm. Experiments in chapter 6.2 include the Secure Scaling Operation (SSO) 2.0 (see section 3.2.1). This standardisation algorithm guarantees data-privacy but also does not leak any mean or standard deviation to the PDSs. We also add a partial implementation from section 2.6.2 which we name Simple Secure Aggregation, which does not allow dropouts. This is due to time-constraints and we do not simulate dropouts in our experiments.

5.6.3 Stopping Criterion

Gradient descent is a step-wise optimisation towards the optimum. The most basic gradient descent algorithms include a number of iterations to be performed. Because of the need of reducing communication costs, the learning process can be terminated at an earlier point in time when the decrease in loss becomes negligible. We use a threshold for the validation set performance increase, to determine whether it is negligible or not. The validation loss can also increase, indicating possible over-fitting of the model. A stopping criterion could prevent this.

In a later version of the architecture, this stopping criterion was removed, as it had the tendency to stop late or early. Extra fine-tuning of this parameter brings extra complexity to the optimisation. Some extra steps to converge the algorithm further is acceptable. Now, we limit the algorithm on its amount of communication steps instead of based on convergence.

5.7 Evaluation

In this research, we are evaluating a framework, so the goal is to make it capable of solving a variety of problems consistently. That is why testing should try to solve a multitude of problems. UCI and Kaggle provide a broad variety of datasets, some of which are popular for testing ML models. The following datasets are used for benchmarking our proposed architecture. Every sample in the dataset must be generated by different subjects but is not limited to people. We prefer datasets with a size of 300 or larger. The information has to be medical, health(care), or related. The target variable should be binary or continuous to apply Logistic and linear regression respectively. Based on these requirements, the following datasets are used for benchmarking our proposed architecture.

- Medical Cost Personal Dataset (*Insurance*)[51]
- City Dataset (*City*)[53]
- CASP Protein Dataset (*CASP*)[19]
- TWOC Generated Dataset (*TWOC*)
- Haberman's Survival Dataset (*Haberman*)[19]
- Breast Cancer Coimbra Dataset (*Breast_cancer*)[45]
- Heart failure Clinical Records Dataset (*Heart_failure*) [14]
- Cardiovascular Dataset (*Cardiovasc / Cardio*) (source: Kaggle)

Applying linear regression to the *Insurance* Dataset [51] generates a predictor that can estimate possible future medical costs of a patient. The *City* Dataset [53] does not include patient info, but rather cumulative regional information. It is still very useful to test it on isolated PDSs. Linear regression can predict the mortality rate based on regional features concerning healthcare and non-healthcare. One dataset *CASP* contains the problem of predicting the tertiary structure of a protein based on other protein features [19], and is solved with linear regression. Final dataset is used to support the Trusted World of Corona project (*TWOC*). We use a subset of the TWOC dataset to fit into our architecture, TWOC-150. Therefore we rename the original TWOC dataset to TWOC-5000. The task of the dataset is to predict the severity of the disease.

The Haberman dataset [19] contains only a few features, including the task of classifying each patient whether the patient survives for at least 5 years or not. Logistic regression tries to model the features to predict this survival rate. The Breast Cancer Dataset from the University of Coimbra [45] tries to predict suffering patients based on patient features. Idem for the Heart Failure Dataset [14] but here its task is to predict death. We also try to predict cardiovascular disease from the Kaggle database.

The train set will be the same between central and federated experiments to eliminate any performance differences caused by chance. An experiment may use a subset, 100 or 150, of the samples available, which will be indicated.

5.7.1 Metrics

The measured performance of the model is highly dependent on the quality of the data for training (variance), but also on the model's ability to fit the data. A model with very little predictive power can causate from noise or a lack of correlation, but also from poor training capabilities of a model. This means that the predictive power alone is not a good measure of how well the model performs. The model must be compared to a model that was trained with a default regression method, centralised linear and logistic regression, or CentrLinReg and CentrLogReg.

As we are considering an architecture that should mathematically approximate another function, we can consider differences in metrics as a measure of how similar the functions are. In the field of data analytics, we could require a model that represents the underlying dataset to draw a conclusion over that dataset, i.e. inference. In the field of machine learning, it would be interesting to compare the models over their predictive capabilities, i.e. performance over unobserved data. In our experiments, we consider the differences in coefficients, differences in metric values over the training set and metric values over an unseen test set.

For linear regression, R-squared (R^2) is a commonly used measure, which indicates the percentage of the variable variance that the independent variables explain. A second metric for linear regression is the Mean Squared Error (MSE). For logistic regression the Area Under the Curve (AUC) indicates how well the model can distinguish between two classes. A second metric commonly used is Binary Cross-entropy.

We compare two different results by the percentage difference from the centrally optimised model. We do not compare coefficients or odds ratios with a relative or absolute distance, as it does not say anything about the performance of the model. Sometimes we take the absolute difference, where we think a relative comparison is illogical or can be interpreted wrong.

Chapter 6

Results

This chapter describes the experiments conducted during this thesis project.

We compare our Federated Linear Regression architecture to a Linear Regression baseline that calculates the underlying coefficients with OLS (Ordinary Least Squares), which is implemented in CentrLinReg. With this in mind, that means when we compare to the CentrLinReg model, we also compare to the optimal linear solution based on the least square error.

Logistic Regression implements the BFGS optimizer (see section 2.3.2), which estimates the logistic regression model parameters (here: CentrLogReg). This could result in a sub-optimal solution. While we regard the logistic regression model as the most optimal model, it could theoretically be possible to outperform CentrLogReg. This central algorithm is compared to our Federated Logistic Regression Architecture (FedLogReg).

Model Name	**Activation**	**Learning Rate**	**Iterations**	**Step Type**
FedLogReg-v1	Sigmoid	(to be optimized)	200	Batch
FedLogReg-v2	Sigmoid	0.01	800	Batch
SFedLogReg-v1	Sigmoid	0.01	800	Mini-batch
SFedLogReg-v2	Sigmoid	20	4000	Mini-batch
FedLinReg-v1	Linear	(to be optimized)	200	Batch
FedLinReg-v2	Linear	0.01	800	Batch

Table 6.1: Summary of different architecture settings.

Table 6.1 summarises different setting used in our experiments. We use two different activation functions, corresponding to Linear and Logistic regression models. The learning rate must be optimised in some settings, where we run the experiment multiple times to find the optimal result. Later experiments converge to a static learning rate. Iterations indicate the amount of sequential updates the final model receives until it terminates. In the field of research, there are three types of updates, Batch Gradient Descent, Mini-batch Gradient Descent, Stochastic Gradient Descent. We only implement the first two. Batch Gradient Descent uses all data available to generate an update step, while mini-batch gradient descent takes a subset of the data for every step. In our federated learning case, this is respectively the same as using all PDSs or a subset of

PDSs per step. All settings, except FedLogReg-v1 and FedLinReg-v1, include a method of standardisation by default.

6.1 FedLinReg-v1 and FedLogReg-v1

The first preliminary research focuses on hyper-parameter tuning and method testing to see its effect on the similarity between the centrally trained regression models and federated regression models.

We will be testing some optimisation parameters in the context of inference, or descriptive analytics. That means the trained model should make the correct assumptions over the trained data. Therefore we can exclude any test set. Later on, predictive analytics over a test set will be included. For the next experiments, we apply adaptive gradient and the secure scaling operation to see its impact on the performance. We also would like to know whether there is a correlation between the amount of noise in the data and the convergence to the correct coefficients.

> RQ2: *How much do adaptive gradient and data standardisation improve convergence?*

6.1.1 Noise Experiment

Our first experiment did not include any extra optimisation methods, i.e. excluding standardisation and adaptive gradient. It will compare the coefficients between the application of regular centralised linear regression and our federated solution.

Table 6.2 summarises the details with which the synthetic data was generated. All data is standardised and differs in the number of features included, the number of informative features included and the noise that was added to the target variable, based on a multiplication of the standard deviation of the original target variable. Only the input variables are standardised.

