PUBLIC ACCESS

GENOMED4ALL

D6.2
Preliminary conclusions about Federated Learning applied to clinical data

GENOMED4ALL receives funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. 101017549

D6.2

Preliminary conclusions about federated learning applied to clinical data

Revision **v1.0**

Work package	WP6
Task	T6.2
Due date	25-11-2021
Submission date	31-12-2021
Deliverable lead	UPM
Version	v1.0
Authors	Federico Álvarez (UPM), Santiago Zazo (UPM), Juan Parras (UPM), Alejandro Almodóvar (UPM), Patricia Alonso (UPM), Enrico Giampieri (UNIBO), Gastone Castellani (UNIBO), Lorenzo Sani (UNIBO), Cesare Rollo (UNITO), Tiziana Sanavia (UNITO), Anders Krogh (UCPH), Íñigo Prada (UCPH), Alexandros Kanterakis (FORTH), Stelios Sfakianakis (FORTH), and Francesco Cremonesi (Datawizard)
Reviewers	Petros Kountouris (CING), Maria Xenophontos (CING), and Gastone Castellani (UNIBO)

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

Abstract

This report comprises the first contributions from different partners on Federated Learning (FL). After a preliminary introductory section where the fundamental procedures and limitations are described, we detail the well-known mathematical foundation of Federated Learning for convex problems. In this case, we present a key algorithm, Alternating Direction Multipliers Method (ADMM), which is able to implement in a distributed way some fundamental problems such as regression (Ridge and LASSO) and classification (Logistic Regression and Support Vector Machines (SVM)). This procedure shares the fundamental approach of FL which consists of performing some local processing, sharing some intermediate information and updating the local information with some global innovation. In a second step we introduce the extension of this approach to non-convex problems using Bayesian Neural Networks (BNN) where the update is based on the cooperative construction of the posterior of weights from different architectures. Several sections follow where different partners provide different contributions describing our first initiatives on the topic. Some preliminary code from all partners has been uploaded to a common repository to start creating a pool of methods and tools to foster incoming synergies.

Keywords

Federated Learning (FL), convex, non-convex, Alternating Direction Multipliers Method (ADMM), Neural Networks (NNs), Bayesian Neural Networks (BNNs), Machine Learning (ML), Partitioned Variational Inference (PVI), Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), clustering, Cox, Transfer Learning, Ensemble Learning, Myelodysplastic Syndromes (MDS)

Document revision history

Version	Date	Description of change	Contributor(s)
v0.1	02-12-2021	1st version of deliverable	Petros Kountouris (CING) and Maria Xenophontos (CING)
v0.2	09-12-2021	2nd version of deliverable	Gastone Castellani (UNIBO)

Disclaimer

Copyright notice

Project co-funded by the European Commission in the H2020 Programme

Nature of the deliverable	R

Dissemination level

PU Public, fully open. e.g., website	✓
CL Classified information as referred to in Commission Decision 2001/844/EC	
CO Confidential to GENOMED4ALL project and Commission Services	

*** Deliverable types:**

R: document, report (excluding periodic and final reports).
DEM: demonstrator, pilot, prototype, plan designs.
DEC: websites, patent filings, press and media actions, videos, etc.
OTHER: software, technical diagrams, etc.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

Table of contents

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

List of Figures

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

List of Tables

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

Abbreviations

ADMM	Alternating Direction Multipliers Method
ALICE	Adversarially Learned Inference with Conditional Entropy
AML	Acute Myeloid Leukemia
BMB	Bone Marrow Blasts
BNN	Bayesian Neural Networks
BP	Back-Propagation
BYOC	Bring Your Own Concepts
CI	Concordance Index
DEC	Deep Embedding for Clustering
DGD	Deep Generative Decoder
DNN	Deep Neural Networks
DP	Differential Privacy
EFS	Event Free Survival
ELBO	Evidence Lower BOund
EP	Expectation Propagation
FL	Federated Learning
GAN	Generative Adversarial Network
GBDT	Gradient Boosting Decision Tree
GM	Generative Model
GVI	Global Variational Inference
HDP	Hierarchical Dirichlet Process
HE	Homomorphic Encryption
KL	Kullback-Leibler
KLD	Kullback-Leibler Divergence
LDA	Latent Dirichlet Allocation
LFS	Leukemia Free Survival
MAP	Maximum A Posteriori
MDS	Myelodysplastic Syndrome
ML	Machine Learning
MoE	Mixture of Experts
MPC	Multi-Party Computation
NN	Neural Network
PoE	Product of Experts
PVI	Partitioned Variational Inference

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

SAE	Stacked Autoencoder
SAVAE	Survival Analysis Variational Autoencoder
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TEE	Trusted Execution Environments
t-SNE	Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection for dimension reduction
VAE	Variational Autoencoder
ZKP	Zero-Knowledge Proof

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

1 Executive summary

This document has several complementary objectives related to the description of the main contributions on Federated Learning (FL) of all partners involved during the first year of the Genomed4All project. It has different sections that we are going to describe in this summary.

The first chapter introduces the main concepts related to FL which can be summarized as follows: "Federated Learning is a machine learning environment in which multiple entities collaborate in solving a machine learning task, under the coordination of a central server or service provider´´. Each entity stores its raw data locally and does not exchange or transfer it to the others; instead, they send updates to aggregate it immediately so that the learning objective is achieved. These updates are limited in scope, so they consist of the minimum necessary information that can be provided by the submitter entity. This configuration helps with the privacy risks and costs inherent in centralised systems.

These ideas will be highlighted with an extended mathematical description of distributed processing. To do so, we start by presenting the background of distributed convex optimisation using Alternating Direction Multipliers Method (ADMM). This approach is very interesting because, although it is limited to solving convex problems, we start by showing the principles in a centralised implementation and how we can extend the ideas to the distributed version. The procedure is quite general because all federated steps related to local processing, intermediate information exchange, global processing on the server and feedback back to local nodes for iterative updates are explored.

In this document we have also described in full detail the concept of Partitioned Variational Inference as a powerful general methodology to implement Bayesian estimation in a distributed way. This approach allows the optimal computation of densities associated to non-convex architectures as Neural Networks (NN) applied to survival analysis, regression or classification.

Eventually, the main contribution of the paper is the proposal of a common pipeline that marks the main stages in global processing ranging from local feature extraction, preprocessing, feature selection/importance and visualisation to improve interpretability. In this process, imputation and learning procedures (e.g. regression, classification, clustering, and survival analysis) have been particularly highlighted as the main benefits obtained from federated implementation.

In the last section, different partners have presented some preliminary results describing their current initiatives. Their contributions can be viewed and downloaded from a private Gitlab repository where the code has been uploaded.

UPM has proposed an original survival analysis algorithm using variational autoencoders architectures (SAVAE) with competitive results compared to other published work. The main highlight here is the com-

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

bination of generative processes and regression tasks (to be extended to classification and clustering in the future). The motivation for including this latent space is the belief that forcing a consensus on the statistics of this domain can be a very promising means of generating new federated architectures. At the moment, UPM is investigating distributed implementation using modified variational autoencoders that explore a collaborative learning procedure based on the joint design of the *a priori* distribution.

However, initial preliminary results have shown some limited performance which has motivated UPM to also explore alternative implementations to come up with a simple idea: if the problem is the amount and variety of data we have at small local sites, instead of moving real patient information that is forbidden we can share some latent information that allows them to build virtual patients following similar statistics. Currently, they are using generative adversarial networks (GANs) that address the two main challenges of trying to use data from multiple sources: feature mismatching and distribution mismatching.

UNIBO is mainly dedicated to clustering as a technique that is nowadays widely used for patient stratification. In general, patients can be stratified according to some characteristics obtained by multi-omics measurements, like genomic and imaging, and, after this step, try to predict some clinical outcome. However, most clustering methods cannot be directly extended in a federated setting, and even those that can may need substantial changes to adapt to the different scenario.

The model they are developing is based on Deep Embedding for Clustering (DEC) implemented using the Flower framework and extended for the MDS case. This model is inspired by parametric tSNE and exploits the ability of autoencoders to project real data into the feature, or hidden, subspace with (usually) very low dimensionality. They have shown that this model is able to simultaneously learn feature representations and cluster assignments. The resulting model weights were aggregated using the FedAvg algorithm.

UNITO is currently exploring the latest techniques in Neural Networks and Machine Learning to develop non-linear survival analysis models, implementing them through a federated approach that extends Cox proportional hazards, which is probably too simplistic an approach because in real-world clinical data, patient characteristics could contribute through more complicated non-linear functions to the definition of risk scores. To capture this behaviour, the model replaces the linear predictor with the output of a neural network with parameters to optimize.

To this end, three possible federated learning architectures, Sequential Federated, Weights Averaging and Ensemble Federated, have been investigated and tested on an MDS patient survival dataset already described in the previous chapters in two possible scenarios, balanced or unbalanced centres, representing a uniform/nonuniform distribution of patients across centres.

UCPH is working on representation learning combined with transfer learning. The idea of representation learning is to learn a low-dimensional representation of high-dimensional data, which can facilitate the subsequent training of a classifier or a survival model. They have proposed a new approach called Deep Generative Decoder (DGD) as a generative neural network, which consists of a probability distribution over the latent space (or representation space) and a neural network (the decoder), which maps the representations to the

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

feature (output) space.

The model is very similar to the variational autoencoder (VAE) mentioned above, but does not have an encoder. The main difference with the VAE is the Maximum A Posteriori (MAP) estimation instead of the variational inference used in the VAE. They will implement the estimation of the model with missing data as well as the imputation method and test it on real data. Very soon they will start with the implementation of the federated framework and test the different ingredients of the model (missing values, imputation, representation) in the tests developed for the project.

FORTH mainly focuses on Ensemble Learning as an interesting meta-learning strategy in which a large number of relatively weak simple models are combined in order to obtain a stronger ensemble prediction, where the most prominent examples of such ensemble machine learning techniques are random forests. It is known that in a centralised version, new base learners are selected according to their correlation with the negative gradient of the loss function, associated to the whole ensemble.

However, the federated implementation is much more complicated and different approaches are currently being studied based on the idea that the different nodes obtain the optimal split while the central node transmits it to the nodes so that they can continue building the tree. Again, some information about the distribution of the data can be deduced, but important information such as class values is not filtered out. Therefore, after a split, each node can simply transmit these two sums to the central node. The central node adds up all the sums and can calculate a solution update for all the leaves, which it transmits back to the clients. It is important to note that during this federated process all nodes ”work” on building the same set of trees.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

2 Federated Learning. An overview

When speaking about huge amounts of data processing and deep learning algorithms combined, several solutions have been implemented by researchers in the past years to make development easier, since massive processing power and ability to handle different data layers are required. Some of these solutions have gained a lot of interest among developers from over the world providing them with different tools, which enable deep learning applications research and production.

Traditional deep learning tasks involve uploading data to a server and using it to train a model. In other words, due to the complexity of the algorithms chosen and that of the project's task, which requires a big representative collection of samples, traditional ways of processing and training are not enough. In recent years, researchers and developers have been provided with devices being able to have enormous amounts of storage space but it never seems to be enough. It is quite normal too to own data on different devices and having to spend lots of time and power to centralize that data in a single one, which will be used to train the model. It can also become a problem with privacy centralizing personally-identifiable information when it comes to using data obtained from different users. These problems described referring to data quantity and quality cannot be resolved using a traditional way of centralized training machine learning models. This is where Federated Learning appears.

This chapter describes the main vision, along with its characteristics and challenges, of a federated learning environment, identifying the most important limitations and considerations from existing research. It also aims to highlight issues that are of both practical and theoretical interest.

2.1 Definition

This learning approach makes it possible for several algorithms to achieve knowledge and experience from a huge spectrum of information stored in distinct locations, without having to share sensitive data with each other. It was first introduced by [1] in 2016 as an *"unbalanced and non-IID (identically and independently distributed) way of partitioning data across a huge number of unreliable devices with limitations such as bandwidth"*. In other words, the term federated learning was initially introduced referring to mobile or edge device tasks, but an interest in applying it to other applications rapidly increased. For example several organizations or companies collaborating to develop a solution to a problem. These two different federated learning settings are lately classified in "cross-device" and "cross-silo" respectively.

[2] suggests a broad definition for this term which might help with the theoretical understanding of the concept: *"Federated learning is a machine learning setting where multiple entities collaborate in solving a machine learning task, under the coordination of a central server or service provider"*. Each entity stores

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

its raw data locally and does not exchange or transfer it to others; instead, they send updates to aggregate them immediately so that the learning target is achieved. These updates are limited in terms of scope so that they consist on the minimum information needed about the user. This setting helps with the privacy risks and costs that are inherent to centralized systems. Table 2.1 contrasts some other different characteristics given a single centralized setting and both federated learning approaches defined before.

	Single Data center	**Cross-Device**	**Cross-Silo**
Setting	Clients belong to a single cluster (flat dataset)	Clients are organizations (siloed data)	Clients are a massive number of IoT devices
Information Disposal	Stored in a central server so it becomes balanced and accessible	Locally stored and decentralized (cannot read from other clients. Non-IID)	
Composition	Centrally orchestrated	Main server coordinates not seeing raw data	
Availability	Almost always available		Unreliable (Only a number of clients available at a time)
Reliability	Few client failures		Unreliable (5% clients fail)

Table 2.1: Characteristics of single distributed learning vs. federated learning approaches

2.2 Architecture

To put us in the picture of how this setting works, figure 2.1 shows a schematic representation of the learning process in a health field, where the entities are hospital and medical centers, among others. Over the years, medical organizations had to have confidence on their own siloed information even though it could have been biased or have had to pool information from other institutions, with the limitations and consequences that this entails mainly due to privacy. Obviously, algorithms deployed in these scenarios to perform some kind of task need to reach clinical-grade accuracy, and thus to meet this standard the problem should be fed a large and diverse number of cases or information. So, federated learning makes the tasks dispersed by abolishing the need to pool data into a unique location and training in multiple entities using their stored data. The consequent model will be sent after a few iterations to a server (in case of a client-server federated approach), then the global model is updated by aggregating local contributions and finally it is shared back with the participants, as figure 2.1 shows.

As it can be seen, three major components in this system can be differentiated taking into account their functionality: the parties or entities, the manager and the communication-computation framework where the training takes places. Depending on which kind of task and the solution desired, as well as the facilities available, each component will trigger different results and challenges, that will be described in section 2.3.