Details	Dataset					
no.	1	2	3	4	5	6
standardized	yes	yes	yes	yes	yes	yes
#features	1	1	2	2	9	9
#informative	1	1	2	2	5	5
output_noise	0σ	1σ	0σ	1σ	0σ	4σ

Table 6.2: Synthetic dataset details

This regular linear regression that is approximated with our federated learning method does not include any regularisation terms. We apply a learning rate of 0.05 after aggregating the gradients. By using a tolerance as stopping criterion, i.e. when the aggregated gradient does not exceed a certain value for a specified number of times, the learning process is terminated.

dataset	variable	Source → output_noise	Origin coef.	CentrLinReg coef.	FedLinReg-v1 coef.
1	1	0σ	60.84	60.84	60.83
2	1	1σ	63.78	70.32	74.29
3	1	0σ	17.83	17.83	17.82
	2		38.08	38.08	38.08
4	1	1σ	32.27	32.07	30.41
	2		86.45	84.45	85.07
5	1	0σ	27.62	27.62	27.63
	2		34.14	34.14	34.14
	3		0.00	0.00	-0.01
	4		0.00	0.00	0.00
	5		0.00	0.00	0.00
	6		99.89	99.89	99.87
	7		8.26	8.26	8.25
	8		0.00	0.00	-0.02
	9		12.31	12.31	12.31
6	1	4σ	0.00	0.50	3.34
	2		0.00	-3.18	-6.63
	3		62.72	58.40	57.72
	4		0.00e	-4.23	-5.22
	5		91.38	97.15	94.24
	6		92.97	91.06	92.15
	7		48.93	41.80	42.51
	8		65.42	65.60	63.76
	9		0.00	-2.31	-1.92

Table 6.3: Final coefficients from the central and federated model over synthetic data.

Observations

As we observe in table 6.3, both the CentrLinReg and FedLinReg-v1 did not converge to the original coefficients in all scenarios. This is because when noise is added to the dataset, this may influence the ideal linear coefficients for that data. The most important aspect of this table, however, is the difference between CentrLinReg and FedLinReg-v1. Datasets with more noise in their correlations come with a bigger deviation between CentrLinReg and FedLinReg-v1. The current architecture does not approximate the original coefficients, nor does it approximate the coefficients of CentrLinReg in all cases.

Interesting is that in FedLinReg-v1, all the data without noise, i.e. clear linear relations, do already generate almost the exact same coefficients as the central model. Unfortunately, this is an unrealistic scenario.

6.1.2 Adaptive Gradient

To test our Adaptive Gradient in a federated setting, we evaluate the similarity to the central least-squares baseline model. This experiment does not suggest anything about its predictive capabilities, only on the capabilities to fit a function over the input data. The difference with the previous experiment is that the data may include different features the synthetic data had not accounted for. A possible scenario is that there is no correlation at all, only noise. As we have seen, this may result in widely different coefficients. This does not necessarily represent any difference in performance. For this reason, only the relevant error rates are regarded. For logistic regression, we use the Log Loss as a measure of performance. For linear regression, we use Mean Squared Error.

The results include columns with results from our basic FedLinReg and FedLogReg solution and added AdaGrad (AG). The percentage indicates the distance to the central algorithm in terms of loss value. The learning rate (η) is the most common hyper-parameter to tune when it comes to gradient optimisation problems. The number of optimisation involved plays a potential hurdle in the computation of a model. In our scenario, we do a parameter sweep of the learning rate with fixed values, by using the formula $\eta = 5 * 10^p$ where $p \in [-3, ..., 2]$. The best performing variable is used. If no convergence is reached (η too small) or loss value only increases (η too big) within the sweep, we continue until we find a learning rate where loss is minimal.

While there was a stopping criterion, there was not any guaranteed convergence. We remove the stopping criterion and train within a fixed number of iterations from now on to make the most effective use of the time-resources available. The number of iterations is capped on 201. Most of the convergence is done within the first iterations and we assume this is a rough indication of how well the model will converge relative to others. This experiment compares the training loss between models that are optimized differently.

Observations

		CentrLinReg	FedLinReg	FedLinReg+AG
Insurance	MSE	1.55E+08	**1.57E+08**	1.58E+08
	% diff	(ref)	+1.62%	+2.06%
	η	-	5E-5	50
City_data	MSE	2.32E+00	1.23E+01	**4.59E+00**
	% diff	(ref)	+429.17%	+97.63%
	η	-	5E-6	50
CASP	MSE	2.47E+01	2.95E+06	**1.39E+05**
	% diff	(ref)	+11947571.29%	+563765.98%
	η	-	5E-14	5000

Table 6.4: Testing results of different FedLinReg implementations

In tables 6.4 and 6.5 the results of this experiment are shown. The most basic form of Federated Learning using FedLinReg and FedLogReg showed signifi-

		CentrLogReg	FedLogReg	FedLogReg+AG
Haberman	log-loss	0.4617	2.9201	**0.5483**
	% diff	(ref)	+532.40%	+18.74%
	accuracy	0.77	0.74	0.74
	η	-	0.005	0.05
Breast_cancer	log-loss	0.3446	inf	**2.7700**
	% diff	(ref)	inf	+703.93%
	accuracy	0.82	-	0.47
	η	-	-	0.05
Readmission	log-loss	0.6388	**1.2079**	1.9001
	% diff	(ref)	+89.08%	+197.43%
	accuracy	0.61	0.41	0.46
	η	-	0.005	0.005

Table 6.5: Test results of different FedLogReg implementations

cant drops in performance. By applying an adaptive gradient, AG, the results generally shifted more towards the centralised least-squares baseline (CentrLinReg or CentrLogReg). AG-optimised models are not necessarily better as the two tasks over the datasets *Readmission* and *Insurance* perform worse. Increase in loss over the *CASP* dataset is still not enough to approximate the centrally trained model. But in most cases it causes some increase in performance.

The learning rate η is easier to optimise when using AG, as optimal learning rate does not vary as much when using AG compared to without.

6.1.3 Standardisation

For the third experiment, we are able to use a maximum of 150 simulated PDSs. The error rates are averaged over ten sessions to get a better estimate of how well the model performs. We iterate 201 times. We generated a new train-test split for this experiment. We use the learning rates to optimize the model which uses the non-standardised data. The model which uses the Secure Scaling Operation (SSO) 2.0 (see section 3.2.1) uses a constant learning rate of 0.1, as the convergence does not depend on the scale of the data anymore. For the original FedLinReg we again do a learning-rate parameter sweep as in the previous experiment.

The advantage of goodness-of-fit methods is that they are not influenced by the scale of the variables, contrary to error measures like Mean Squared Error (MSE) that depend on the exact values of the output. The R^2-test is the most common measure for goodness-of-fit in linear regression models. Because we compare a model for scaled and unscaled data, it is not possible to use error measures like MSE. For logistic regression, the dependent variable is not scaled, as this is categorical data that the algorithm requires. Thus the log-loss error rates can be used for comparison.

Observations

Dataset	Metric	CentrLinReg	FedLinReg	FedLinReg + SSO 2.0
Insurance	R^2	0.11542	0.10972	**0.11542**
	abs-diff	(ref)	-0.0057	-0.0
	η		5E-05	0.1
city_data	R^2	0.14370	-546.10922	**0.14370**
	abs-diff	(ref)	-546.25292	-0.0
	η		5E-09	0.1
CASP	R^2	0.29178	-75750.62633	**0.29084**
	abs-diff	(ref)	-75750.91659	-0.00094
	η		5E-14	0.1

Table 6.6: Results of performance through standardisation. Higher is better.

By comparing table 6.6 and 6.7, we can see the impact of our federated standardisation. For these datasets, the architecture which includes SSO 2.0 outperforms the architecture without standardisation and drastically improves on the convergence to the central model.

Earlier, we hypothesised that no parameter sweep was needed for SSO 2.0 to generate a better performance. This is indeed the case for those samples, but it could still be that the learning rate is sub-optimal. We do however conclude that results are comparable deeming the optimisation of the learning rate significantly less important. In federated learning, the execution of a learning rate parameter sweep can take very long, as the optimisation over a network

Dataset	Metric	CentrLogReg	FedLogReg	FedLogReg + SSO 2.0
Haberman	log-loss	0.49374	0.67341	**0.49395**
	% diff	(ref)	+36.38%	+0.04%
	η		5E-3	0.1
Breast_cancer	log-loss	0.48160	15.48325	**0.52266**
	% diff	(ref)	+3114.96%	+8.52%
	η		5E-6	0.1
Readmission	log-loss	0.67563	2.54954	**0.67564**
	% diff	(ref)	+277.35%	+0.00%
	η		5E-3	0.1

Table 6.7: Results of performance through standardisation. Lower is better.

like the internet may take significantly longer time. This makes SSO 2.0 very advantageous.

6.2 FedLinReg-v2 and FedLogReg-v2

Previous preliminary research showed that without standardisation the resulting model performance is significantly lower. The AdaGrad method did generally improve the performance but with margins that are relatively small. We, therefore, decided to exclude it from our subsequent experiments as we hypothesise that the performance will be sufficient. All other subsequent experiments are done with the default FedLinReg-v2 and FedLogReg-v2, which includes the SSO 2.0 algorithm, static learning rate of 0.1, and a number of iterations equal to 800 instead of 200. With this architecture, we try to answer the following research question.

> RQ3: *How well does our architecture approximate linear regression and logistic regression?*

With an improved standardisation protocol from section 3.2.1, we continue to run the following experiments to answer the main research questions for this thesis. We decided to drop the non-standardised regression method as a means of calculating the regression model, as previous experiments exposed too many issues. We also decided to drop the adaptive gradient method that increased the convergence rate slightly, because improvements were below our expectations.

6.2.1 Coefficients in FedLinReg-v2

We run the same experimental setup as the experiment in section 6.1.1, but here we use real data. We use two different subsets of the CASP dataset, splitting vertically (feature selection). This could maybe give us hints on the influence of the number of variables on the similarity of the models.

> RQ3a: *How do coefficients of our linear regression model, trained via federated learning, compare to the coefficients of classic linear regression?*

Observations

While the results over the *Insurance* and *City_data* datasets are extremely similar, the same cannot be said about the *CASP* dataset (see figure 6.8). We argued that the big deviation of coefficients in the *CASP* dataset may be caused by the amount of variables that are included in the model. Therefore we ran the same experiment on a subset of the variables of *CASP* (*CASP_subset*).

CASP_subset only includes four features and still deviates quite significantly from the original parameters. We hypothesise that this must be caused by some other aspect of the dataset that influences the convergence rate. In sum, OLS (CentrLinReg) does not necessarily converge to the same coefficients as its federated counterpart, at least for our data samples.

		CentrLinReg	FedLinReg-v2
Dataset	**Vars**	**Beta-coefficient**	**Beta-coefficient**
Insurance	age	0.34568	0.34568
	bmi	0.06179	0.06179
	children	-0.07045	-0.07045
city data	doc-avail	0.16844	0.16844
	hosp_avail	0.11678	0.11678
	income_1000s	-0.21360	-0.21361
	pop_density	-0.26855	-0.26854
CASP_subset	F1	-0.69146	-0.10181
	F2	1.19925	0.60785
	F3	-0.03126	0.20029
	F4	-0.35510	-0.46978
CASP	F1	0.92523	0.17667
	F2	0.81175	0.85069
	F3	-0.03544	-0.05841
	F4	-1.23966	-1.24282
	F5	-1.07380	-0.41327
	F6	0.73016	0.79381
	F7	-0.38525	-0.38342
	F8	0.15355	0.16858
	F9	-0.0717	-0.05954

Table 6.8: Final coefficients from the central and federated model.

6.2.2 Odds Ratio in FedLogReg-v2

As we explained in section 2.3.2, logistic regression uses Odds Ratio as a measurement of constant influence by an independent variable on the predictor likelihood. The modelled effect can best be described by the Odds Ratio, which is calculated from the coefficient.

When the original data is categorical, the Odds Ratio of a certain category value represents the increase in the likelihood of the predictor variable occurring compared to a baseline category value. This value can sometimes be manually calculated or from the fitted logistic regression model. With these values, A decentrally optimised model can be compared to a central.

When the original data is continuous, the Odds Ratio depicts the increase in likelihood of the predictor variable when the independent variable is increased with one unit. This value can be derived from the logistic regression model only, and not directly from the raw data itself. That means we cannot check the validity of the Odds Ratio against the original but can check the odds ratio of different logistic regression models. We compare a centrally optimised logistic regression model against one which is decentrally optimised.

The OR is calculated by comparing the odds of the predictive variable given a categorical value to the predictive variable given the reference value. The Adjusted Odds Ratio (AOR) has the same meaning, except for that it accounts for all other independent variables. Both OR and AOR are calculated in two different scenario's: over the source data, and over the central and federated logistic regression model. When the OR of an input variable is 1.1, it can be interpreted as: "An increase of one standard deviation in the input variable increases the likelihood of having cardiovascular disease with 1.1". The algorithm used 800 iterations for model training.

The data used to run this experiment is the Cardiovascular dataset. We split the dataset on the feature axis, one with mainly binary variables (*cardio1*), one with continuous ones (*cardio2*). Thus the AOR only includes the variables in the same subset.

> RQ3b: *How do the odds ratios of our logistic regression model, trained via federated learning, compare to the odds ratios of classic logistic regression?*

Observations

					CentrLogReg		FedLogReg-v2	
ds	Variable Type	Variable	Grouped	ref?	OR	AOR	OR	AOR
cardio1	continuous	Height (cm)	-		1.0987	1.0961	1.0987	1.0961
		Weight (kg)	-		1.2195	1.2824	1.2195	1.2824
		Age (yrs)	-		1.8482	1.8366	1.8482	1.8366
cardio2	binary	Gender	female	(ref)				
			male		1.1075	1.1609	1.1021	1.1609
	binary	Smoking	no	(ref)				
			yes		0.9649	1.1787	0.9649	1.1777
	binary	Alcohol	no	(ref)				
			yes		0.6138	0.5537	0.6138	0.5545

Table 6.9: Results, with the OR indicating the influence of different variables X on having a cardiovascular disease

Results of this experiment are in table 6.9 The maximum difference between central and federated OR/AOR values is 5/100th, which is a very small margin. We cannot conclude with certainty that this resemblance holds in other datasets. We even hypothesise that this is not the case, just like in the previous experiment, where the coefficients deviated when the correlation is noisy.

6.2.3 Inference in FedLinReg-v2

In previous experiments, we checked whether the model parameters are similar in both the centrally optimised model and the one fitted in our architecture. Both methods try to create a model that best fits the observations, thus it would be useful to see whether different fit-metrics results diverge from the central one. Those methods are not applicable in a distributed way, as the data is not centrally accessible, but may be calculated in our experimental setup and be compared to the CentrLogReg values.

For linear regression and FedLinReg, we use standardised input and output to get optimal convergence. The model can later be converted to non-standardised parameters, see sec 3.1.

> RQ3c: *How well can a linear regression model, trained via federated learning, fit the data compared to classic linear regression?*

Observation

	CentrLinReg		**FedLinReg-v2**		
Measure	R^2	**MSE**	R^2	**MSE**	% Diff
Insurance	0.13323	0.86676	0.13323	0.86676	+0.0
City	0.14370	0.85629	0.14370	0.85629	+0.0
CASP	0.39233	0.60767	0.39087	0.60913	+0.24026

Table 6.10: The R-squared goodness-of-fit measurement for central and federated models over some datasets

As can be seen in table 6.10 Federated architecture mirrors the same R-squared values calculated over the test data. For the datasets included in this experiment, the goodness-of-fit converges even for the *CASP* dataset. A model with different coefficients does not necessarily mean that it has a worse fit on the data. For example, the CASP dataset with 9 independent variables, has an almost identical R-squared value, indicating that the goodness-of-fit is almost identical, while the coefficients from those models deviate quite drastically (see the experiment in section 6.2.1).

6.2.4 Inference in FedLogReg-v2

For CentrLogReg and FedLogReg-v2, we use standardised input variables only, as most logistic regression libraries require the standard binary values 0 and 1 in the dependent output variables.

> RQ3d: *How well can a logistic regression model, trained via federated learning, fit the data compared to classic logistic regression?*

The difference is calculated depending on the log-loss, as this metric is non-thresholded, i.e. taking into account the probability of a class (between 0 and 1) and not only the predicted class (0 or 1). Taking a percent difference from the AUC that sits between 0 and 1 and how it is interpreted can be ambiguous.

Observations

		CentrLogReg		**FedLogReg-v2**	
Dataset	AUC	**Log-Loss**	AUC	**Log-Loss**	% Diff
haberman	0.649315	0.57381	0.649315	0.57381	+0.0
readmission	0.566428	0.68334	0.566250	0.68334	+0.0
breast_cancer	0.848558	0.48160	0.843450	0.48318	+0.32807
*cardiovasc**	0.678929	0.64174	0.678393	0.64178	+0.00623
*TWOC-150**	1.0	1.61522e-06	1.0	0.04132	+2558165.4

Table 6.11: The AUC values calculated over the train data after training. *= unused datasets in preliminary experiments.