Talking about the system itself, the federated learning growing claim has gathered some help in the form of frameworks and tools to make its development and simulation easier, dealing with issues that do not arise in the entity research like the efficient processing of the partitioned data. There are several platforms that can be highlighted:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

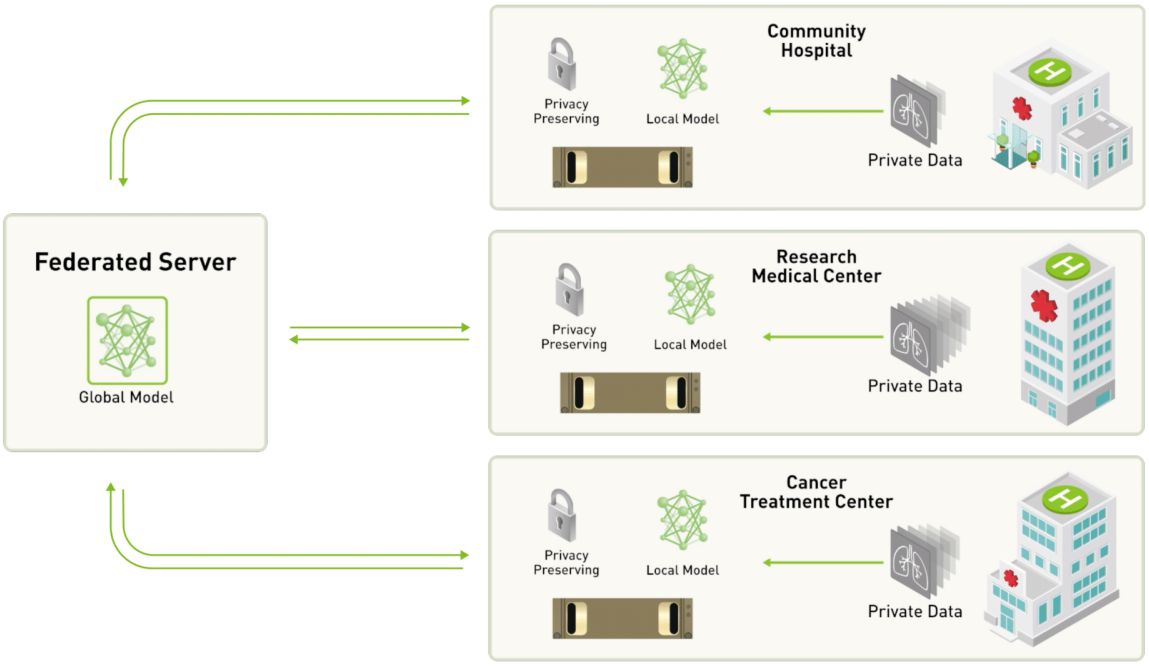


Figure 2.1: Federated learning general representation example [3]

1. **FATE[4]:** Framework which aims to provide services to the entities or organizations. It is structured in six modules that manage different areas, such as secure protocols, storage or the inference between clients. It also provides a visualization tool to track the job execution and performance. There is also detailed documentation but, since it provides interfaces for the algorithm development, the users should change the code to develop their own desired tasks. The code is available in [5].

2. **PaddelFL[6]:** Based on four components in the compile time that include federated learning strategies, models and algorithms with a distributed configuration while training. This setting only includes horizontal algorithms. which uses data from the same feature space across all devices, but it is meant to include vertical ones too (different datasets of different feature space) in the near future. Three main factors can be highlighted in the runtime: two of them coincide with the functionalities of the parties and the manager, and the third one selects who participates in each iteration. Since its development is still at an early stage, there are few and unclear documents. The code is available in [7].

3. **PySyft[8]:** Python library for developing deep learning in a secure way, that provides interfaces to implement the respective algorithm separating private data from the model training by making use of differential privacy and multi-party computation. It works with PyTorch and TensorFlow but only supporting FedAvg[1] too, which is the basic framework where, in each iteration, the updated models from each party are sent to the main server to average the information and get a new global model. However, although it provides tutorials, there is no detailed document. The code is available in [9].

4. **TensorFlow Federated[10]:** It provides construction blocks for federated learning based on Tensor-

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

Flow, including models, data and federated computation constructors (averaging algorithm). Developers are able to make use of the different Python interfaces and define new functionalities too, since it is quite easy to use. It also only supports FedAvg but, in this case, it does not come with privacy technology and it can only be set up on a single device. The code is available in [11].

5. **IBM Federated Learning Framework[12]:** In this case, a proprietary framework is described. IBM research[13] announced at the ICML Federated Learning workshop a framework for federated learning that focuses on enterprise use cases providing an architecture that integrates well with current machine learning libraries like Keras and TensorFlow. It has APIs for algorithm development and secure multi-party computation (SMC) approaches in terms of privacy. The code can be found in [14] and there is some extra information to help with the installation in [15].

6. **NVIDIA Clara [16]:** This is the last proprietary framework described. It is a healthcare framework dedicated for AI-powered imaging, genomics and smart sensors. It implements a centralized scheme having some built-in solutions with privacy mechanisms, which can be improved by developers by using the bring-your-own-concepts (BYOC) method even though it is not an open source platform. It supports GPU, unlike the previous platforms, requiring the support of CUDA 6.0 or higher.

Finally, to guide the federated learning system research, a comparison evaluation makes some sense to review how the setting is working. It requires some kind of metrics, like counting the bytes downloaded or uploaded in each client, the client dropout rate or an statistical evaluation to check the system privacy, for example. LEAF[17] is a benchmark proposed in [18] that provides evaluation capabilities, not including efficiency or privacy ones, but federated data and some reference implementations. It is the only open source benchmark available, although it is not exhaustive enough.

2.3 Challenges

As discussed in the previous section the problem of federated learning can be tackled by a number of solutions, however, FL poses core challenges as exposed in [19]. Thus, accounting for these challenges will only increase the performance of the whole system.

2.3.1 Statistical Heterogeneity

Data variability is a major challenge/factor, inherited from the fact that data are not identically distributed (non-IID) and exist on different distributed clients and are often generated and collected in ways that may not match across the network. The paradigm of data generation adds complexity while analyzing and modeling increasing the likelihood of stragglers. The presence of non-IDD data can be the principal challenge when trying to address efficiency and effectiveness, which can happen due to each entity having data that

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

corresponds to a specific user, a specific time window or location. Therefore, differences in data distribution on each client need to be considered, since it might be a good way to start with the analysis. Real datasets contain a mixture of different non-identical client distributions, such as different features owning the same label or vice versa, and skewed distributions for both feature and labels. Each of those might need a particular mitigation strategy: for example, when there are different labels for the same features, some type of personalized learning could be key to learn the true labeling functions. When dealing with the type of data, recent studies propose in some applications to augment data in order to make it more similar across the different clients or to make a small dataset to be shared (not privacy sensitive). Another natural solution is creating or modifying algorithms changing e.g, the hyperparameters, in order to make the setting more effective.

Also, the convergence behaviour of this kind of settings has to be analyzed, since techniques such as FedAvg have demonstrated to diverge in different practices [1][20]. Recent works like FedProx[20] make a modification to the previous one, FedAvg, to ensure convergence in heterogeneous environments. There exist also heuristic approaches that try to deal with statistical heterogeneity [21], but this would mean sharing local data violating the privacy assumption of federated learning.

But, overall, there is one question that arises if the clients own the capability to train locally: is training a single model the right target? Maybe, it is feasible for each entity to own a model customized for itself turning the problem to a feature of the model. This will also help addressing the changes in client availability. The "multi-model" approach will be addressed in the following sections.

2.3.2 Privacy

In the case of FL model configuration, one of the most important benefits is that it is able to give a level of privacy to the different clients by never letting raw data out of the device and just sending updates of the obtained models from the client to the main server. In other words, the updates do not contain additional user information. However, there is still no guarantee that this updates might not be leaked and, therefore, someone could infer data of the training process by, for example, taking into account a model and the gradient update. For example, [22] shows that one can extract patterns from a network.

Beyond this kind of attacks concerning user privacy, there are some others, such as preventing a model from being learned or biasing the model, that should be taken into account. Reaching the privacy characteristics desired for this setting is going to require using the techniques that are described below but maintaining a cheap computation, efficient communication whilst not compromising accuracy. This section will try to review recent methods that try to preserve the setting-s privacy.

Although there are several definitions for privacy in federated learning, they can be separated into three different groups.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

1. **Secure computations:** The way the learning result is computed and how the information flows during the process need to be considered. In these cases, technologies such as Secure Multi-Party Computation (MPC) and Trusted Execution Environments (TEEs) are used to address the former.

   (a) The first one is a field of cryptography that tries to make the parties compute a particular function using their private inputs but just revealing the output. Recent advances in MPC are attributed to transfer protocols[23] and homomorphic encryption (HE), which allows mathematical operations to be performed on texts encrypted. HE can range from fully homomorphic encryption [24] to more efficient variants [25] (code available in [26]). A review of HE software can be found in [27]. In the federated learning field, a problem occurs when deciding who knows how to decrypt the scheme. An external party or a distributed encryption scheme can be explored as solutions.

   (b) TEEs provide facilities for establishing confidentiality, integrity and measuring that the code has been executed privately. However, it is necessary to take into account other aspects like how to structure the code so that it does not reveal anything about data.

2. **Privacy-Preserving Disclosures:** It is important to check what or how much information about a party is being revealed. The main technique used is differential privacy (DP)[28] that brings uncertainty to the model so that the contribution of any party is hidden. It is quantified by privacy loss parameters (smaller parameters mean increased privacy). Several techniques have been brought over the years for this technique where the server is a trusted implementer of it, although it would be optimal not to rely on a trusted party.

3. **Verifiability:** This problem[29] addresses the capacity of the parties to show that they have performed correctly. In this context, federated learning would allow the main server to show that the different parties worked faithfully and to enable the clients to prove that the server did so also. There are two main technologies to to verify that key functions are running on the orchestration device: remote attestation and zero-knowledge proofs (ZKPs).

   (a) **Remote Attestation[30]:** Might be helpful to verify functions that are key running on the orchestration device. In addition, it may allow a server attest requirements from the entities training.

   (b) **ZKPs:** Cryptographic primitive that enables one entity to prove statements to another, that depend on data known just to the first one not revealing those to the second one. In the recent years, several applications have used this technique motivated by block chains. Yet it remains a challenge in federated learning settings.

2.3.3 Systems Heterogeneity

As it is obvious, in the federated learning network, clients could differ in hardware, connectivity and battery, among others. To handle this heterogeneity recent works suggest three different techniques based on [19] and described in the following:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

1. **Asynchronous communication:** This schemes are attractive to mitigate stragglers in devices that show different characteristics, in particular, in shared-memory systems such as [31]. Nonetheless, this systems rely on the assumption of a limited delay to control the staleness, which is a bit unpractical in federated learning because the delay is not bounded, being this hours or days.

2. **Active device sampling:** Since just a few number of clients engage in each round due to any reason, it could be an approach to actively select which devices participate at each round[32] based on resources, for example. Withal, these technique assume a static system, in other words, a system whose characteristics remain the same.

3. **Fault tolerance:** Finally, since it is not uncommon for some participants to fail at some point during the execution process of a federated system, fault tolerance has been studied extensively in machine learning workloads. There are several techniques such as ignoring the failure (bias) and coded computation, where algorithmic redundancy is introduced [33], but this violates the FL rule of not sharing data across devices.

2.3.4 Communication

Finally, the last but not least important challenge, depending on the system desired, is this key bottleneck: communication. This happens specially in cross-device settings where connectivity may operate at a lower rate than data centers or directly drop. As the previous challenge, there are three directions exposed in recent works to handle this situation, although it remains ambiguous if the cost of communication can be decreased without affecting accuracy in federated learning.

1. **Local Updates:** Recent methods propose increasing competence in communication by updating in a variable way the local models on each device providing a more flexible system[34]. The most commonly used technique is FedAvg, already mentioned (no convergence guarantees).

2. **Compression schemes:** Limited resources (computation and memory) of the devices in the system motivate these. There are several situations such as gradient compression, model broadcast compression and local computation reduction, being them defined in [35]. In the situation of a federated learning approach, recent works show that enforcing a low-rank on the updating models by using Golomb [36] encoding or techniques such as lossy compression and dropout to reduce the communication links [35], could work.

3. **Decentralized training:** Decentralized topologies, where there is only communication between neighbors, have shown that they are faster than centralized ones in low bandwidth networks[37]. Works like [38] have shown that decentralized settings perform well for heterogeneous data and locally updated models however they are limited to linear situations. There are also papers discussing hierarchical communication patterns [39], but they rely on a cloud server to aggregate updates.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

2.4 Conclusion

Over the past few years, federated learning has become an explosive topic of interest, both in research and industry fields, since it enables distributed clients to learn and share a model in a collaborative way whilst keeping the in-out data private. The huge amount of reviews referred to this study show that it is gaining attraction in several disparate areas even though there are several constraints or challenges imposed by the system itself, requiring efficiency and effectiveness, among others. There are many open problems in this approach that are being discussed in several works but not yet solved, showing that this field is in constant growth and researchers or developers have to investigate which characteristics they desire for their settings and, more important, for their type of data and task to be implemented.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

3 Convex optimization. Distributed implementation

This chapter presents the main ideas related to convex optimization. Very briefly we will talk about some well-known convex problems in supervised learning such as Ridge Regression, Lasso, Logistic Regression and SVM formulated from a common perspective emphasizing the effect of function differentiability. We will solve all of them under a common algorithm known as Centralized ADMM because this kind of algorithms is suitable for distributed implementations. Then, in a second section we will describe distributed implementations following the same ADMM principles.

3.1 Brief definition

As [40] defines, a convex optimization problem is one where the functions to minimize $f_0, ..., f_m : \mathbb{R}^n \to \mathbb{R}$ are convex, i.e., satisfy

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \tag{3.1}$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$. This implies at most one global minimum: either there is one optimal solution or no feasible solution to the problem. This is the basic difference between convex and non-convex optimization problems: non-convex ones can have multiple locally optimal points, being time-consuming to identify whether the problem has no solution or it is globally optimal.

3.2 General formulation

The most popular algorithms used in Machine Learning can be presented by a unifying formula:

$$\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \left(\frac{1}{N} \sum_{i=1}^{N} l(w^T v_i + b, y_i) \right) \tag{3.2}$$

where v_i represents the vector inputs while y_i are the outcomes. w, b represent the parameters to be optimized as an affine combination of the input variables. In practice, we will use an equivalent definition compacting $x = [\ w \quad b\ ]^T$ and $u = [\ v \quad 1\ ]^T$. In addition, an extra term $R(x)$ is usually added to include some

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

desired feature of the solution:

$$\min_{x\in\mathbb{R}^{d+1}} \left(\frac{1}{N} \sum_{i=1}^{N} l(x^T u_i, y_i) + R(x) \right) \tag{3.3}$$

We assume that both l and R are convex and x is a $(d+1) \times 1$ vector as the unknown vector variable to be obtained. The first term is referred as *empirical loss function* whereas the second is the *Regularizer*. The loss function typically represents the objective of the machine learning application as regression, classification, features selection, compress sensing and basis pursuit, among others. The regularizer on the other hand adds an extra contribution that penalizes / enforces certain characteristics of the variable x (as its energy or its sparsity) although you may notice that some bias is included. Typically this term is a *norm* as L1 or L2.

The Ridge for instance is given by the linear regression formulation with an L2 norm

$$\min_{x\in\mathbb{R}^{d+1}} \frac{1}{N} \sum_{i=1}^{N} (x^T u_i - y_i)^2 + \frac{\lambda}{2} \|x\|_2^2 \tag{3.4}$$

while the Lasso uses an L1 norm

$$\min_{x\in\mathbb{R}^{d+1}} \left(\frac{1}{N} \sum_{i=1}^{N} (x^T u_i - y_i)^2 + \lambda \|x\|_1 \right) \tag{3.5}$$

Regarding classification, the most popular algorithms are Logistic regression

$$\min_{x\in\mathbb{R}^{d+1}} \left(\frac{1}{N} \sum_{i=1}^{N} \ln(1 + \exp(-y_i(x^T u_i))) + \frac{\lambda}{2} \|x\|_2^2 \right) \tag{3.6}$$

and SVMs

$$\min_{x\in\mathbb{R}^{d+1}} \left(\frac{1}{N} \sum_{i=1}^{N} \max(1 - y_i(x^T u_i), 0) + \frac{\lambda}{2} \|x\|_2^2 \right) \tag{3.7}$$

If the functions are differentiable, standard procedures are based on iterative gradient (first-order or quasi-Newton) algorithms. However, it can be observed that Lasso (and any implementation of L1) and SVM are not differentiable and this fact complicates the analysis. Note that, in general, subgradient methods produce undesirable oscillations and that, alternatively, the solution is obtained by proximal algorithms.