ROC curves are included in the appendix (table 8.1). In table 6.11, the differences between the central and federated architecture are smaller than previous architecture FedLogReg-v1.

While the AUC uses the predicted output class (0 or 1), the log-loss uses the predicted output probability (between 0 and 1) as a comparison against the true values. Although the error over the TWOC-150 dataset is quite high compared to the central model, this does not damage its fitting capacity as the AUC remains high. The difference measurement is heavily dependent on the size of the central log-loss. A small log loss does increase the impact of small deviations, as happened in the TWOC-150 dataset. If we calculated the absolute difference, both AUC and log-loss differences would be almost 0. Maybe the absolute difference would have been a better measurement but that is up for debate.

When regarding the best fit for our data, error becomes more important. For models that cannot completely distinguish between classes (i.e. AUC < 1) the Loss differences are all between 0% and 0.33%. If the model can separate the classes completely (=TWOC-150), it models the data classes correctly, but the coefficient estimate is infinite, which is where maximum likelihood estimator used in CentrLinReg breaks down. There are solutions to this, such as regularisation,

but we omit such implementations due to time-constraints. This means that for all generated models that do not separate output classes perfectly, differences between error rates and AUC are negligible.

6.2.5 Prediction

In machine learning, models may be used for prediction. We estimate its performance by using metrics that measure performance on an unseen test set. There are solutions better suited for prediction, such as ridge regression and regularised logistic regression. It is however still valuable to see its effect over unseen data.

We excluded some of our datasets, as we would like to train our model with 150 PDS instances, and test over another 150 data samples. This means that there should be at least 300 samples per dataset. The TWOC-5000 dataset has enough samples but is severely imbalanced, i.e. some classes are extremely prevalent and others are rare. We do not know what the effects are of testing over a severely imbalanced dataset and assumed for now that it is not guaranteed to be a good measure for our architecture. Therefore we excluded it. Later on, we do train the TWOC-5000 to analyse its fit over the train data in section 6.2.6. We did already use the TWOC-150 dataset for fitting a model.

RQ3e: *How well can a linear regression model, trained via federated learning, predict compared to classic linear regression?*

RQ3f: *How well can a logistic regression model, trained via federated learning, predict compared to classic logistic regression?*

Observations

Dataset	CentrLinReg **MSE**	FedLinReg-v2 **MSE**	**% difference**
Insurance	1.04783	1.04783	+0.00000
CASP	1.27546	1.27833	+0.22501

Table 6.12: Squared error values calculated over an unobserved test set

Dataset	CentrLogReg **Log-Loss**	FedLogReg-v2 **Log-Loss**	**% difference**
cardiovasc	0.67747	0.67588	-0.23469
haberman	0.51933	0.51933	+0.00000
readmission	0.69812	0.69812	+0.00000

Table 6.13: Log-loss values calculated over an unobserved test set

According to the results in tables 6.12 and 6.13 it is possible for our federated algorithm to generate a better performance over the test set. This is caused

by small differences in the model. As the error rates over the training sets deviate only slightly, the difference in coefficients can also be advantageous to the federated model.

Differences in prediction capabilities only deviated 0.24% at maximum for the included datasets.

6.2.6 Mini-batch Learning

The TWOC dataset is a realistic generated dataset with a significant imbalance, which could also occur in realistic situations. The dataset includes 5000 different patients, which means we are unable to run all of our patient data in PDS instances, as this is computationally too heavy. We can either create a simulation of the federated learning algorithm to see its performance, or we can use a mini-batch of PDSs each update step.

PDS mini-batch selection is different in the sense that it does not use all available data in each iterative step, instead, we take a subset of the available PDSs, generate an update step, and update the global model. This means that updating the global model is a stochastic update, i.e. an approximation of the total gradient is used. The next iteration takes a different subset of the PDSs. With a simple programmable conversion, we were able to generate a new subset of PDSs for every iteration.

We ran the algorithm with 800 iterations and a learning rate of 0.1. Under those settings, we call the algorithm "SFedLogReg". The data was not standardised as all the data was already in the range of 0-1.

> RQ4: *How does a logistic regression model, trained via federated learning, converge when using mini-batch updates?*

Observations

	CentrLogReg		**SFedLogReg-v1**		**SFedLogReg-v2**	
Variables	**Coefficients**	**Odds Ratio**	**Coefficients**	**Odds Ratio**	**Coefficients**	**Odds Ratio**
M	-57.80842	7.83649e-26	-2.68384	0.06830	-5.49016	4.12716e-03
P	56.92476	5.27364e+24	-0.01337	0.98670	20.04858	5.09319e+08
*M*P*	64.18585	7.50869e+27	-0.48963	0.61285	3.34734	28.42714
AUC	0.99983		0.81014		0.99975	
Log-Loss	**0.00294**		**0.06768**		**0.00879**	
Precision	0.98019		-		0.97872	
Recall	0.99		0.0		0.92	

Table 6.14: Experimental results over the *TWOC-5000* dataset.

The results of our SFedLogReg-v1 algorithm is a sub-optimal performance and does not predict the correct classes (see tables 6.14 and 6.15). While in previous experiments a single batch learning phase did practically always converge, for this TWOC-5000 dataset a mini-batch optimisation method does not converge. Recall is 0 as there are only negative retrieved predictions, while Recall focuses only on positive samples.

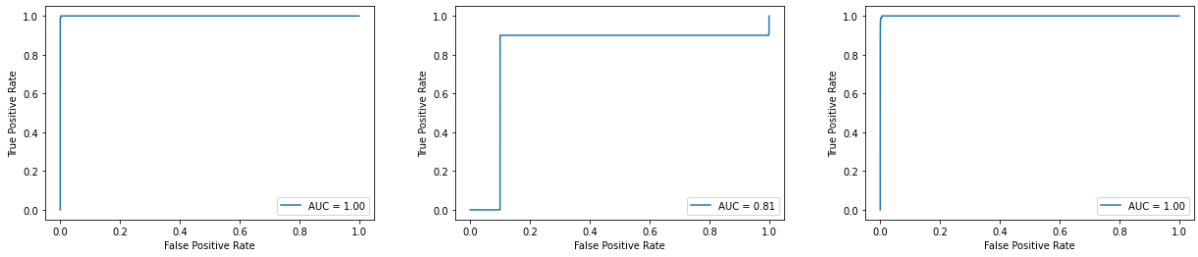


Table 6.15: CentrLinReg (left) ROC curve, SFedLogReg ROC curve (middle and right)

Only after changing the learning rate from 0.1 to 20 and increasing the number of iterations from 800 to 4000, we get a method (SFedLogReg-v2) that generates a model that somewhat fits the data. The log-loss drops from 0.07 to 0.009, which is much closer to the value of 0.003 from the central model. According to the AUC value and comparing it to SFedLogReg-v1, classification of the data samples is many times closer to CentrLogReg.

This experiment indicates that using a subset of the data from TWOC-5000 dataset in every update step (under the same learning rate and amount of iterations as FedLogReg) is not optimal. A possible hypothesis is that stochastic updating needs more iterations and/or different learning rate. This is up to future research. It would be useful to find settings that are optimal for both batch updating (no subset) and mini-batch updating (subset).

6.2.7 Efficiency

The simple secure aggregation protocol (SSA, see section 2.6.2) makes use of the Diffie-Hellmann key exchange, which is known for its high computational load for especially edge devices with limited power. We, therefore, propose an experiment that explores the influence of the number of PDSs on the computation time on both a server and a client.

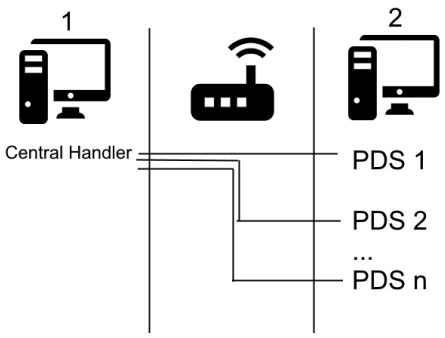


Figure 6.1: Setup for testing efficiency of Central Handler

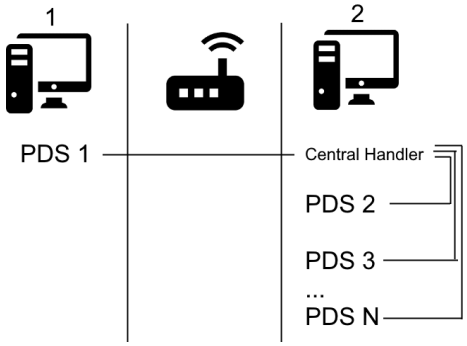


Figure 6.2: Setup for testing efficiency of a PDS

Figure 6.1 and 6.2 give a graphical network visualisation to test the efficiency of our algorithm. When simulating the algorithm on a single device,

computation time may be influenced by other PDS instances. To create a stable environment for the PDS, we isolate one PDS and locate it onto a secondary device within the same local network. We do the same for the central handler which will then communicate with all the PDSs simulated on a different device.

The code for both central handler and PDS mainly consist of single-threaded computation, except for efficient communication. We used two devices, with Intel Core i7-8550U and AMD Ryzen 3100 CPU. When calculating computation time on a device, the architecture is mainly run on the Ryzen device, while the computation-time was measured when running a separate instance (central handler or PDS) on the Intel device.

RQ5a:*What is the influence of the amount of personal data stores in the network on the computation time for the central handler?*

RQ5b: *What is the influence of the amount of personal data stores in the network on the computation time for every personal data store?*

Observations

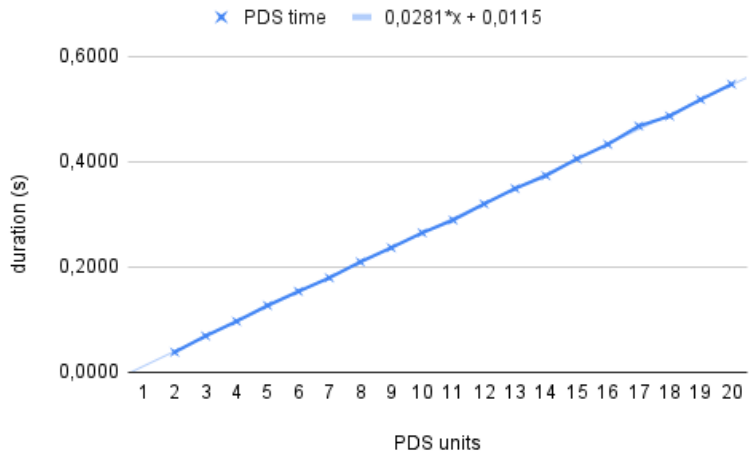


Figure 6.3: Computation time of a client in relation to the number of clients included in simple secure aggregation.

To summarise, the duration in the graphs 6.3 and 6.4 is measured on separate devices without interference of other parts of the network, and therefore being a good indication of real practical duration. The communication-time however is not included as no distant communication via the internet was used.

See figure 6.3. Incrementation in the number of PDSs included in simple secure aggregation causes a linear increase in computation time for a single PDS device. The computation time for one iteration in a PDS roughly increases with one second when 36 PDSs are added to the network. Without applying Simple Secure Aggregation, the algorithm only takes a constant 0.02 seconds to

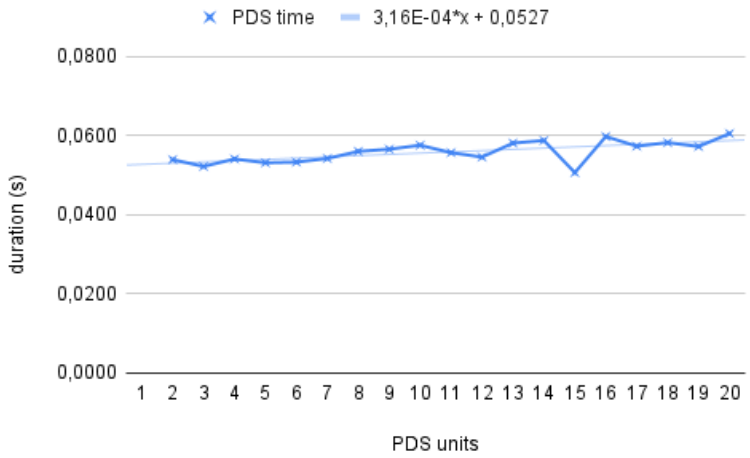


Figure 6.4: Computation time of the central handler in relation to the number of clients.

finish an iteration for all amounts of PDSs involved. Therefore, the key agreement algorithm for sharing secrets between clients was the sole cause for all the computational expenses.

This means that using hundreds or thousands of PDSs in our network is unfeasible. One solution would be to split up the PDSs in mini-batches, while maintaining a single batch gradient descent step. In other words, for every PDS mini-batch, we apply simple secure aggregation. For example, from 200 PDSs we create 10 mini-batches of 20 PDSs. Every mini-batch applies simple secure aggregation internally. This will reduce computational costs and will roughly be equal to the duration of only applying simple secure aggregation to one group of 20 PDSs (see graph 6.3) instead of 100 PDSs. The smaller the mini-batches are, the faster computation time will be but updates will be less secure.

So what is the security risk of this reduction in computation cost? For example, running simple secure aggregation over 2048 PDSs for 800 iterations will take an estimated time of 13 hours. If the simple secure aggregation algorithm is only applied over mini-batches of 16 PDSs, run-time is reduced to approximately 15 minutes, excluding communication-time. In practice, this means that if an honest-but-curious adversary controls the central handler and a group of 15 specific PDSs, it would be able to extract the data of the 16^{th} PDS, which is a privacy leak. This architecture would be t-private, where $t = 15 + 1 - 1$, i.e. 15-private. If any 15 honest-but-curious parties collaborate (15 PDSs + 1 central handler is required), still no private information is leaked.

In graph 6.4, the central handler computation time roughly increases with one second when 3200 PDSs are added to the network. This means the performance of the central handler is depending significantly less on the amount of PDSs involved than the PDSs themselves. Still for big amounts of clients it could slow down the training process.

Chapter 7

Other Methods for Data Modelling in Personal Data Stores

The field of federated learning heavily depends on gradient optimisation. We would like to discuss other gradient and non-gradient models that could be optimised in a decentralized setting. We again prioritise the data-privacy of the data owners and do not necessarily focus on model-privacy.

7.1 Gradient models

Regularized Regression

Regularisation is the act of adding information to a model mainly to prevent it from overfitting. It is possible to add a regularisation term to some regression models, but then the regularised model will carry a completely different name. Depending on the size regularisation term λ, the coefficients are more drawn towards lower values. The resulting simpler model may have a lower error rate on unseen test sets.

A regularisation value is added to the loss function by multiplying the trade-off parameter, λ, to the sum of the regularisation terms for all parameters, for example *ridge regression* uses L2-regularisation, $Loss = MSE + \lambda \sum_{j=1}^{n} \theta_j^2$. The loss is higher for higher coefficient values, where minimising the regularisation term means setting all coefficients to zero. *Lasso regression* uses L1-regularisation $\lambda \sum_{j=1}^{n} |\theta_j|$. For *elastic net regularisation* both types of regularisation are used, $\lambda_1 * L1_{ratio} * \sum_{j=1}^{n} |\theta|_n + 0.5 * \lambda * (1 - L1_{ratio}) * \sum_{j=1}^{n} \theta_n^2$

Linear regression could be optimised with Ordinary Least Squares, which could provide a single best answer. For ridge regression, a slightly altered closed-form solution can locally be calculated via the Tikhonov regularisation [24], see equation 7.1.

$$\beta = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T y \tag{7.1}$$

But all regularised regression models mentioned above may be optimised with loss minimisation by calculating the gradient of the regularisation terms. For lasso, ridge, and elastic net regression holds, the loss functions are convex,

but not all are strictly convex. This means that a local loss minimum is always the global minimum but it may not be the only global minimum. For our research this means that when we minimise the loss, it always finds an optimal solution in a possible range of best solutions (minimal loss). This is identical to our research on non-regularised regression methods.

Regularised regression methods are extremely similar to their non-regularised counterparts and with slight alterations of our architecture it is possible to create regularised models. As we have seen in experiment 6.2.4, with a logistic regression task that is completely separable, it is even advised to use regularised models. We hypothesise that the convergence rate is similar under similar settings but this is up to future research.