Among the variety of convex optimisation solutions we will describe only the ADMM which scales very well and is therefore very attractive for big data scenarios. In fact, it can be implemented in a fully distributed

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

way making it a strong candidate for solving convex problems in non-centralised scenarios. Let's start with the centralised case to understand the main ideas and then we will consider the distributed case.

3.3 Centralized ADMM

The following ideas are reflected from [41]. We start from a convex optimization problem with equality constraints

$$\begin{aligned} &\min f(x) \\ &\text{s.t.: } Ax = b \end{aligned} \tag{3.8}$$

The Lagrangian for that problem is

$$L(x,y) = f(x) + y^T(Ax - b) \tag{3.9}$$

To guarantee convergence without assuming strict convexity of f we define the augmented Lagrangian:

$$L_\rho(x,y) = f(x) + y^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2 \tag{3.10}$$

where ρ is called the penalty parameter. Note that including the term $\|Ax - b\|_2^2$ guarantees strong convexity improving the convergence characteristics. This problem is clearly equivalent to the original one and can be solved by the dual ascent method which is known in this case as the method of multipliers where the penalty parameter is used as the fixed step size.

$$\begin{aligned} x_{k+1} &= \arg\min_x L_\rho(x, y_k) \\ y_{k+1} &= y_k + \rho(Ax_{k+1} - b) \end{aligned} \tag{3.11}$$

because this choice of ρ guarantees the convergence of the primal residual to 0.

The ADMM algorithm is strongly based on this simple procedure, although in its more general perspective it solves problems in the form:

$$\begin{aligned} &\min g(x) + h(z) \\ &\text{s.t.: } Ax + Bz = c \end{aligned} \tag{3.12}$$

where g and h are convex and a new variable z is proposed. The augmented Lagrangian is:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$L_\rho(x,z,y) = g(x) + h(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \tag{3.13}$$

and ADMM consists of the iterations:

$$\begin{aligned} x_{k+1} &= \arg\min_x L_\rho(x, z_k, y_k) \\ z_{k+1} &= \arg\min_z L_\rho(x_{k+1}, z, y_k) \\ y_{k+1} &= y_k + \rho(Ax_{k+1} + Bz_{k+1} - c) \end{aligned} \tag{3.14}$$

Although (x,z) can be updated simultaneously, ADMM proposes a sequential form in alternating directions. This is precisely what allows the decomposition when g or h are separable. These equations can be written more adequately by combining the linear and quadratic terms in the augmented Lagrangian and scaling the dual variable. Defining the residual $r = Ax + Bz - c$, we have:

$$y^T r + \frac{\rho}{2}\|r\|_2^2 = \frac{\rho}{2}\left\|r + \frac{1}{\rho}y\right\|_2^2 - \frac{1}{2\rho}\|y\|_2^2 = \frac{\rho}{2}\|r + w\|_2^2 - \frac{\rho}{2}\|w\|_2^2 \tag{3.15}$$

where $w = \frac{1}{\rho}y$ is the scaled dual variable. Finally, ADMM looks like:

$$\begin{aligned} x_{k+1} &= \arg\min_x (g(x) + \tfrac{\rho}{2}\|Ax + Bz_k - c + w_k\|_2^2) \\ z_{k+1} &= \arg\min_z (h(z) + \tfrac{\rho}{2}\|Ax_{k+1} + Bz - c + w_k\|_2^2) \\ w_{k+1} &= w_k + Ax_{k+1} + Bz_{k+1} - c \end{aligned} \tag{3.16}$$

If g and h are closed, proper and convex and strong duality is satisfied, the ADMM satisfies convergence guarantees.

Let us recall the original problem that we have proposed in terms of the loss function and the regularizer:

$$\min_x f(x) = \min_x (g(x) + h(x)) \tag{3.17}$$

It can be seen that we can express it in terms of the ADMM formulation very easily

$$\begin{aligned} &\min_{x,z} (g(x) + h(z)) \\ &s.t. : x - z = 0 \end{aligned} \tag{3.18}$$

If we particularise the equation in the above optimization problem we get

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$\begin{aligned} x_{k+1} &= \arg\min_x(g(x) + \tfrac{\rho}{2}\|x - z_k + w_k\|_2^2) \\ z_{k+1} &= \arg\min_z(h(z) + \tfrac{\rho}{2}\|x_{k+1} - z + w_k\|_2^2) \\ w_{k+1} &= w_k + x_{k+1} - z_{k+1} \end{aligned} \tag{3.19}$$

which are the equations to be implemented.

3.3.1 Formulation of ADMM Lasso: an example

As we have seen, ADMM solves problems in the general form:

$$\begin{aligned} &\min g(x) + h(z) \\ &\text{s.t.: } Ax + Bz = c \end{aligned} \tag{3.20}$$

where g and h are convex and a new variable z is proposed. In our case, we want to solve the Lasso problem:

$$\begin{aligned} &\min(\tfrac{1}{N}\|Ax - b\|_2^2 + \lambda\|z\|_1) \\ &\text{s.t.: } x - z = 0 \end{aligned} \tag{3.21}$$

where $g(x) = \frac{1}{N}\|Ax - b\|_2^2 \quad h(z) = \lambda\|z\|_1$. The augmented Lagrangian is:

$$L_\rho(x, z, y) = \frac{1}{N}\|Ax - b\|_2^2 + \lambda\|z\|_1 + y^T(x - z) + \frac{\rho}{2}\|x - z\|_2^2 \tag{3.22}$$

After some manipulations and using the scaled dual variable, ADMM consists of the iterations:

$$\begin{aligned} x_{k+1} &= \arg\min_x(L_\rho(x, z_k, w_k)) = \left(\tfrac{2}{N}A^TA + \rho I\right)^{-1}\left(\tfrac{2}{N}A^Tb - \rho(z_k - w_k)\right) \\ z_{k+1} &= \arg\min_z(L_\rho(x_{k+1}, z, w_k)) = \text{Soft}\left(x_{k+1} + w_k, \tfrac{\lambda}{\rho}\right) \\ w_{k+1} &= w_k + x_{k+1} - z_{k+1} \end{aligned} \tag{3.23}$$

where soft refers to the *Soft Thresholding* operation:

$$\text{soft}(a, b) = (a - b)_+ - (-a - b)_+ \tag{3.24}$$

where $(\cdot)_+ = \max\{0, \cdot\}$.

A sketch of ADMM can be seen in Algorithm 1.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

Algorithm 1 Centralized ADMM Lasso algorithm

1: We are given A, b, λ and ρ.
2: Initialize x_1, z_1 and w_1 as a zero vectors.
3: **for** k=1: Number of iters **do**
4: Obtain x_{k+1} using the first expression from (3.23).
5: Obtain z_{k+1} using the second expression from (3.23).
6: Obtain w_{k+1} using the third expression from (3.23).
7: **end for**

3.3.2 Code guide

The provided implementation of ADMM is tested on Python 3.6. There are two scripts, where the first one is «admm_toolbox.py» which implements the code needed to run a centralized (or distributed) ADMM based version of two classification problems (SVM and logistic regression), and two regression problems (LASSO and ridge). The second script, «admm.py», contains an example on how to run each problem. Also, note that we include two example datasets: the housing dataset for regression purposes and the banknote dataset for classification.

To run the problems, the following steps should be followed:

1. First of all, a Python 3.6+ environment is required.
2. The environment requirements should be installed. To do so, PIP is a package management system used to install and manage software packages written in Python:
   (a) *pip install -r requirements.txt*
3. Note that a required package is CVX optimization toolbox, that is only required to evaluate how good the solution given by ADMM is. The toolbox could be modified to not use this package, as it is only needed for visualization and debugging purposes.
4. Run the main script by using *python admm.py.* Note that this will solve all problems in a distributed and centralized way, and show the results through the screen.

You can also run your own script for your concrete problem. In order to do so, prepare a python script that contains the following commands:

1. Load the ADMM solver using *cent_solver = AdmmCentralized(data_in, data_out, problem)*, where data_in denotes the dataset features, data_out the regression target or labels, and problem is a string "ridge", "lasso", "svm" or "logistic" depending on the problem we want to solve. This function creates an instance of the AdmmCentralized class, which is used to optimize. Note that this class implements three functions x_update, y_update and z_update that correspond to equations 3.23.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

2. We can train by using *cent_solver.train(niter)*, where niter is the number of iterations of the solver. The *train()* method calls the different update methods implemented in the class.

3. The results can be seen by using *cent_solver.plot()*

Note that we provide four example problems for illustration purposes: the code is commented so that it facilitates incorporating other problems to the toolbox.

3.4 Distributed ADMM

Let us consider a global problem $f(x)$ that can be split into a set of local separable problems $g_i(x)$ (for simplicity we currently skip the regularizer effect to be included later):

$$\min_x f(x) = \min_x \sum_{i=1}^{N_b} g_i(x) \tag{3.25}$$

where $x \in \mathbb{R}^{d+1}$ and $g_i : \mathbb{R}^{d+1} \rightarrow R \cup \{+\infty\}$ are convex. The goal now is to solve this problem in such a way that each term can be handled by its own processing element. This problem can be rewritten with local variables $x_i \in \mathbb{R}^{d+1}$ and a common global variable z :

$$\min_{x_i} \sum_{i=1}^{N_b} g_i(x_i) \text{s.t.:} x_i - z = 0 \quad i = 1, \cdots, N_b \tag{3.26}$$

This formulation is called as the *Global Consensus Problem*, since the constraint is that all the local variables should agree. ADMM can be derived directly for an arbitrary number of nodes N_b including the corresponding Lagrange multipliers y_i and the quadratic term to ensure strong convexity:

$$L_\rho(x_1, \cdots, x_{N_b}, z, y_1, \cdots y_{N_b}) = \sum_{i=1}^{N_b} \left(g_i(x_i) + y_i^T (x_i - z) + \frac{\rho}{2} \|x_i - z\|_2^2 \right) \tag{3.27}$$

Making a similar development as in the centralized ADMM described in previous section, we obtain the following algorithm:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$
\begin{aligned}
x_{i,k+1} &= \arg\min_{x_i} \left(g_i(x_i) + y_{i,k}^T (x_i - z_k) + \frac{\rho}{2} \|x_i - z_k\|_2^2 \right) \\
z_{k+1} &= \frac{1}{N_b} \sum_{i=1}^{N_b} \left(x_{i,k+1} + \frac{1}{\rho} y_{i,k} \right) \\
y_{i,k+1} &= y_{i,k} + \rho (x_{i,k+1} - z_{k+1})
\end{aligned}
\tag{3.28}
$$

Let us now show the case where the regularization effect has to be included in the analysis:

$$
\min_{x,z} \sum_{i=1}^{N_b} g_i(x_i) + h(z) \text{s.t.:} x_i - z = 0 \quad i = 1, \cdots, N_b \tag{3.29}
$$

where N_b again refers to the number of blocks we are splitting our data.

The resulting ADMM algorithm is:

$$
\begin{aligned}
x_{i,k+1} &= \arg\min_{x_i} \left(g_i(x_i) + y_{i,k}^T (x_i - z_k) + \frac{\rho}{2} \|x_i - z_k\|_2^2 \right) \\
z_{k+1} &= \arg\min_{z} \left(h(z) + \sum_{i=1}^{N_b} (-y_{i,k}^T z + \frac{\rho}{2} \|x_{i,k+1} - z\|_2^2) \right) \\
y_{i,k+1} &= y_{i,k} + \rho (x_{i,k+1} - z_{k+1})
\end{aligned}
\tag{3.30}
$$

In many cases it is simpler and easier to work with the scaled form (it is the same procedure as in previous section). Defining the residual $r = (x_i - z_k)$, we have that the two last terms in previous equations become:

$$
y_{i,k}^T r + \frac{\rho}{2} \|r\|_2^2 = \frac{\rho}{2} \left\| r + \frac{1}{\rho} y_{i,k} \right\|_2^2 - \frac{1}{2\rho} \|y_{i,k}\|_2^2 = \frac{\rho}{2} \|r + w_{i,k}\|_2^2 - \frac{\rho}{2} \|w_{i,k}\|_2^2 \tag{3.31}
$$

where $w_{i,k} = \frac{1}{\rho} y_{i,k}$ is the scaled dual variable. ADMM finally looks like:

$$
\begin{aligned}
x_{i,k+1} &= \arg\min_{x_i} (g_i(x_i) + \tfrac{\rho}{2} \|x_i - z_k + w_{i,k}\|_2^2) \\
z_{k+1} &= \arg\min_{z} (h(z) + \tfrac{N_b \rho}{2} \|z - \bar{x}_{k+1} - \bar{w}_k\|_2^2) \\
w_{i,k+1} &= w_{i,k} + x_{i,k+1} - z_{k+1}
\end{aligned}
\tag{3.32}
$$

where $\bar{x}_k = \frac{1}{N_b} \sum_{i=1}^{N_b} x_{i,k}$ and $\bar{w}_k = \frac{1}{N_b} \sum_{i=1}^{N_b} w_{i,k}$.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

These are the equations to be implemented. Note that $x_{i,k+1}$ and $w_{i,k+1}$ are local variables that are computed on isolated nodes and shared as intermediate variables with the server that calculates the global variable z_{k+1} and shares it again with all nodes.

3.4.1 Formulation Formulation of ADMM SVM: an example

As we have seen, distributed ADMM can be used to solve problems in the following way:

$$\begin{aligned} \min_{x,z} \sum_{j=1}^{N_b} g_j(x_j) + h(z) \\ \text{s.t.: } x_j - z = 0 \quad j = 1, \cdots, N_b \end{aligned} \tag{3.33}$$

where N_b is the number of workers, indexed by j. Note that we assume that each worker has its own set of points A_j,b_j. In our case, we want to solve the next problem as a distributed SVM classifier:

$$\begin{aligned} \min_{x,z} \sum_{j=1}^{N_b} \frac{1}{N} \sum_{i=1}^{N} \max\left(1 - b_{j,i} a_{j,i}^T x_j, 0\right) + \frac{\lambda}{2}\|z\|_2^2 \\ \text{s.t.: } x_j - z = 0 \quad j = 1, \cdots, N_b \end{aligned} \tag{3.34}$$

assuming that we have i samples at every node. The augmented Lagrangian is:

$$\begin{aligned} L_\rho(x_j, z, y_j) = \sum_{j=1}^{N_b} \left(\frac{1}{N} \sum_{i=1}^{N} \max\left(1 - b_{j,i} a_{j,i}^T x_j, 0\right) \right) + \frac{\lambda}{2}\|z\|_2^2 \\ + \sum_{j=1}^{N_b} y_j^T (x_j - z) + \sum_{j=1}^{N_b} \frac{\rho}{2}\|x_j - z\|_2^2 \end{aligned} \tag{3.35}$$

and ADMM consists of the iterations:

$$\begin{aligned} x_{j,k+1} &= \arg\min_{x_j}(L_\rho(x_j, z_k, y_{j,k})) \\ z_{k+1} &= \arg\min_{z}(L_\rho(x_{j,k+1}, z, y_{j,k})) \\ w_{j,k+1} &= w_{j,k} + x_{j,k+1} - z_{k+1} \end{aligned} \tag{3.36}$$

where w is y scaled by ρ. In this case, we can easily obtain the subgradient of L_ρ, and use it to update x_j : this subgradient is:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$\delta(x_j) = \rho(x_j - z + w_j) + \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 0 & if\ b_{j,i}a_{j,i}^T x_j > 1 \\ -b_{j,i}a_{j,i} & if\ b_{j,i}a_{j,i}^T x_j < 1 \end{cases} \tag{3.37}$$

And the update equation of subgradient descent method was:

$$x_{j,k+1} = x_{j,k} - \mu\delta(x_{j,k}) \tag{3.38}$$

where μ is the step size. Regarding the update of the z term, we can derivate the Lagrangian to obtain the following expression:

$$\frac{\partial L_\rho(x_j, z, w_j)}{\partial z} = \lambda z - \sum_{j=1}^{N_b}\rho(x_j - z + w_j) = z\,(\lambda + N_b\rho) - \rho\sum_{j=1}^{N_b}(x_j + w_j) = 0 \tag{3.39}$$

Therefore, we have that:

$$z_{k+1} = \rho\frac{\sum_{j=1}^{N_b} x_{j,k+1} + w_{j,k}}{\lambda + N_b\rho} \tag{3.40}$$

Thus, note that in this case, we update x_j locally using subgradient steps, then update z globally using the information from all workers, and then update w_j locally. A sketch of ADMM can be seen in Algorithm 2.

Algorithm 2 Distributed ADMM SVM algorithm

```
Each worker knows A_j, b_j, λ, ρ, μ and N_b.
Initialize x_{j,1}, z_1 and w_{j,1} as a zero vectors for each worker j = 1, .., N_b.
for k=1:Number of iters do
    for j = 1 : N_b do
        for Number of subgradient steps do
            Obtain the local subgradient δ(x_j) using (3.37).
            Update x_{j,k+1} using (3.38).
        end for
    end for
    Obtain z_{k+1} using (3.40).
    for j = 1 : N_b do
        Update w_{j,k+1} using the third expression from (3.23).
    end for
end for
```

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

3.4.2 Code guide

We can use the same code described in the centralized ADMM case for the distributed ADMM problem. We use the same two scripts as in the centralized case, where the first one is «admm_toolbox.py» which implements the code needed to run a centralized and distributed ADMM based version of two classification problems (SVM and logistic regression), and two regression problems (LASSO and ridge). The second script, «admm.py», contains an example on how to run each problem. By running the main script by using *python admm.py,* the program solves all problems in a distributed and centralized way, and shows the results through the screen.

You can also run your own script for your concrete problem. In order to do so, prepare a python script that contains the following commands:

1. Load the ADMM solver using *dist_solver = AdmmDistributed(data_in, data_out, problem)*, where data_in denotes the dataset features, data_out the regression target or labels, and problem is a string "ridge", "lasso", "svm" or "logistic" depending on the problem we want to solve. Note that in this case, data_in and data_out must be lists of elements, where each element in the list contains the private dataset for each node; also note that the number of nodes is automatically set to the number of private datasets. This function creates an instance of the AdmmDistributed class, which is used to optimize. Note that this class implements three functions x_update, y_update and z_update that correspond to equations 3.23.
2. We can train by using *dist_solver.train(niter)*, where niter is the number of iterations of the solver. The *train()* method calls the different update methods implemented in the class.
3. The results can be seen by using *dist_solver.plot()*

Note that we provide four example problems for illustration purposes: the code is commented so that it facilitates incorporating other problems to the toolbox.

3.5 Federated learning interpretation

Let us have a look at the previous main equations now from a more illustrative perspective. As we have seen, the centralized ADMM can be summarized by the following equations:

$$\begin{aligned} x_{k+1} &= \arg\min_x (g(x) + \tfrac{\rho}{2}\|x - z_k + w_k\|_2^2) \\ z_{k+1} &= \arg\min_z (h(z) + \tfrac{\rho}{2}\|x_{k+1} - z + w_k\|_2^2) \\ w_{k+1} &= w_k + x_{k+1} - z_{k+1} \end{aligned} \tag{3.41}$$

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

where we can have analytical or gradient based solutions depending on the expression of $g(x)$ or $h(z)$. If we denote all these variables with a common notation:

$$\theta_k = \begin{bmatrix} x_k \\ z_k \\ w_k \end{bmatrix} \tag{3.42}$$

ADMM represents an iterative (coordinate descent) algorithm with convergence guarantees:

$$\theta^* = \lim_{k\to\infty} \theta_k \tag{3.43}$$

If we now revisit the distributed algorithm, we obtain:

$$\begin{aligned} x_{i,k+1} &= \arg\min_{x_i}(g_i(x_i) + \tfrac{\rho}{2}\|x_i - z_k + w_{i,k}\|_2^2) \\ z_{k+1} &= \arg\min_{z}(h(z) + \tfrac{N_b\rho}{2}\|z - \bar{x}_{k+1} - \bar{w}_k\|_2^2) \\ w_{i,k+1} &= w_{i,k} + x_{i,k+1} - z_{k+1} \end{aligned} \tag{3.44}$$

where $\bar{x}_k = \frac{1}{N_b}\sum_{i=1}^{N_b} x_{i,k}$ and $\bar{w}_k = \frac{1}{N_b}\sum_{i=1}^{N_b} w_{i,k}$. Following the same notation as before, we have that at each node we have

$$\theta_{i,k} = \begin{bmatrix} x_{i,k} \\ z_k \\ w_{i,k} \end{bmatrix} \tag{3.45}$$

where some components are obtained locally, $(x_{i,k}, w_{i,k})$ while the other is obtained in a centralized form z_k in order to guarantee that the individual optimization problems provide the same solution as the centralized one:

$$\theta^* = \lim_{k\to\infty} \theta_{i,k}, \quad \forall i \tag{3.46}$$

It is also interesting to mention that we only share the part of the local model $(x_{i,k}, w_{i,k})$ but the data remain private. Schematically, we have:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

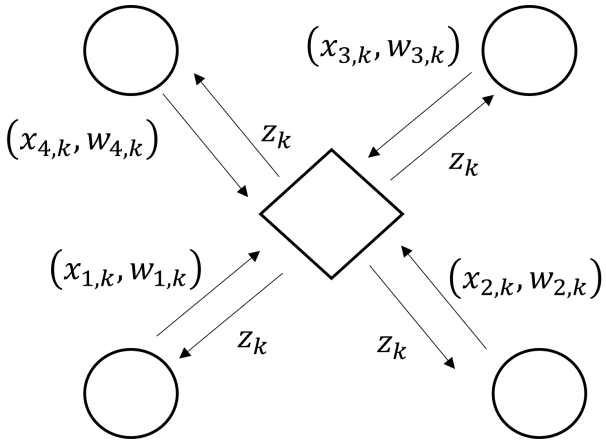


Figure 3.1: Federated learning ADMM distribution (convex optimization).

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

4 Non-Convex optimization. Distributed implementation

4.1 Motivation

The main conclusion from the previous section is that for convex problems, ADMM is a very efficient procedure to implement federated architectures. As the cost function is forced to be strongly convex, convergence is usually very fast. In this chapter we will try to extend these ideas to non-convex optimization scenarios. The motivation is that previous methodologies are limited to linear architectures that in most of the cases are not able to adequately express the statistical dependencies of complex data, such as in medical records, to satisfactorily perform regression or classification requests. In this sense, we know that neural networks (NNs) are universal approximators meaning that they are able to approximate any function in deep architectures with an adequate number of neurons. However, we also know that cost optimizations become highly non-convex. Therefore, if we want to use NNs we have to deal with non-convex optimization principles. The main problem to consider so far is that simply averaging the updates of the weights of the NNs at different locations does not work because, as they are driven by different data, gradient Back-Propagation (BP) updates will put the coefficients in quite different states and therefore any operation on these values will not guarantee the desired behaviour.

It is clear that we have to follow a different perspective which is going to be based on a random description of the weights. This formulation is very suitable because this way all users will contribute to the expression of the joint distribution according to the available data. These architectures are known as Bayesian Neural Networks and typically have to be solved using approximations. We are fortunate because, as we will see, these approximations can also be built in a distributed way. We are going to focus on the discriminative model where we have some data and also we have labels that can be real variables (estimation) or categorical (classification).

The main point is that now our derivations will be motivated by the calculation of the posterior predictive distribution. Assuming that we have the training data $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$ where $\{x_n\}_{n=1}^N$ correspond to the input data and $\{y_n\}_{n=1}^N$ will be the labels, and that we want to characterize a new sample y^* for a certain input x^*, we need to evaluate:

$$p\left(y^* \mid \mathcal{D}, x^*\right) \tag{4.1}$$

Notice that the result of this analysis is not a particular value but a distribution that can be continuous (regression) or discrete (classification). This fact is very important because the mode will provide the highest

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

probability value (the mean will also be very informative) but also we have the convariance matrix that will provide information about the concentration of the probability or in other words, how much we can trust our result.

Using standard probability principles, the posterior predictive distribution can be expressed as:

$$p\left(y^* \mid \mathcal{D}, x^*\right) = \int p\left(y^* \mid x^*, w\right) p\left(w \mid \mathcal{D}\right) dw \tag{4.2}$$

where w represents the model variables that transform $\{x_n\}_{n=1}^N$ into $\{y_n\}_{n=1}^N$ and also $x^* \rightarrow y^*$. As we have already said, w are treated as random variables with a certain prior $p\left(w\right)$. You can immediately recognize the posterior $p\left(w \mid \mathcal{D}\right)$ and the likelihood $p\left(y^* \mid x^*, w\right)$. The posterior can be further derived as:

$$p\left(w \mid \mathcal{D}\right) = \frac{p\left(w\right) p\left(y \mid w, x\right)}{p\left(y \mid x\right)} \tag{4.3}$$

where we have used the shorthand notation:

$$x = \{x_n\}_{n=1}^N, \;\; y = \{y_n\}_{n=1}^N \tag{4.4}$$

Typically, the likelihood $p\left(y \mid x, w\right)$ is well known and expressed even analytically. On the other hand, usually the posterior $p\left(w \mid \mathcal{D}\right)$ is intractable mainly due to the denominator.

The most standard approach to calculate the posterior is based on the Maximum A Posteriori (MAP) criteria where:

$$w_{MAP} = \arg\max_w \ln p\left(w \mid \mathcal{D}\right) = \arg\max_w \left(\ln p\left(y \mid w, x\right) + \ln p\left(w\right)\right) \tag{4.5}$$

approximating

$$p\left(w \mid \mathcal{D}\right) \approx \delta\left(w - w_{MAP}\right) \tag{4.6}$$

The above optimisation problem can be solved analytically (not very often), but mostly using gradient techniques.

Under this analysis, the predictive distribution simplifies:

$$\begin{aligned} p\left(y^* \mid \mathcal{D}, x^*\right) = \int p\left(y^* \mid x^*, w\right) p\left(w \mid \mathcal{D}\right) dw = \\ = \int p\left(y^* \mid x^*, w\right) \delta\left(w - w_{MAP}\right) dw = p\left(y^* \mid x^*, w_{MAP}\right) \end{aligned} \tag{4.7}$$

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

In high dimensional problems with arbitrary distributions this approximation is unfeasible and we have to resort to some approximations based on Variational Inference principles. As we will see, this approach will allow us to find a distributed implementation.

4.2 A Centralized approach.

4.2.1 Theoretical derivations

Variational inference intends to find an approximate (variational) distribution $q(w)$ such that

$$p(w \mid \mathcal{D}) \approx q(w) \tag{4.8}$$

with some amenable structure (it will usually be parameterized). For discriminative configuration, the posterior can be expressed as in terms of the individual likelihoods:

$$p(w \mid \mathcal{D}) = \frac{p(w)\, p(y \mid w, x)}{p(y \mid x)} = \frac{p(w) \prod_{n=1}^{N} p(y_n \mid w, x_n)}{p(y \mid x)} \tag{4.9}$$

If we proceed with the expression of the log marginal likelihood

$$\ln p(y \mid x) = \ln \int p(w)\, p(y \mid w, x)\, dw \tag{4.10}$$

that can be properly bounded

$$\ln p(y \mid x) = \ln \int p(w)\, p(y \mid w, x)\, dw \geq \int q(w) \ln \frac{p(w)\, p(y \mid w, x)}{q(w)} dw \tag{4.11}$$

where

$$\mathcal{F}_{GVI}(q(w)) = \int q(w) \ln \frac{p(w)\, p(y \mid w, x)}{q(w)} dw \tag{4.12}$$

is the Evidence Lower BOund (ELBO) and notation Global Variational Inference (GVI) is convenient to compare with further results on distributed implementations. Note that if we maximize the ELBO, we are also minimizing the Kullback-Leibler (KL) divergence with the posterior:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$\mathcal{F}_{GVI}\left(q\left(w\right)\right) = \int q\left(w\right) \ln \frac{p\left(w\right) p\left(y \mid w, x\right)}{q\left(w\right)} dw = -KL\left(q\left(w\right) \mid\mid p\left(w \mid \mathcal{D}\right)\right) \tag{4.13}$$

The ELBO can be expanded in a more interpretable form:

$$\mathcal{F}_{GVI}\left(q\left(w\right)\right) = -KL\left(q\left(w\right) \mid\mid p\left(w\right)\right) + \sum_{n=1}^{N} \int q\left(w\right) \ln p\left(y_n \mid w, x_n\right) dw \tag{4.14}$$

where the variational problem is usually stated as:

$$q^*\left(w\right) = \arg\max_{q(w)} \mathcal{F}_{GVI}\left(q\left(w\right)\right) \tag{4.15}$$

In practice, the variational distributions often parameterized in a simple way to facilitate the analysis. For instance, we can assume the Mean Field approach and Gaussian model for each independent component:

$$q\left(w\right) \approx \prod_{i=1}^{N} q_{\theta_i}\left(w_i\right) = \prod_{i=1}^{N} \mathcal{N}\left(w_i; \mu_i, \sigma_i^2\right) \tag{4.16}$$

where $\theta_i = \left\{\mu_i, \sigma_i^2\right\}$.

So, the idea is that instead of calculating the posterior $p\left(w \mid \mathcal{D}\right)$ to generate the weights, we will use a surrogate distribution (the variational approximation). Note that the calculation of this surrogate function has been posed as an optimization problem that usually we will present in terms of the adjustment of certain θ parameters:

$$q\left(w\right) \rightarrow q_\theta\left(w\right) \tag{4.17}$$

4.2.2 Practical implementations

Solving this problem typically requires finding condition

$$\nabla_\theta \mathcal{F}\left(q_\theta\left(w\right)\right) = 0 \tag{4.18}$$

where

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$\begin{aligned} \mathcal{F}\left(q_\theta\left(w\right)\right) &= \int q_\theta\left(w\right) \ln \frac{p\left(w\right) p\left(y \mid w, x\right)}{q_\theta\left(w\right)} dw = \\ &= \int q_\theta\left(w\right) \ln p\left(w\right) p\left(y \mid w, x\right) dw - \int q_\theta\left(w\right) \ln q_\theta\left(w\right) dw \end{aligned} \tag{4.19}$$

can be expressed in an alternative way using expectations

$$\mathcal{F}\left(q_\theta\left(w\right)\right) = E_{q_\theta(w)}\left[\ln p\left(w\right) p\left(y \mid w, x\right) - \ln q_\theta\left(w\right)\right] = E_{q_\theta(w)}\left[f\left(w, \theta\right)\right] \tag{4.20}$$

where $f\left(w, \theta\right) = \ln p\left(w\right) p\left(y \mid w, x\right) - \ln q_\theta\left(w\right)$.