Support Vector Machines

Support Vector Machines (SVMs) were originally developed for object classification [41]. The algorithm is a linear classifier where the goal is to maximise the margin between two (separable) classes in a dataset. A learned classifier can calculate whether a new data sample is belonging to class A (positive) or B (negative) and how far it is from the separation line. The basic SVMs include linear separation. Instead of using a logistic function, the linear function is used, as the result can better indicate the distance of a data sample to the separation line. We choose to apply soft-margin gradient descent instead of hard-margin, as this may be applied to both separable and non-separable data. It also generalises better. The hard-margin variant has a similar loss function and is possibly similar when creating a federated learning solution to this problem. Some SVMs have been trained in a federated fashion [59, 10]

The classifier can be optimised by using gradient descent, as it is possible to derivate the linear activation function and the hinge loss function. The classifier optimises equation 7.2.

$$\lambda||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b)) \tag{7.2}$$

The prior part of the formula includes a regularisation term, while the latter part implements the hinge loss function, punishing values that are too close to the separation line or on the wrong side. The vector $\mathbf{w}$ along with intercept b are to be optimised. The gradient can be calculated for every weight or intercept on every PDS. The regularization can be calculated in the central handler or on the PDS. For the latter, the value must be divided by the amount of PDS in the network, otherwise the regularisation term will be calculated many times and summed, instead of just applying it once per iteration.

SVM Implementation

We were able to practically simulate an architecture where the gradients were separately calculated and later summed. Regularisation was not calculated on

the central handler but in every PDS. No simple secure aggregation is applied as we only would like to see the potential performance.

For this short research we use two-dimensional samples all belonging to one of two classes. It is also possible to apply the algorithm to more than two dimensions, which will create a hyper-plane that can separate the samples. We generate different separable dataset and compare a classic Support Vector Classifier to our federated implementation. Learning rate, η, of 0.01 is used. λ is the strength of regularisation, in our case it is equal to 1. We iterate a thousand times without any early stopping methods. SVMs may have different purposes and quality can be measured in different ways. For this limited research on SVMs, we only visualise the plotted results which may be interpreted differently by the reader.

Results comparing it to regular SVM training can be seen in figure 8.2 in the Appendix. In general, SVM with federated learning is comparable to its central counterpart. Updates can be masked the same way as in federated regression, as has been discussed in section 2.6.

Deep Learning

When regarding deep learning, the techniques compared to training SVMs or Regression models is similar. Most papers regarding Federated Learning are focused on the optimisation of artificial neural networks [32]. Deep learning is focused on optimising neural networks but also requires extremely many data samples.

The biggest difference is that the neural network does not have a convex error landscape, which means that it is highly likely to convert to a local optimum. As we explained in a previous chapter, stochastic updating is required, i.e. selecting a subset of the PDS's which may cause the learning process to escape a local optimum and arrive at a better model.

Local training of a neural network is already very computation-heavy and thus takes longer than previously discussed models. It could take tens of thousands of iterations. In our scenario with PDSs, every iteration needs a communication step. That means that the learning duration of a neural network may be extremely high. This is up to future research.

Some extra tools for the training of neural networks is required. Overfitting is a more prominent threat in neural networks and the federated learning architecture must split up the PDSs in different sets, for training, validation and testing. Other metrics to measure the optimisation must also be implemented and securely aggregated, to better analyse the training process. The current architecture for regression must also be expanded in this way.

Yes, it is possible to optimise neural networks this way, but, neither central training nor federated learning guarantees an optimal solution.

7.2 Non-gradient models

We only discuss one non-gradient model which is popular in medicine due to time-constraints. We focus on analysing the solution in the context of Personal Data Stores and privacy preservation.

Decision Trees and Random Forests

Optimising random forests for horizontally-split data is a not commonly researched topic [35]. Random forests iterate the creation of decision trees and uses ensemble decision making to evaluate the results from all trees. So is it possible to create a decision tree while preserving user-privacy? Xiao et al. proposed a method to create a decision tree while keeping the privacy of data-owners [58]. It alters the ID3 algorithm which is normally used to optimise decision trees centralised. The algorithm and its federated counterpart uses entropy and information gain to choose the correct attribute to split the data over. We can calculate the information gain from the entropy. ID3 uses entropy to build its tree. The entropy is calculated with equation 7.3, and consequently the information gain is calculated with equation 7.4.

$$Entropy(S) = -\sum_{i=1}^{n_{target}} p_i * log_2(p_i) \tag{7.3}$$

$$InfGain(S, A) = Entropy(S) - \sum_{v \in values(A)} (((|S_v|/|S|) * Entropy(S_v)) \tag{7.4}$$

ID3 calculates the entropy by calculating p_i. This value is calculated by summing the amount of samples of a certain target category i (e.g. positive or negative) and dividing it by the total amount of samples in the dataset. Then the information gain can be calculated by calculating the change in entropy between the total dataset S and all the categories S_v of a certain attribute A. This algorithm is repeated by splitting the original dataset into the categories of the attribute that maximises Information Gain and process is repeated over the resulting datasets. This attribute is also added to the tree.

It is possible to ask a PDS whether it is from a certain category, and secure aggregation can sum the resulting responses privately, thus making it possible to request p_i. Both $|S_v|$ and $|S|$ can also be requested and aggregated the same way. This algorithm is repeated by splitting the original dataset into the categories of the attribute that maximises Information Gain. But the central handler does not know which PDSs belong to which category of the attribute, but this is not a problem. Concurrent requests to the PDSs should incorporate the current decision tree and a PDS only answers when it is supposed to. When it is not supposed to, a response is still generated but this does not influence the algorithm. Thus, the PDSs function as being one single server, p_i is requested and information gain can be calculated, which is needed for the decision tree to be built.

Because the feedback of the PDSs is very limited, in some cases the values of certain data stores may be estimates quite accurately. If the aggregate has an extremely low or extremely high value, the certainty of all the PDSs in the group belonging to respectively 0 or 1 increases. For example, if the aggregate is 0, it is certain that all the included PDSs returned 0, even if it was masked. The same holds if all PDSs returned 1. With continuous variables like the gradient values, single updates cannot be estimated. Thus, this method cannot be deemed totally private. And if we would apply mini-batch secure aggregation that we proposed in section 6.2.7, the risk of data leakage increases.

So what does this mean for Random Forests? Multiple decision trees can be generated over different subsets of the data. The process remains exactly the same for two exceptions. The starting dataset for every decision tree is (slightly) different. And the attribute selection process uses dropouts, meaning that some attributes are excluded when calculating which attribute generates the highest information gain. These methods may safely be implemented in the central handler without any privacy issues. This is a short theoretical foundation to semi-privately learn decision trees and random forests, and further practical research should expose possible problems in efficiency.

A more popular topic in federated learning are Gradient Boosted Decision Trees. Some published papers propose algorithms that can optimise Gradient Boosted Decision Trees in a federated way [60, 33, 13]. This thesis is limited to decision trees and random forests and GBDT will not be discussed further.

Chapter 8

Conclusion

> RQ1: *How do we ensure data-privacy and model-privacy when creating regression models trained with federated learning?*

Literature research provided an answer to this research question, mainly explained in chapter 4. While literature provides different Federated Learning solutions to optimise models such as regression models, those models do not guarantee data-privacy and model-privacy per se. A widely accepted solution, Secure Aggregation, proved to be a solution to keep the data and gradients hidden from the server and other PDSs when parties are honest-but-curious. Differential Privacy does not work in our case. It is possible to guarantee both model-privacy and data-privacy when training regression models with data from personal data stores by implementing the *PrivFL* architecture.

> RQ2: *How much do adaptive gradient and data standardisation improve convergence?*

In our preliminary experiments we tried to improve convergence as it was insufficient. We applied a slightly different implementation of Secure Scaling Operation 2.0 (see section 3.2.1) which is able to standardise the data. AdaGrad was able to adapt the gradient to potentially improve convergence. While AdaGrad only saw slight inconsistent improvements, standardisation was significantly better.

> RQ3: *How well does our architecture approximate linear regression and logistic regression while applying federated learning and preserving data-privacy?*

We have built our final architecture FedLinReg-v2 and FedLogReg-v2 as an attempt to approximate linear and logistic regression respectively. It averages all the gradients of personal data stores and updates the central model. For best convergence, we used Secure Scaling Operation 2.0. For optimal data-privacy, we implemented simple secure aggregation, a simpler version of secure aggregation without possible dropouts.