There are several practical implementation of this optimization problem, for instance [42] proposes the Bayesian BP algorithm that extends the applicability of standard BP to NN weights represented by random variables. Subsequently, it is very remarkable the result of [43] following an alternative approach where the weights are generated by a deterministic operation and adding some random effect or just using Black-Block optimization techniques. The latter approach has been selected for implementation in the next section.

Once we have the variational distribution and we can generate samples from it, we obtain the following approximation for the predictive distribution using Monte-Carlo techniques:

$$\begin{aligned} p\left(y^* \mid \mathcal{D}, x^*\right) &= \int \underbrace{p\left(y^* \mid x^*, w\right)}_{likelihood} \underbrace{p\left(w \mid \mathcal{D}\right)}_{posterior} dw \approx \\ &\approx \int \underbrace{q_{\theta^*}\left(w\right)}_{Variational} \underbrace{p\left(y^* \mid x^*, w\right)}_{likelihood} dw \approx \frac{1}{S} \sum_{s=1}^{S} p\left(y^* \mid x^*, w^{(s)}\right) \end{aligned} \tag{4.21}$$

where we can generate S samples as follows $w^{(s)} \sim q_{\theta^*}\left(w\right)$.

4.2.3 Example. Centralized MNIST classification. Formulation.

We are going to study a problem using the previous ideas [43]. As it has been said, in this case, the variational learning tries to find the θ parameters of a neural network weight distribution, which is done by minimising the KL divergence with the true Bayesian posterior on the weights of the model. In this way, the intractable problem of bayesian inference, where infinitely many expectations under the posterior distribution over the weights must be computed, is solved by approximation.

To get a more practical view of the variational approach and to present some empirical evaluation of this method described before, the BP approximation made by Blundell has been implemented to solve the MNIST

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

classification problem obtaining a similar performance to the dropout implementation from [44] converging at similar rates. This could mean that uncertainty in the weights captures the most suitable neural network leading to a regularisation technique.

This implementation provides a PyTorch interface using Python 3.6+. The MNIST dataset has been trained using 60.000 pixel images of size 28x28 on a network which consisted of two hidden layers each one with 1200 rectified linear units and a softmax output layer of 10 units (10 classes), and tested using 10.000 pixel images of the same size. From those 60.000 samples to train, 10.000 were separated to pick the best hyper-parameters.

Although the scheme referred to in the paper allows for interleaving combinations of priors/posterior, the algorithm proposes to use a Gaussian prior or a scaled mixture of two Gaussian densities as a prior. The latter one gives the first component a larger variance than the second causing, firstly, a heavy tail in the prior density and, secondly, many of the weights concentrate around zero. In combination with this prior, a Gaussian variational posterior is proposed.

Method	#Units/Layer	Test Error (%)
[44] SGD	1200	1.88%
[44] SGD, dropout	1200	1.36%
[43] BBB, Gaussian	1200	2.04%
[43] BBB, Scale Mixture	1200	1.32%
[Implementation] Gaussian	1200	1.82%
[Implementation] Scale Mixture	1200	1.8%

Table 4.1: Results obtained using different BNN methods.

Table 4.1 shows a comparison between the results using either a Gaussian or Gaussian scale mixture prior obtained in [44][43] and the implementation proposed based on this latter one too. As it can be seen, the classification error results show similar values, being higher the ones obtained in our implementation, compared to the original one. This might happen due to the fact that [43] uses a really small variance for the distribution, in other words, almost a delta, which would mean the value of the weights is close to the traditional one, a concrete value, not a distribution. Finally, the next figures show this classification error convergence through all the epochs during the training process for a learning rate of 1e-5, showing that the one using a Gaussian scale mixture prior converges a little bit before than the other one.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

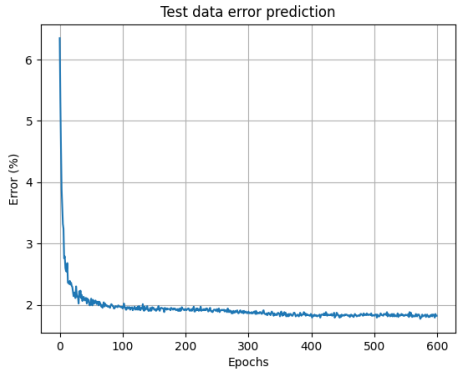


Figure 4.1: Test error using Gaussian prior

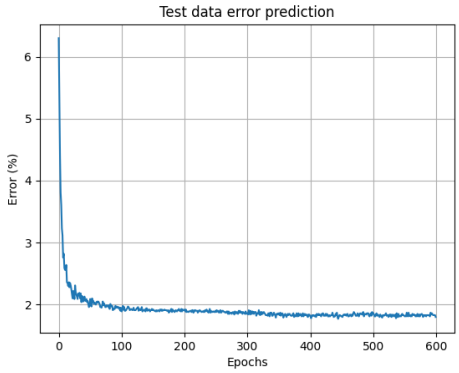


Figure 4.2: Test error using Gaussian scale mixture prior

4.2.4 Code guide

There are two scripts implemented to carry out the classification problem, «bnn.py» and «bnn_toolbox.py», and «requirements.txt» that specifies the different libraries needed for the environment on which it will be executed. As a general overview, the code provides a simple interface designed to learn a probability distribution on the weights of a NN and allowing to modify the dataset used through a simple user implementation. The first script, «bnn.py», loads the MNIST dataset, preprocesses the data and initializes the model and its training; the second one, «bnn_toolbox.py», has the whole network implemented, with the steps to follow while training and evaluating the results.

To run the problem, the following steps should be followed:

1. First of all, a Python 3.6+ environment is required.
2. The environment requirements should be installed. To do so, PIP is a package management system used to install and manage software packages written in Python:
   (a) *pip install -r requirements.txt*
3. Talking about data, the main script, «bnn.py», downloads the MNIST data, which is already provided in the repository, and saves it both in raw and processed ways in the path data\MNIST\. There are several datasets provided in the «torchvision» library, which are subclasses of «Dataset» owning methods like *__geitem__* and *__len__*. Hence, they can be passed to a «Dataloader», useful to load multiple samples in parallel and specifying the batch size or whether to shuffle the data or not, for example. It is easy enough, therefore, to change the dataset required by the user, since PyTorch facilities, such as the modules «Datasets» and «Dataloader», have been used.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

4. Finally, the main script should be run by setting the arguments desired.
   (a) To check which arguments can be set: *python bnn.py --help*
   (b) To run the code using a scale mixture of two Gaussian densities as the prior and 1200 hidden units (default), for example: *python bnn.py*
   (c) To run the code using a Gaussian prior and 600 hidden units, for example: *python bnn.py - -model gaussian_prior - -hidd_units 600*
5. The final results (model, .csv file with metrics and figures) will be stored in the following path: results\

4.3 Distributed implementations. Fundamentals

Recalling the expression of the posterior where we have now specified the local and global likelihoods, we have

$$p(w \mid \mathcal{D}) = \frac{p(w)\prod_n p(y_n \mid w, x_n)}{p(y \mid x)} \propto \underbrace{p(w)}_{prior}\underbrace{\prod_n \underbrace{p(y_n \mid w, x_n)}_{local\ likelihood}}_{Global\ likelihood} \tag{4.22}$$

4.3.1 Partitioned variational inference

4.3.1.1 Basic ideas

The basic idea is that we are going to approximate the target posterior by a factorized distribution as follows

$$p(w \mid \mathcal{D}) \propto \underbrace{p(w)}_{prior}\underbrace{\prod_n \underbrace{p(y_n \mid w, x_n)}_{local\ likelihood}}_{Global\ likelihood} \approx g(w) \propto g_0(w)\prod_{n=1}^{N} g_n(w) \tag{4.23}$$

where $g_0(w) \approx p(w)$ is calculated at a fictitious node, for instance at the server. Clearly, each term of $g(w)$ will intend to approximate the corresponding local likelihood

$$g_n(w) \approx p(y_n \mid w, x_n) \tag{4.24}$$

It is important now to introduce the following notation because we will need it in the forthcoming analysis: we define the *cavity distribution* (to be used as prior in some implementations) for an arbitrary node i.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$g_{-i}(w) = \frac{g(w)}{g_i(w)} = \prod_{n \neq i} g_n(w) \tag{4.25}$$

4.3.1.2 Main definitions

Reference [45] provides an interesting result providing a strong procedure for solving the GVI in a distributed way. Recalling the definition of the GVI

$$\mathcal{F}_{GVI}(q(w)) = -KL(q(w) \,||\, p(w)) + \sum_{n=1}^{N} \int q(w) \ln p(y_n \mid w, x_n)\, dw \tag{4.26}$$

it is shown that the optimization of this ELBO expression is equivalent if at each node n the following Local ELBO expression is optimized, which they call the Partitioned VI problem (PVI)

$$\mathcal{F}^n_{PVI}(q(w)) = -KL\left(q(w) \,||\, \underbrace{q^n_{-n}(w)}_{Local\ \ prior}\right) + E_{q(w)}\left(\underbrace{\ln p(y_n \mid w, x_n)}_{Local\ \ likelihood}\right) \tag{4.27}$$

interpreting that the cavity distribution at node n $q^n_{\theta,-n}(w)$ acts as a local prior. They proved that the convergence of the PVI algorithm leads to the optimal solution of GVI because:

$$\mathcal{F}_{GVI}(q(w)) = \sum_{n=1}^{N} \mathcal{F}^n_{PVI}(q(w)) \tag{4.28}$$

i.e., the PVI fixed point is an optimum of the GVI. So,

$$q^*(w) = \arg\max_{q(w)} \mathcal{F}^n_{PVI}(q(w)), \ \ \forall n \rightarrow q^*(w) = \arg\max_{q(w)} \mathcal{F}_{GVI}(q(w)) \tag{4.29}$$

Therefore, the procedure can be summarized as follows. Suppose that an arbitrary node i at a certain instant time t has a local copy of $q(w)$ that we denote as $q^{i,t}(w)$ that has been provided by the server. We calculate

1. Cavity distribution: $q^{i,t}_{-i}(w) = q^{i,t}_0(w) \prod_{n \neq i} q^{i,t}_n(w) = \frac{q^{i,t}(w)}{q^{i,t}_i(w)}$
2. Solve PVI: $q^{i,t+1}(w) = \arg\min_{q(w)} \mathcal{F}^i(q(w))$
3. Sending to the server: $\tilde{q}^{i,t+1}_i(w) = \frac{q^{i,t+1}(w)}{q^{i,t}_i(w)}$

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

4. At the server generate $q^{t+1}(w) = \frac{q^t(w)\tilde{q}_i^{i,t+1}(w)}{\tilde{q}_i^{i,t}(w)}$, $\forall i$ and share with other nodes.

You can see that this procedure is a particular implementation of the Expectation Propagation (EP) algorithm. The main procedure is based on the fact that locally we have to solve

$$\begin{aligned}\mathcal{F}^n\left(q_\theta(w)\right) &= -KL\left(q_\theta(w) \,||\, \underbrace{q_{-n}^n(w)}_{Local\ \ prior}\right) + E_{q_\theta(w)}\left(\underbrace{\ln p\left(y_n \mid w, x_n\right)}_{Local\ \ likelihood}\right) = \\ &= \int q_\theta(w) \ln \frac{q_{-n}^n(w)\, p\left(y_n \mid w, x_n\right)}{q_\theta(w)} dw = \int q_\theta(w) \ln \frac{q_{\setminus n}^n(w)}{q_\theta(w)} dw = \\ &= E_{q_\theta(w)}\left[\ln q_{\setminus n}^n(w) - \ln q_\theta(w)\right]\end{aligned} \tag{4.30}$$

Also, notice that operations based on multiplications and divisions of distributions remain at the core of the procedure as a non trivial task except for conjugate exponential models.

The following picture illustrates the idea for the distributed implementation:

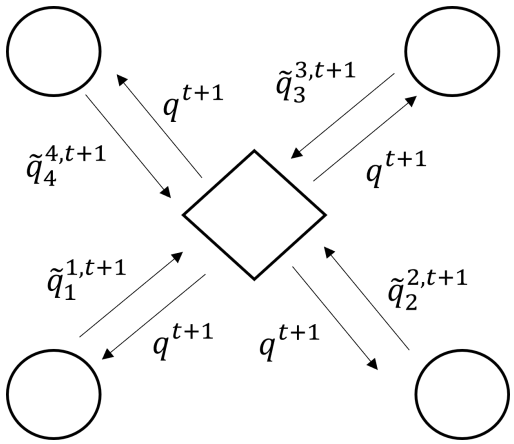


Figure 4.3: Federated learning ADMM distribution (non-convex optimization).

Different practical implementations are proposed in [46] and also in [47]. This last one is particularly interesting because it introduces an efficient procedure using a modified cost function and a two-loop update that has convergence guarantees (recall that standard EP does not). Clearly, the latter reference is a very strong candidate for general EP methods. Another method related to PVI, although obtained independently, is the one proposed in [48] called VIRTUAL that has been selected for practical implementation.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

4.3.2 Example. Distributed MNIST classification. Formulation

Let us apply VIRTUAL to the same MNIST classification problem used in the centralized setting. In this case, the variational learning tries to find the θ parameters of a weight distribution of a neural network, which is done by minimising the KL divergence with the true Bayesian posterior on the weights of the model. The algorithm proposed by [48] is named VIRTUAL and is a distributed version of Blundell's ideas, as we have already mentioned.

To get a more practical view of the variational approach and to present some empirical evaluation of this method, we adapt the author code and show how it solves the MNIST classification problem. The implementation provided only uses the common factor from the prior, i.e., the weights of the neural networks are parameterized by θ only. It could be possible to add also the private weights ϕ_i by using the bias of the neural network or setting the flag «hierarchical» to True in the provided code. The code uses a TensorFlow interface tested on Python 3.6. The MNIST dataset has been trained using 60.000 pixel images of size 28x28 on a network which consisted of two hidden layers each one with 100 rectified linear units and a softmax output layer of 10 units (10 classes), and tested using 10.000 pixel images of the same size. With these parameters, we obtain an accuracy of 97.15 % in the server, and an average error of 96.94% in the clients.

4.3.3 Code guide

The main script to be used is named «main_federated_bayesian.py» and it contains the instructions to run VIRTUAL algorithm. We also provide a «requirements.txt» that specifies the different libraries needed for the environment on which it will be executed. To run the problem, the following steps should be followed:

1. First of all, a Python 3.6+ environment is required.
2. The environment requirements should be installed. To do so, PIP is a package management system used to install and manage software packages written in Python:
   (a) *pip install -r requirements.txt*
3. Open the script «main_federated_bayesian.py», and note that there is a dictionary of hyperparameters named «config». By changing the parameters, we can change our neural network architecture, the number of clients, and the rest of hyperparamenters that have an influence on the training procedure. We provide also an example on how to load a dataset by making use of the MNIST problem, please note that you need to invoke the *prepare_dataset()* method in order to obtain the data that is going to be send to each node.
4. Finally, the last call is to the *run_experiment()* method, which invokes all the functions needed for training. Note that this code makes an extensive use of TensorFlow Distributions, so all the computation and updates of distributions are done under the hood, which is a significant difference regarding

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

the implementation provided for the centralized case. Also, note that the implementation heavily depends on the use of Deferred Tensors, which are a special class of TensorFlow Probability that updates the whole chain of tensors every time that one of the tensors of the chain changes. That is, if the parameters of a distribution changes, then all the distributions that depend on this one change as well.

5. In order to run the code with the default hyperparameters, execute *python main_federated_bayesian.py*
6. The final results will be stored in the following path: logs\

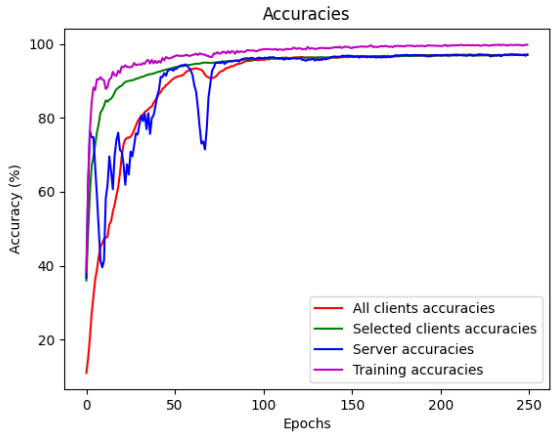


Figure 4.4: BNN's performance in federated environment

The previous figure shows an example of this model's execution where four different accuracy curves obtained during the process of training can be analyzed: the whole model's training accuracy, both the server's and the clients' performance, and a selected group of clients.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

5 Federated Data Augmentation

In the previous chapter we have presented a powerful approach that permits the implementation of discriminative problems in a distributed way using neural networks. However in practice it can be difficult to calculate the product/quotient of complicated distributions such as those we expect in medical records. Furthermore, it should be noted that the optimization procedure involves several iterations running through the nodes, where each node has to solve a variational inference problem in high-dimensional settings.

These limitations motivated us to also explore alternative implementations and we came up with a simple idea: if the problem is the amount and variety of data we have in small local sites, instead of moving real patients information that is forbidden we can share some latent information that allows us to build virtual patients following similar statistics.

We were lucky because there is a good publication [49] that proposes a powerful architecture using generative adversarial networks (GANs) that meet the two main challenges of trying to use data from multiple sources: feature mismatch and distribution mismatch.

- Feature mismatch refers to the fact that even between datasets drawn from the same field (such as medicine), the features that are actually recorded for each dataset may vary. Therefore, on the one hand, we need to deal with the fact that the ”auxiliary” hospitals’ datasets do not contain all the features that have been measured by the target hospital and, on the other hand, we have to take advantage of the information contained in the features that are measured by the ancillary hospitals but not by the target hospital.
- Distribution mismatch refers to the fact that the patient population of two hospitals may vary. Therefore, we need to prevent the federated implementation from bias created by different populations.

The architecture that they are proposing is known as RadialGAN and will be described in next section.

5.1 RadialGAN fundamentals

Radial GAN generalizes and outperforms several previous approaches: let us highlight a proposal for transfer learning with multiple data sets proposed in [50] and also Adversarially Learned Inference with Conditional Entropy (ALICE) [51] which is an extension of ALI that learns mappings which satisfy both reversibility (i.e. that the distribution of mapped samples match those of the other distribution in both directions) and cycle-consistency. More specifically, ALICE introduces conditional entropy loss in addition to the adversarial loss

This project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

to restrict the conditionals, thus enforcing cycle-consistency. In addition, CycleGAN [52] and DiscoGAN [53] propose further frameworks for estimating reversible cycle-consistent mappings between two domains. Using explicit reconstruction error instead of conditional entropy, CycleGAN and DiscoGAN ensure cycle-consistency. Finally, multi-domain translation as StarGAN [54] can be considered as a preliminary work that proposes a framework for multi-domain translation that is scalable to multiple domains by using a single generator (mapping) that takes as an additional input the target domain that the sample is to be mapped to.

The main idea is that they use multiple GAN architectures to translate patient information from one hospital to another, leveraging the adversarial framework to ensure that the learned translation respects the distribution of the target hospital. To learn multiple translations efficiently and simultaneously, they introduce a latent space through which each translation occurs. This has the added benefit of naturally addressing the problem of feature mismatch - all samples are mapped into the same latent space.

5.2 RadialGAN formulation

Suppose that we have M spaces $\mathcal{X}^{(1)};\cdots;\mathcal{X}^{(M)}$, and that for each i, $X^{(i)}$ is a random variable taking values in $\mathcal{X}^{(i)}$. Suppose further that Y is a random variable taking values in some label space $\mathcal{Y}$. Suppose also that we have M datasets $\mathcal{D}^1;\cdots;\mathcal{D}^M$ with

$$\mathcal{D}^i = \left\{\left(x_j^{(i)}, y_j^i\right)\right\}_{j=1}^{n_i} \tag{5.1}$$

where $\left(x_j^{(i)}, y_j^i\right)$ are i.i.d. realizations of the pair $(X^{(i)};Y)$ and n_i is the total number of realizations (observations) in dataset i.

The goal is to learn M predictors, $f_1;\cdots;f_M$ (with $f_i : \mathcal{X}^{(i)} \to \mathcal{Y}$) such that we want to use all datasets in learning f_i, not only $(X^{(i)};Y)$ but also $(X^{(j)};Y)$, $\forall j \neq i$. The procedure consists of significantly augmenting the local dataset with some virtual patients created from the data of the rest of the centres, but maintaining a certain consistency behaviour that must be conveniently applied:

$$\widetilde{D}_i = \mathcal{D}_i \underset{j\neq i}{\cup} G_i\left(F_j\left(\mathcal{D}_j\right)\right) \tag{5.2}$$

where F_j refers to the encoder at node j such that

$$F_j : \mathcal{X}^{(j)} \times \mathcal{Y} \to \mathcal{Z} \tag{5.3}$$

as a mapping into a common latent space and G_i corresponds to the decoder part

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$G_i : \mathcal{Z} \rightarrow \mathcal{X}^{(i)} \times \mathcal{Y} \tag{5.4}$$

The architecture that is proposed is composed of two terms, one is an adversarial loss, while the other is cycle-consistency loss that ensures that virtual data has been properly generated. The following figure illustrates the idea where a common latent space is used for sharing the local information preserving the privacy of local records.

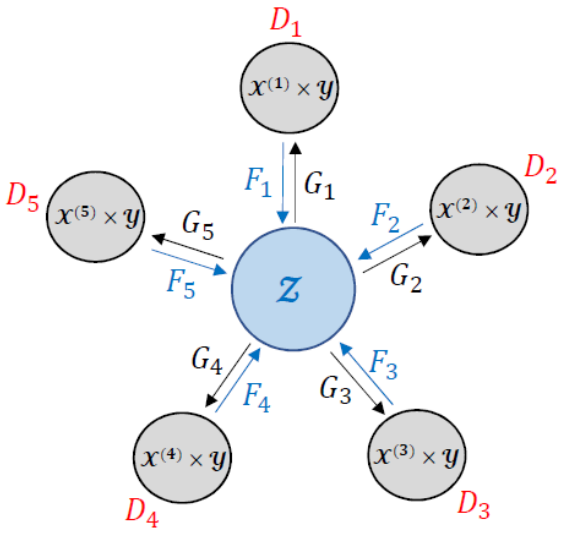


Figure 5.1: Local information shared through latent space

5.2.1 Training

In this section we will explain both criteria and give some intuition. The adversarial term is described in the following formula:

$$\begin{aligned} \mathcal{L}^i_{adv} &= E[\ln D_i(X^{(i)};Y)] + E[\ln(1\text{-}D_i(\widehat{X}^{(i)}\widehat{Y}))] = \\ &= E[\ln D_i(X^{(i)};Y)] + E[\ln(1\text{-}D_i(G_i(Z_i))] = \\ &= E[\ln D_i(X^{(i)};Y)] + \sum_{j\neq i}\alpha_{ij}E[\ln(1\text{-}D_i(G_i\left(F_j\left((X^{(j)};Y)\right)\right))] \end{aligned} \tag{5.5}$$

where α_{ij} represents the assigned weight of different nodes, usually defined by the number of samples.

On the other hand, the cycle-consistency loss is expressed as follows:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$\begin{aligned}\mathcal{L}^i_{cyc} = E[\left\|(X^{(i)};Y) - G_i\left(F_i\left((X^{(i)};Y)\right)\right)\right\|_2] + E[\|Z_i - F_i\left(G_i\left(Z_i\right)\right)\|_2] = \\ = E[\left\|(X^{(i)};Y) - G_i\left(F_i\left((X^{(i)};Y)\right)\right)\right\|_2] + \\ + \sum_{j\neq i}\alpha_{ij}E[\left\|F_j(X^{(j)};Y) - F_i\left(G_i\left(F_j\left((X^{(j)};Y)\right)\right)\right)\right\|_2]\end{aligned} \tag{5.6}$$

Clearly, cycle-consistency loss guarantees that the translation into and back to the latent space returns something close to the original input and that mapping from the latent space into one of the domains and back again also returns something close to the original input. Schematically, both criteria can be represented in the following figure.

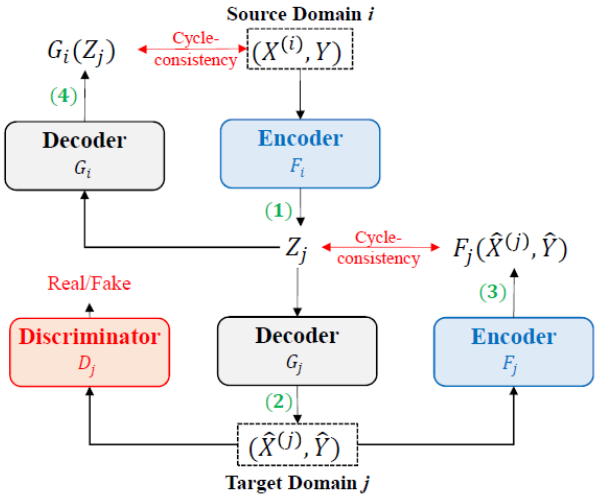


Figure 5.2: RadialGAN schematic architecture

For the training, the global cost function is:

$$\min_{G,F}\max_{D}\left(\sum_{i=1}^{M}\mathcal{L}^i_{adv}\left(D_i, G_i, \{F_j : j \neq i\}\right) + \lambda\sum_{i=1}^{M}\mathcal{L}^i_{cyc}\left(G_i, F\right)\right) \tag{5.7}$$

where λ is just a weighting factor.This approach can be interpreted more easily for two users splitting the two components. The adversarial network is:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

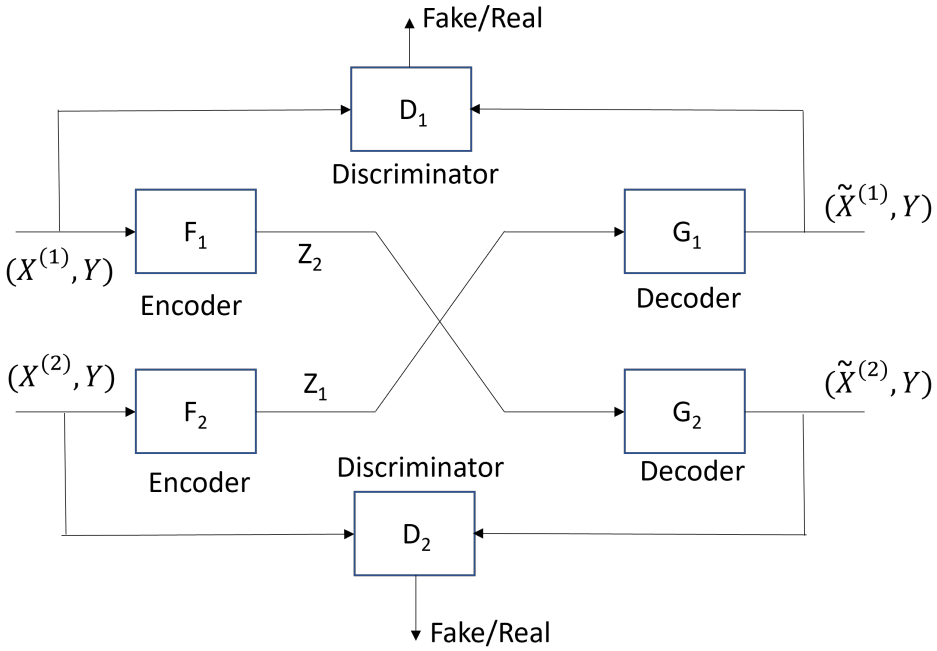


Figure 5.3: RadialGAN schematic training process for two users

and the cycle consistency is

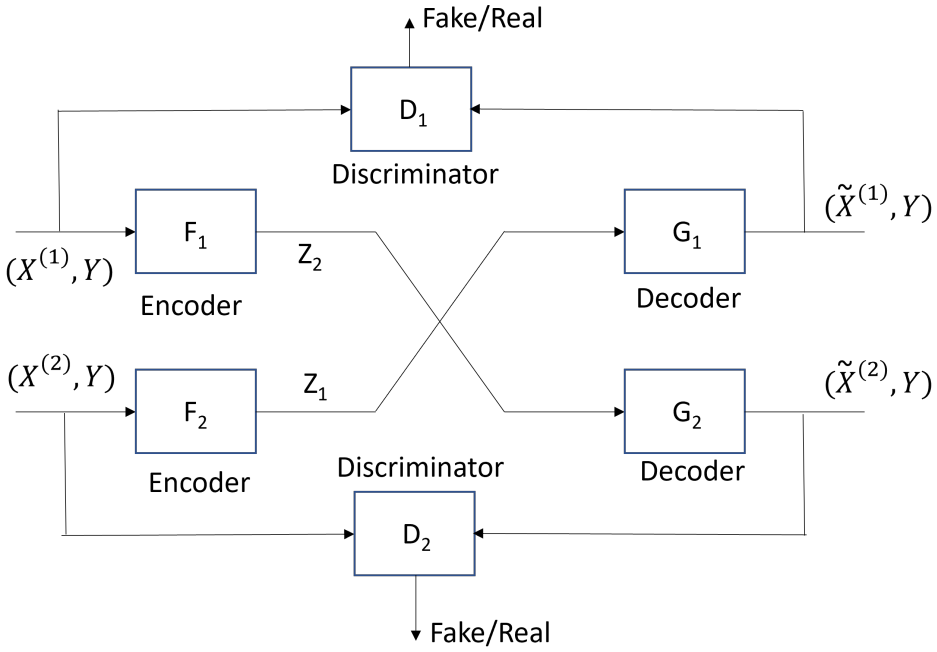


Figure 5.4: RadialGAN schematic cycle-consistency process for two users

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

5.2.2 Prediction

After we have trained the translation functions $G_1; \cdots ; G_M$ and $F_1; \cdots ; F_M$, we create M augmented datasets

$$\widetilde{D}_i = \mathcal{D}_i \bigcup_{j \neq i} G_i \left(F_j \left(\mathcal{D}_j \right) \right) \tag{5.8}$$

that we have already commented while the predictors $f_1; \cdots ; f_M$ are then learned in a separate phase on these augmented datasets. This phase could use an arbitrary convex or neural network architecture and perform training in a completely completely standard way.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

6 UPM contribution

6.1 Motivation

A desirable initial key feature of our contribution is to propose a single architecture (possibly with multiple functionalities) capable of solving different problems as survival analysis, prediction, stratification (supervised learning based on regression and classification) together with clustering and feature extraction (unsupervised learning). Furthermore, it is desirable that it is a common framework for all three case studies.