We observed that coefficients may not be equal, but no large deviations in fit and prediction scores have been measured. All loss differences are between

0% and 0.33%. There is one exception where the model was able to separate the data perfectly (TWOC-150). It did not percentually approximate the loss, but its absolute difference is still negligible.

> RQ4: *How does a logistic regression model, trained via federated learning, converge when using mini-batch updates?*

Results showed that the convergence is insufficient when using a mini-batch in every update step. Changing learning settings like the learning rate and number of iterations can improve the convergence, but future research has to validate this on other data as well.

> RQ5: *How does the amount of personal data stores influence computation time?*

The amount of personal data stores has a strong positive correlation with the computation time, increasing linearly with the amount of PDSs in every secure aggregation step. Computation-time can still be decreased by decreasing the security requirements. This does not influence the convergence rate.

A weak positive correlation in computation time can be seen in the central handler. Influence on the central handler is significantly less big. Calculation time can only be decreased further when using mini-batch updates, but our experiments do not guarantee good convergence under this setting.

> RQ6: *How can we privately optimise other types of models in a federated way while preserving privacy?*

Chapter 7 explains possible ways to optimise gradient descent models such as regularised regression, support vector machines and artificial neural network. It also elaborates how to theoretically optimise decision trees and random forests by the application of federated analytics, i.e. privacy-preserving requests to the data owners.

8.1 Advice for Personal Health Train

The Personal Health Train can successfully connect different Personal Data Stores to train regression models. We propose to use our algorithm, which includes the federated learning algorithm FedSGD, with a learning rate of 0.1 and 800 iterations, and request updates from all available PDSs at once. Use SSO 2.0 (Secure Scaling Operation) to standardise the data stores, which drastically improves convergence-rate and is very efficient. This architecture may be implemented without any privacy concerns, as absolutely no data is leaked as long as the protocols stay intact. That means the receiving party does not have to abide to any of the rules in the General Data Protection Regulation (GDPR) concerning data-privacy as it does not receive any personal data.

Resulting coefficients from a regression model do not quantify the strength of the relation and thus coefficients do not tell the complete story. Models

trained over PDSs data without relationships can significantly deviate from a centrally trained model, but as there are no relations, the resulting coefficients and the regression model do not hold any value. It would be beneficial to create extra implementations to monitor the learning process and analyse whether regression is useful.

8.2 Future Work

As we mentioned, it is interesting to see the effect of stochastically updating the model instead of using all the available PDSs, as this could potentially slow down the convergence when the number of PDSs is extremely large. Stochastic updating may be a possible solution in a situation when there are too many PDSs that it becomes extremely slow to use them all at once. It is interesting to see whether this proposed stochastic method improves convergence on larger datasets and/or will damage the performance or computation-time on smaller datasets.

Future research on federated statistics could calculate confidence intervals for the coefficients privately. Although not a significant problem, the model itself should never leak any private information, i.e. differential privacy. It is interesting to see how this DP reduces final model performance and whether it is suitable for isolated patient data.

Bibliography

[1] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35, 2018.

[2] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Scalable and secure logistic regression via homomorphic encryption. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 142–144, 2016.

[3] Oya Beyan, Ananya Choudhury, Johan van Soest, Oliver Kohlbacher, Lukas Zimmermann, Holger Stenzhorn, Md Rezaul Karim, Michel Dumontier, Stefan Decker, Luiz Olavo Bonino da Silva Santos, et al. Distributed analytics on sensitive medical data: The personal health train. *Data Intelligence*, 2(1-2):96–107, 2020.

[4] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[7] Charlotte Bonte and Frederik Vercauteren. Privacy-preserving logistic regression training. *BMC medical genomics*, 11(4):13–21, 2018.

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

[9] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[10] Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[11] Ken Chang, Niranjan Balachandar, Carson Lam, Darvin Yi, James Brown, Andrew Beers, Bruce Rosen, Daniel L Rubin, and Jayashree Kalpathy-Cramer. Distributed deep learning networks among institutions for medical imaging. *Journal of the American Medical Informatics Association*, 25(8):945–954, 2018.

[12] Massimo Chenal and Qiang Tang. On key recovery attacks against existing somewhat homomorphic encryption schemes. In *International Conference on Cryptology and Information Security in Latin America*, pages 239–258. Springer, 2014.

[13] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 2021.

[14] Davide Chicco and Giuseppe Jurman. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC medical informatics and decision making*, 20(1):16, 2020.

[15] Ananya Choudhury, Johan van Soest, Stuti Nayak, and Andre Dekker. Personal health train on fhir: A privacy preserving federated approach for analyzing fair data in healthcare. In *International Conference on Machine Learning, Image Processing, Network Security and Data Sciences*, pages 85–95. Springer, 2020.

[16] Timo M Deist, Frank JWM Dankers, Priyanka Ojha, M Scott Marshall, Tomas Janssen, Corinne Faivre-Finn, Carlotta Masciocchi, Vincenzo Valentini, Jiazhou Wang, Jiayan Chen, et al. Distributed learning on 20 000+ lung cancer patients–the personal health train. *Radiotherapy and Oncology*, 144:189–200, 2020.

[17] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005.

[18] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[19] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[20] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[21] Benedikt Fecher, Sascha Friesike, and Marcel Hebing. What drives academic data sharing? *PloS one*, 10(2):e0118053, 2015.

[22] Yakov Flaumenhaft and Ofir Ben-Assuli. Personal health records, global policy and regulation review. *Health Policy*, 122(8):815–826, 2018.

[23] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *Proceedings on Privacy Enhancing Technologies*, 2017(4):345–364, 2017.

[24] Gene H Golub, Per Christian Hansen, and Dianne P O'Leary. Tikhonov regularization and total least squares. *SIAM journal on matrix analysis and applications*, 21(1):185–194, 1999.

[25] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[26] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[27] David Kaelber and Eric C Pan. The value of personal health record (phr) systems. In *AMIA Annual Symposium Proceedings*, volume 2008, page 343. American Medical Informatics Association, 2008.

[28] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[29] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics*, 6(2):e19, 2018.

[30] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[31] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[32] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, page 106854, 2020.

[33] Qinbin Li, Zeyi Wen, and Bingsheng He. Practical federated gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4642–4649, 2020.

[34] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[35] Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng, Junbo Zhang, and Yu Zheng. Federated forest. *IEEE Transactions on Big Data*, 2020.

[36] Kalikinkar Mandal and Guang Gong. Privfl: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 57–68, 2019.

[37] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[38] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[39] 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.

[40] Scott Menard. Six approaches to calculating standardized logistic regression coefficients. *The American Statistician*, 58(3):218–223, 2004.

[41] David Meyer and FH Technikum Wien. Support vector machines. *The Interface to libsvm in package e1071*, 28, 2015.

[42] Yongli Mou, Sascha Welten, Yeliz Ucer Yediel, Toralf Kirsten, and Oya Deniz Beyan. Distributed learning for melanoma classification using personal health train. *arXiv preprint arXiv:2103.13226*, 2021.

[43] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348. IEEE, 2013.

[44] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. A performance evaluation of federated learning algorithms. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, pages 1–8, 2018.

[45] Miguel Patrício, José Pereira, Joana Crisóstomo, Paulo Matafome, Manuel Gomes, Raquel Seiça, and Francisco Caramelo. Using resistin, glucose, age and bmi to predict the presence of breast cancer. *BMC cancer*, 18(1):1–8, 2018.

[46] Jason DM Rennie. Regularized logistic regression is strictly convex. *Unpublished manuscript. URL people. csail. mit. edu/jrennie/writing/convexLR. pdf*, 745, 2005.

[47] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

[48] Micah J Sheller, G Anthony Reina, Brandon Edwards, Jason Martin, and Spyridon Bakas. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *International MICCAI Brainlesion Workshop*, pages 92–104. Springer, 2018.

[49] Zhenwei Shi, Ivan Zhovannik, Alberto Traverso, Frank JWM Dankers, Timo M Deist, Petros Kalendralis, René Monshouwer, Johan Bussink, Rianne Fijten, Hugo JWL Aerts, et al. Distributed radiomics as a signature validation study using the personal health train infrastructure. *Scientific data*, 6(1):1–8, 2019.