This decision cannot be made without taking into account that the final implementation should be federated, so a clear procedure for influencing some users over others must be taken into account. Our proposal is based on Generative Models (GM) using Variational Autoencoders (VAEs) that will probably be combined with some adversarial training (GANs-like). The general idea is that different nodes have to reach a consensus on a certain space representing the data. Clearly, this space cannot be the original data because it cannot be shared, so, it makes perfect sense to think of a common latent space that is created in a collaborative way. This section illustrates the reasoning.

6.1.1 Single node

Let us formulate the original VAE algorithm to clarify the notation we will follow:

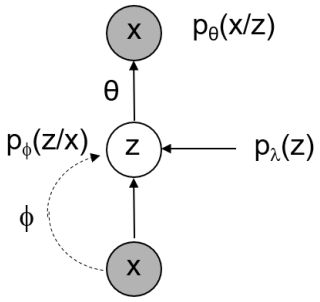


Figure 6.1: Original VAE algorithm scheme.

The ELBO is

$$\mathcal{L}_S(\theta,\phi,\lambda) = \int q_\phi(z \mid x)\left[\ln p_\theta(x \mid z) + \ln p_\lambda(z) - \ln q_\phi(z \mid x)\right] dz \tag{6.1}$$

where you can notice that the only innovation comes from the fact that we have included a parameterized

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

prior $p_\lambda(z)$. The idea is that instead of using a trivial distribution as usual, we can provide some more elaborate proposals [55] [56] [57].

6.1.2 Multiple nodes, Centralized solution

This VAE formulation can be extended to the multiple nodes scenarios (suppose only two in the figure for the sake of clarity)

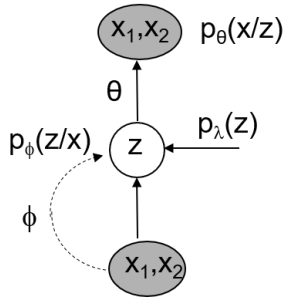


Figure 6.2: Multiple nodes VAE algorithm scheme.

where we have a common latent space z, a single set of parameters $\{\theta, \phi\}$ while we maintain different sources $\{x_1, x_2, \cdots, x_M\}$.

The ELBO can be formulated identically

$$\mathcal{L}_M(\theta, \phi, \lambda) = \int q_\phi(z \mid x_1, \cdots, x_M) \left[\ln p_\theta(x_1, \cdots, x_M \mid z) + \ln p_\lambda(z) - \ln q_\phi(z \mid x_1, \cdots, x_M)\right] dz \tag{6.2}$$

6.2 Multiple nodes, Distributed solution

It is clear that previous approach is unfeasible in a distributed setting, so we have to trend to a different architecture as the one showed in next figure:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

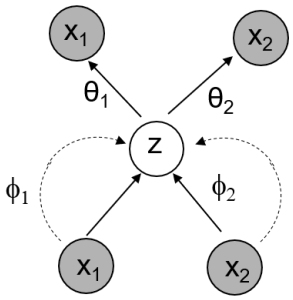


Figure 6.3: Multiple nodes VAE distributed algorithm scheme.

where we have separated sources (never shared) and also private NNs for the encoders $\{\phi_1, \phi_2, \cdots, \phi_M\}$ and decoders $\{\theta_1, \theta_2, \cdots, \theta_M\}$ while we keep a common latent space to introduce the simplest understanding of this family of Federated Learning (FL) algorithms. In practice, we have to be focused in more efficient architectures, each one with its private latent space $\{z_1, z_2, \cdots, z_M\}$ but sharing some information among the nodes in order to emulate the *common latent space* already presented. These practical approaches can be viewed in next picture.

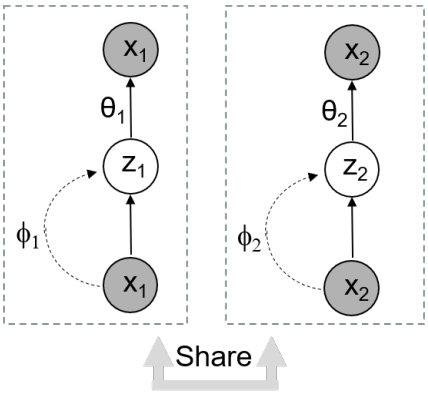


Figure 6.4: Multiple nodes architecture sharing information scheme.

Therefore, the key point is how to decide what information must be shared respecting the privacy requirements.

6.3 Are GMs a suitable architecture?

This question concerns the reasoning on the suitability of this architecture for solving general supervised and unsupervised algorithms taking into account that VAEs are generative models. The answer is probably yes, emphasizing the role of the latent space, if we consider it as an accurate representation of the input data into which we can also introduce additional features in order to create a certain desired structure such as

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

clustering.

The following figure schematically illustrates the approach:

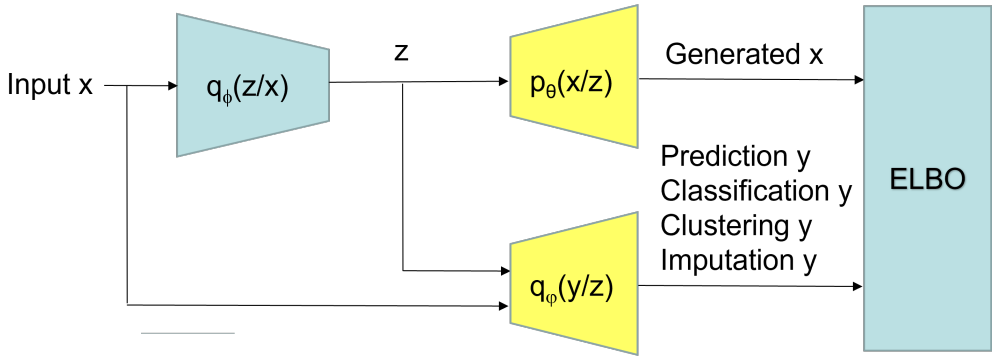


Figure 6.5: General scheme.

Let us review the literature in order to provide some support to this consideration. One key element must be clear: all involved processes are coupled, which means that we have to derive a suitable bound to the specific problem (an ELBO) and optimise it appropriately using variational techniques. The computation of specific ELBO usually requires solving two important challenges: the application of the parameterization trick beyond the Gaussian distribution and the computatuib of the Kullback-Leibler divergence, which must be solved analytically for the involved distributions (otherwise, the problem becomes very complicated).

6.3.1 Clustering

Clustering is a very important topic that aims to obtain some hidden structurefrom the incoming data to be clustered in a certain latent space. There are many papers on the topic where probably the most standard approach is to enforce a certain prior as a Gaussian mixture [58] [59] [60]. This approach requires defining a hierarchical model on the latent space to represent the mixture as a categorical distribution and the cluster distribution usually as a multivariate Gaussian model.

6.3.2 Imputation

Imputation can also be improved using VAEs architectures. As a good example we have [61] where the latent space is enlarged in order to include also the missing features that are generated by a separate network in a similar way as the standard latent space. The inference model is:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$q(z_n, x_n^m | x_n^o) = q(z_n | x_n^o) \prod_{d \in \mathcal{M}_d} p(x_{nd} | z_n). \tag{6.3}$$

where $x_n^m \in \mathcal{M}_d$ is the set of missing attributes and $x_n^o \in \mathcal{O}_d$ is the set of observed attributes.

6.3.3 Supervised (semisupervised) learning

This topic has also been addressed from different perspectives where probably [62] was the first publication who studied how can properties of the data be used to improve decision boundaries and to allow for classification/regression that is more accurate than that based on classifiers/regressors built on labelled data alone.

6.3.4 Survival Analysis

Survival analysis models the time to an event from a common start. Examples of survival data include time to death or to hospitalization when suffering a certain disease. Survival observations consist of two varieties: the first are observations for which the exact failure time is known; the second, called censored observations, are observations for which the failure time is known to be greater than a particular time. Both types are represented as $(t; c)$, pairs of positive times and binary censoring status. Probably [63] was the first proposal on the topic using VAEs although the paper is not clear and it does neither provide any code. They introduce deep survival analysis, a hierarchical generative approach for survival analysis. It departs from previous approaches in two primary ways. First, all observations, including covariates, are modeled jointly and conditioned on a rich latent structure. Second, patient records align by their failure time rather than by entry time, thus resolving the ambiguity of entry to the analysis.

6.4 Survival analysis using Variational Autoencoders

In this chapter, we provide detailed view on methods and ideas used for Survival Analysis (SA) using Variational AutoEncoders (VAEs) as main tool. This has been the main contribution of UPM but many results are still to come.

6.4.1 Vanilla Variational Autoencoder

In 2013, the original VAE paper was published [64] where a powerful approach based on Deep Neural Networks (DNNs) was used in order to perform Bayesian inference on the problem depicted in Figure 6.6.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

We consider a dataset composed by N i.i.d. samples $x_i, i \in 1, 2, ..., N$ of a continuous or discrete variable. The data x_i is generated by the following random process:

1. A value z_i is sampled from a certain prior distribution $p(z)$, where z_i is denoted as latent variable. The original paper assumes a shape $p_\theta(z)$, that is, the prior depends on some parameters θ, but their main result does drop that dependence. Hence, we assume a simple prior $p(z)$.
2. A value x_i is generated following a certain conditional distribution $p_\theta(x|z)$, where θ are parameters of this distribution, which we denote as generative model.

Some assumptions are that the pdfs $p(z)$ and $p_\theta(x|z)$ are differentiable almost everywhere with respect to θ and z. Note that we do not know neither the values of the latent variable z nor the parameters θ. If we do not make any simplifying assumption on the marginal likelihood $p_\theta(x) = \int p(z)p_\theta(x|z)dz$, then, we cannot evaluate the true posterior density, which is defined as follows by using Bayes' Theorem:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p(z)}{p_\theta(x)} \tag{6.4}$$

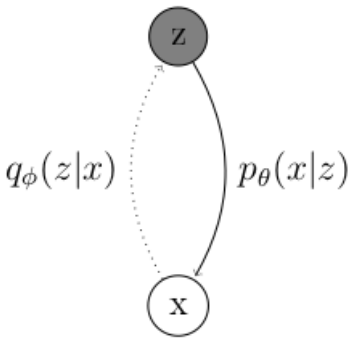


Figure 6.6: Vanilla VAE Bayesian model, where the shadowed circle refers to latent variable and white circle to the observable. Note that the probabilities p and q denote, respectively, the generative model $p_\theta(x|z)$ and the variational approximation to the posterior $q_\phi(z|x)$, as the true posterior $p(z|x)$ is unknown.

A common approach to solving this kind of problem consists on using variational methods, which rely on defining a variational distribution from a parametric family and fitting the parameters to those that best approximate the desired distribution. Note that this approach involves exchanging a complex problem (finding a probability distribution, which is a functional analysis problem) by a less complex one (finding the best parameters for a given distribution family, which is an optimization problem). Of course, the quality of the approximation will be strongly conditioned by the expressiveness of the chosen parametric family.

In our case, we introduce the variational approximation $q_\phi(z|x)$, which is our approximation to the true posterior.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

6.4.2 ELBO expression

The expression of the ELBO for an arbitrary sample x_i is

$$\log p_\theta(x_i) \geq -D_{KL}(q_\phi(z|x_i)||p(z)) + \mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right] = \mathcal{L}(x_i, \theta, \phi) \tag{6.5}$$

that is, the ELBO $\mathcal{L}(x_i, \theta, \phi)$ is a lower bound for the marginal log-likelihood of our set of points. Thus, by maximizing the ELBO, we maximize the log-likelihood of our data, and that is the optimization problem we want to solve.

6.4.2.1 Implementation of Vanilla VAE

It is possible to implement the ELBO obtained in (6.5) by making use of an architecture based on DNNs. However, computing the gradient of the ELBO with respect to ϕ is difficult due to the fact that ϕ appears in the expectation (i.e., the second term of the ELBO in (6.5)). In order to overcome this problem, [64] uses the reparameterization trick, which consists on using a deterministic transformation g_ϕ of a random variable ϵ such that $z = g_\phi(x, \epsilon)$ where $z \sim q_\phi(z|x)$ and $\epsilon \sim p(\epsilon)$. In this case, the ELBO can be estimated as follows:

$$\hat{\mathcal{L}}(x_i, \theta, \phi) = -D_{KL}(q_\phi(z|x_i)||p(z)) + \frac{1}{L}\sum_{l}^{L} \log p_\theta(x_i|g_\phi(x_i, \epsilon_{i,l})) \tag{6.6}$$

where we obtain L samples from $\epsilon \sim p(\epsilon)$ to estimate the second term. notes that sometimes it suffices with using $L = 1$, although $L > 1$ can be used when required. Now, we can compute the gradient of the ELBO with respect to θ and ϕ, and this allows using standard gradient techniques to maximize it. Even more, expression (6.6) can be used to solve the problem using DNNs as the functions parameterized by ϕ and θ respectively, as all the gradients can be obtained by using Backpropagation algorithm, which several programming libraries obtain in an automatic fashion.

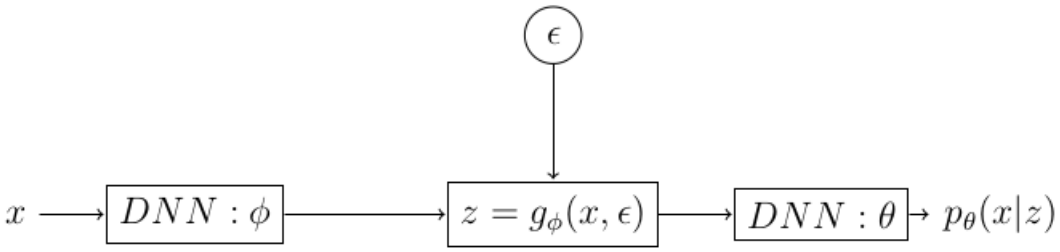


Figure 6.7: Vanilla VAE implementation using DNNs.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

6.4.3 Survival Analysis Model

In a conventional time-to-event (Survival Analysis, SA) setup, we are given N observations. Individual observation are described by triplets

$$\mathcal{D} = (x_i, t_i, d_i)_{i=1}^{N} \tag{6.7}$$

where $x_i = x_{i1}, ..., x_{p1}$ is a p-dimensional vector of covariates, t_i is the time-to-event and $d_i \in 0, 1$ is the censoring indicator. When $d_i = 0$ (censored) the subject has not experienced an event up to time t_i, while $d_i = 1$ indicates observed (ground truth) event times. Time-to-event models are conditional on covariates: the event time density function $f(t|x)$, the hazard rate (risk score) function $h(t|x)$ or the survival function

$$S(t|x) = P(T > t) = 1 - F(t|x) \tag{6.8}$$

also known as the probability of failure occurring after time t, where $F(t|x)$ is the cumulative density function. From standard survival function definitions, the relationship between these three characterizations is formulated as:

$$f(t|x) = h(t|x)S(t|x) \tag{6.9}$$

6.4.3.1 Weibull model

Regarding the chosen distribution for the time event we have followed several publications where the Weibull distribution model is used. The Weibull distribution is a two parameter distribution whose support is positive, i.e., $f(t) = 0, \forall t < 0$. The two scalar parameters of the distribution are λ, α, where $\lambda > 0$ controls the scale and $\alpha > 0$ controls the shape as follows:

$$\begin{cases} f(t \mid \alpha(x), \lambda(x)) = \frac{\alpha(x)}{\lambda(x)} \left(\frac{t}{\lambda(x)}\right)^{\alpha(x)-1} \exp\left(-\left(\frac{t}{\lambda(x)}\right)^{\alpha(x)}\right) \\ S(t \mid \alpha(x), \lambda(x)) = \exp\left(-\left(\frac{t}{\lambda(x)}\right)^{\alpha(x)}\right) \\ h(t \mid \alpha(x), \lambda(x)) = \frac{\alpha(x)}{\lambda(x)} \left(\frac{t}{\lambda(x)}\right)^{\alpha(x)-1} \end{cases} \tag{6.10}$$

Note that when $\alpha = 1$, we obtain the exponential distribution. In our case, α can be understood as the evolution of the probability of the event over time: $\alpha = 1$ means that the probability of event is constant over time, $\alpha < 1$ means that the probability of event decreases over time, and $\alpha > 1$ means that the probability increases over time. It should also be noted that sometimes the scale parameter λ is inverted: this is something we must pay attention to in order to avoid errors (i.e., some code that generates samples from the Weibull distribution take as scale parameter $1/\lambda$.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

6.4.4 The initial approach: Survival Analysis VAE (SAVAE)

Let us propose an initial approach, that we name Survival Analysis VAE (SAVAE). As mentioned before, we are interested in obtaining the predictive distribution of the time-to-event, given a certain set of covariates. Let us assume the Bayesian model in Figure 6.8, where we have:

- A latent variable z, that we assume continuous.
- Two observables, namely, the covariate vector x and the time-to-event t. We assume that the two generative models $p_{\theta_1}(x|z)$ and $p_{\theta_2}(t|z)$ are conditionally independent, that is, if we know z, we can generate either x or t.
- As we are interested in the predictive distribution as a function of the covariates of a new patient, we only have to use a single variational distribution to estimate the true variational posterior $p(z|x)$. Even though it could be possible to include also the effect of the time, and hence, obtaining $p(z|t,x)$, in this initial approach we only use the covariates to obtain the latent space, while the inclusion of time is left for future work.