[50] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[51] Z. Stednick. Medical cost personal datasets. `https://github.com/stedy/Machine-Learning-with-R-datasets`, 2018.

[52] Paul C Tang, Joan S Ash, David W Bates, J Marc Overhage, and Daniel Z Sands. Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption. *Journal of the American Medical Informatics Association*, 13(2):121–126, 2006.

[53] G.S. Thomas. *The Rating Guide to Life in America's Small Cities*. G - Reference, Information and Interdisciplinary Subjects Series. Prometheus Books, 1990.

[54] CH Van Berkel. Multi-core for mobile phones. In *2009 Design, Automation & Test in Europe Conference & Exhibition*, pages 1260–1265. IEEE, 2009.

[55] Johan Van Soest, Chang Sun, Ole Mussmann, Marco Puts, Bob van den Berg, Alexander Malic, Claudia van Oppen, David Townend, Andre Dekker, and Michel Dumontier. Using the personal health train for automated and privacy-preserving analytics on vertically partitioned data. In *MIE*, pages 581–585, 2018.

[56] Xing Wan. Influence of feature scaling on convergence of gradient iterative algorithm. In *Journal of Physics: Conference Series*, volume 1213, page 032021. IOP Publishing, 2019.

[57] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

[58] Ming-Jun Xiao, Liu-Sheng Huang, Yong-Long Luo, and Hong Shen. Privacy preserving id3 algorithm over horizontally partitioned data. In *Sixth international conference on parallel and distributed computing applications and technologies (PDCAT'05)*, pages 239–243. IEEE, 2005.

[59] Kai Yang, Tao Jiang, Yuanming Shi, and Zhi Ding. Federated learning via over-the-air computation. *IEEE Transactions on Wireless Communications*, 19(3):2022–2035, 2020.

[60] Lingchen Zhao, Lihao Ni, Shengshan Hu, Yaniiao Chen, Pan Zhou, Fu Xiao, and Libing Wu. Inprivate digging: Enabling tree-based distributed data mining with differential privacy. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2087–2095. IEEE, 2018.

[61] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Appendix

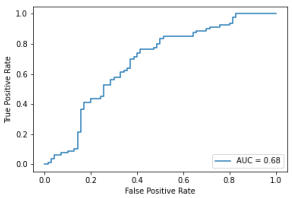
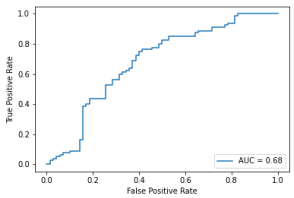
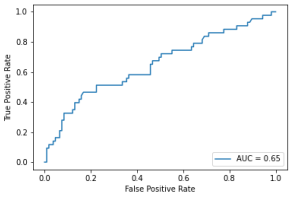
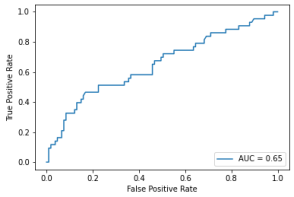
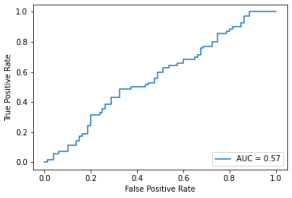
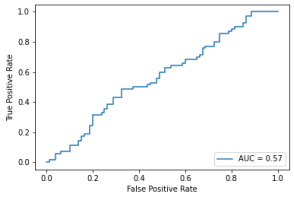
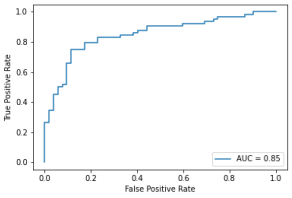
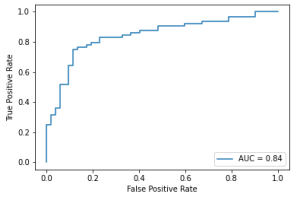
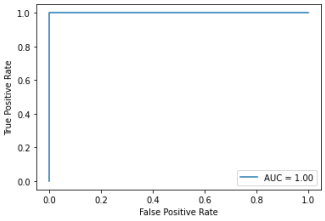
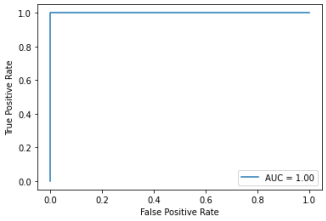
	CentrLogReg	FedLogReg
cardiovasc		
haberman		
readmission		
breast_cancer		
Twoc-150		

Table 8.1: The ROC curves from the corresponding AUC values in table 6.11

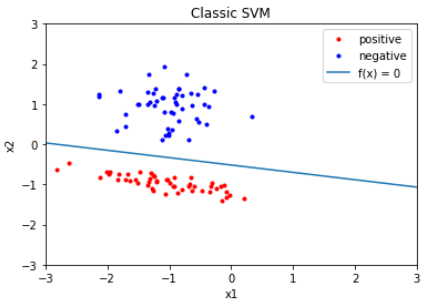
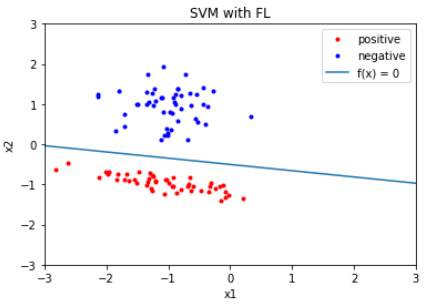
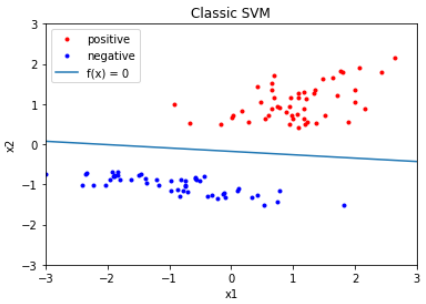
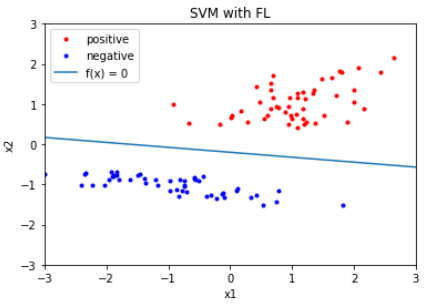
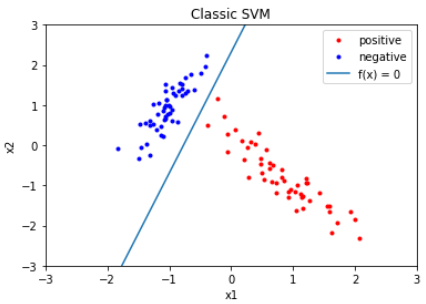
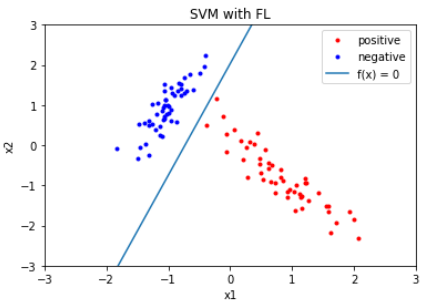
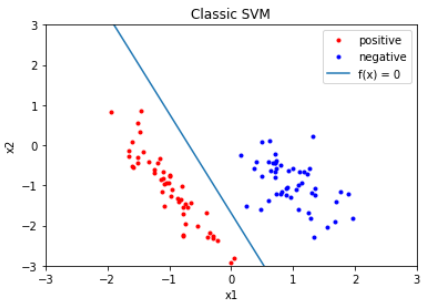
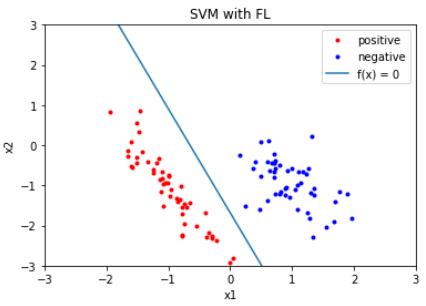
	SVM - no FL	SVM - FL
rnd1		
rnd2		
rnd3		
rnd4		

Table 8.2: Training samples visualised with the trained SVMs; with and without using the federated learning optimization technique; over 4 random datasets