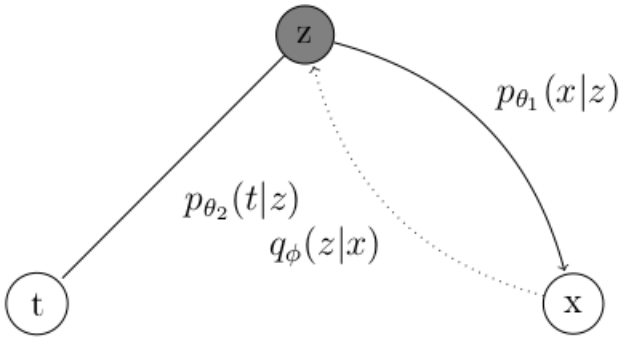


Figure 6.8: SAVAE Bayesian model, where the shadowed circle refers to latent variable and white circles to the observables. Note that the probabilities p and q denote, respectively, the generative models $p_{\theta_1}(x|z)$ and $p_{\theta_2}(t|z)$, and the variational approximation to the posterior $q_\phi(z|x)$, as the true posterior $p(z|x)$ is unknown.

6.4.4.1 Goal

Our main target is to use variational methods, namely VAEs, in order to obtain the predictive distribution for the time-to-event as follows [63]

$$p\left(t^* \mid x^*, \{x_i, t_i\}_{i=1}^N\right) = \int p\left(t^* \mid z, \{x_i, t_i\}_{i=1}^N\right) p\left(z \mid x^*, \{x_i, t_i\}_{i=1}^N\right) dz \tag{6.11}$$

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

where x^* represents the covariates of a certain patient, and we want to estimate its survival time distribution $p\left(t^* \mid x^*, \{x_i, t_i\}_{i=1}^N\right)$. Again, note that we consider that the latent variable z depends only on the training set (as they determine the DNN weights) and the latent variable, whereas the latent variable depends on the training set and the covariates. As mentioned before, we could include the effect of the covariates on the time prediction, but that is left to future work.

6.4.5 ELBO derivation

The main assumptions of our model are the following ones:

- The two generative models $p_{\theta_1}(x|z)$ and $p_{\theta_2}(t|z)$ are conditionally independent, that is, if we know z, we can generate either x or t. We further assume that each component of the covariates vector is conditionally independent given z. Hence, $p(x,t,z) = p(x_1|z)p(x_2|z)...p(x_p|z)p(t|z)p(z) = p_{\theta_1}(x|z)p_{\theta_2}(t|z)p(z) = p_\theta(x,t|z)p(z)$.
- We know the distribution shape of $p_{\theta_1}(x|z)$ and $p_{\theta_2}(t|z)$, but we do not know the parameters θ_1 and θ_2.

With these assumptions, we can compute the ELBO in a similar way to the Vanilla VAE case as follows. Let us start by noting that the conditional likelihood of a set of points $\{x_i, t_i\}_i^N$ can be expressed as follows:

$$\begin{aligned}\log p_\theta(x_1, x_2, ..., x_N, t_1, t_2, ..., t_N|z) &= \sum_i^N \log p_\theta(x_i, t_i|z) \\ &= \sum_i^N \left(\log p_{\theta_2}(t_i|z) + \sum_{m=1}^p \log p_{\theta_{1m}}(x_{im}|z)\right)\end{aligned} \tag{6.12}$$

After some derivations we obtain:

$$\begin{aligned}\hat{\mathcal{L}}(x_i, \theta_1, \theta_2, \phi) = &-D_{KL}(q_\phi(z|x_i)||p(z)) \\ &+\frac{1}{L}\sum_l^L \left[\log p_{\theta_2}(t_i|g_\phi(x_i, \epsilon_{i,l})) + \sum_{m=1}^p \log p_{\theta_{1m}}(x_{im}|g_\phi(x_i, \epsilon_{i,l}))\right]\end{aligned} \tag{6.13}$$

In terms of implementation, we use three DNNs as in Figure 6.9. Note that the main difference is that now, the decoder DNNs outputs the parameters of each distribution. Also, we noted that there is a trade off between accuracy in the times predicted (measured in terms of C-index) and the weight w assigned to the reconstruction term of the t variable:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

$$\hat{\mathcal{L}}(x_i, \theta_1, \theta_2, \phi) = -D_{KL}(q_\phi(z|x_i)||p(z)) + \frac{1}{L}\sum_{l}^{L}\left[w \cdot \log p_{\theta_2}(t_i|g_\phi(x_i, \epsilon_{i,l})) + \sum_{m=1}^{p} \log p_{\theta_{1m}}(x_{im}|g_\phi(x_i, \epsilon_{i,l}))\right] \tag{6.14}$$

where the divergence and the marginal log-likelihood computations require specific derivations.

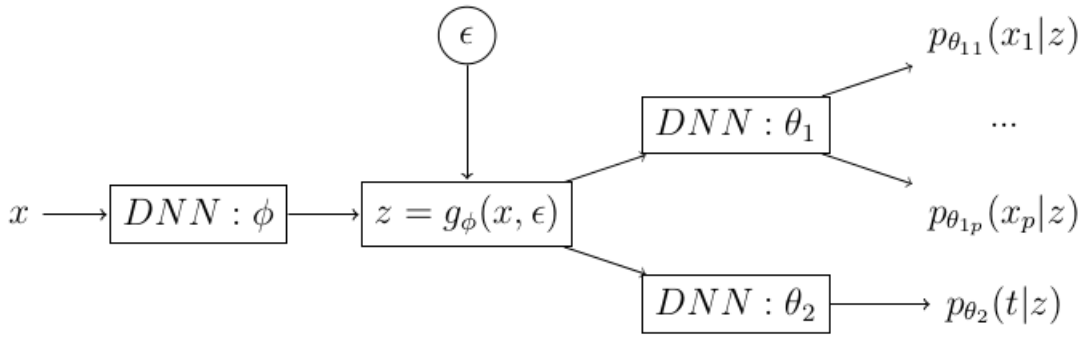


Figure 6.9: SAVAE implementation using DNNs.

6.5 Procedures to couple the learning of different nodes in VAEs' architectures

In this section we are going to introduce two different procedures in order to introduce the federated learning procedure in VAEs. On the one hand, jointly conforming the prior. On the other, using virtual patients models.

- At this time, we have not found satisfactory performance using the first approach. It seems that the effect of the prior is not so critical and it is difficult to drive the evolution of the encoder-decoder just pushing from this side.
- The use of virtual patients from good nodes (hospitals with high number of users and with most of the features' space complete) feedback into bad nodes seems to be a more direct way to influence these nodes that are able to estimate the missing features and perform improved supervised or unsupervised tasks.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

6.5.1 Joint design of the prior

6.5.1.1 The isolated node

Remembering the original VAE algorithm:

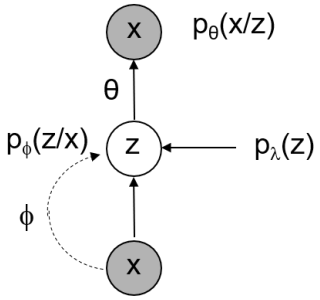


Figure 6.10: Original VAE algorithm scheme.

where the ELBO was already presented

$$\mathcal{L}_S(\theta,\phi,\lambda) = \int q_\phi(z \mid x)\left[\ln p_\theta(x \mid z) + \ln p_\lambda(z) - \ln q_\phi(z \mid x)\right] dz \tag{6.15}$$

where you can notice that the only innovation comes from the fact that we have included a parameterized prior $p_\lambda(z)$.

The optimum solution regarding the prior choice is given by the following condition

$$\frac{\partial \mathcal{L}_S(\theta,\phi,\lambda)}{\partial \lambda} = 0 \rightarrow p_{\lambda^*}(z) = q_\phi(z) \tag{6.16}$$

that is known as the aggregated posterior. Typically, it is estimated as an average

$$p_{\lambda^*}(z) \approx \frac{1}{N}\sum_{n=1}^{N} q_\phi(z \mid x_n) \tag{6.17}$$

but however, this choice may potentially lead to over fitting, so in practice it can be approximated by

$$p_{\lambda^*}(z) = \frac{1}{N}\sum_{n=1}^{N} q_\phi(z \mid x_n) \approx \frac{1}{K}\sum_{k=1}^{K} q_\phi\left(z \mid \alpha^{(k)}\right) \tag{6.18}$$

where α_k are known as the pseudopoints that have be optimized, so

$$\lambda = \left\{\phi, \alpha^{(1)}, \alpha^{(2)}, \cdots, \alpha^{(K)}\right\} \tag{6.19}$$

Notice that $p_{\lambda^*}(z)$ dose not depend on data! so, the pseudopoints can be shared!.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

6.5.1.2 Multiple nodes

Also remembering the introduction where we presented the general idea:

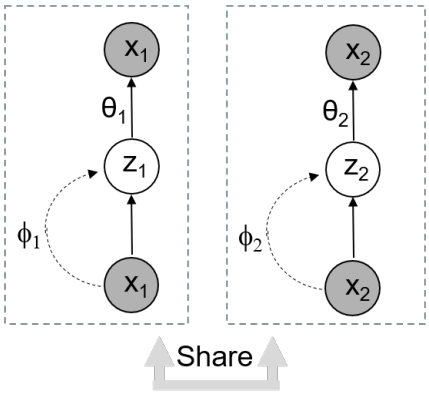


Figure 6.11: Multiple nodes architecture sharing information scheme.

we can propose a constructive methodology defining the following optimization set up. Given a local i node we have the corresponding VAE architecture:

$$\mathcal{L}_i\left(\theta_i, \phi_i, \lambda\right) = \int q_\phi\left(z_i \mid x_i\right)\left[\ln p_\theta\left(x_i \mid z_i\right) + \ln p_\lambda\left(z_i\right) - \ln q_\phi\left(z_i \mid x_i\right)\right] dz_i \tag{6.20}$$

where $\{\theta_i, \phi_i\}$ are local variables while λ is a global variable. It is clear that previous formulation will run independently at different nodes unless we tie the optimization process with a common constraint. The idea is to reach a consensus in the distribution of the prior $p_\lambda\left(z_i\right),\ z = z_i,\ \forall i$. The criteria is very intuitive, in the centralized case, we can solve

$$\frac{\partial \mathcal{L}_M\left(\theta, \phi, \lambda\right)}{\partial \lambda} = 0 \rightarrow p_{\lambda^*}\left(z\right) = q_\phi\left(z\right) \approx \frac{1}{N}\sum_{n=1}^{N} q_\phi\left(z \mid x_1^{(n)}, x_2^{(n)}, \cdots, x_M^{(n)}\right) \tag{6.21}$$

A first obstacle is that this aggregated version depends on the local data, that can not be shared, so we are going to approximate it by the pseudopoints

$$p_\lambda\left(z\right) \approx \frac{1}{N}\sum_{n=1}^{N} q_\phi\left(z \mid x_1^{(n)}, x_2^{(n)}, \cdots, x_M^{(n)}\right) \approx \frac{1}{K}\sum_{k=1}^{K} q_\phi\left(z \mid \alpha_1^{(k)}, \alpha_2^{(k)}, \cdots, \alpha_M^{(k)}\right) \tag{6.22}$$

This distribution can be considered as separable assuming that data are independent, so we have

$$p_\lambda\left(z\right) \approx \frac{1}{\mathcal{Z}}\prod_{m=1}^{M} \underbrace{\left(\frac{1}{K}\sum_{k=1}^{K} q_{\phi_m}\left(z \mid \alpha_m^{(k)}\right)\right)}_{p_{\lambda_m}(z)} = \frac{1}{\mathcal{Z}}\prod_{m=1}^{M} p_{\lambda_m}\left(z\right) \tag{6.23}$$

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

where $\mathcal{Z}$ is the normalizing factor and at each node we have to approximate

$$p_{\lambda_m}(z) = \frac{1}{\mathcal{Z}} \sum_{k_m=1}^{K_m} q_{\phi_m}\left(z \mid \alpha_m^{(k_m)}\right) \tag{6.24}$$

where local parameters are

$$\lambda_m = \left\{\phi_m, \alpha_m^{(1:K_m)}, K_m\right\} \tag{6.25}$$

Thus, the formulation will be

$$\begin{aligned} \max \mathcal{L}_i(\theta_i, \phi_i, \lambda) &= \int q_\phi(z_i \mid x_i)\left[\ln p_\theta(x_i \mid z_i) + \ln p_\lambda(z_i) - \ln q_\phi(z_i \mid x_i)\right] dz_i, \quad \forall i \\ s.t.: p_\lambda(z) &\approx \frac{1}{\mathcal{Z}} \prod_{m=1}^{M} \underbrace{\left(\frac{1}{K_m} \sum_{k_m=1}^{K_m} q_{\phi_m}\left(z \mid \alpha_m^{(k_m)}\right)\right)}_{p_{\lambda_m}(z)} \\ z_i &= z, \quad \forall i \end{aligned} \tag{6.26}$$

Notice that we are proposing a Product of Experts (PoE) approach because of the inertia of Expectation Propagation but clearly other options and Mixtures of Experts (MoE) can be applied (for instance a mixture of Gaussians that we know are universal density approximators). Therefore, the idea is to solve locally this formulation and after that send to the server a new set of pseudopoints (realize that you do not need to send the points but you can adjust a mixture of Gaussians model and share the corresponding parameters).

6.5.2 Federated Data Augmentation

We have already explained the idea in the introduction section where we have describe the use of a common architecture denoted as Radial GAN. As we have seen, a loss function as a combination of adversarial component and cycle-consistency component forcing a common latent space.

After we have trained the translation functions $G_1; \cdots; G_M$ and $F_1; \cdots; F_M$, we create M augmented datasets

$$\widetilde{D}_i = \mathcal{D}_i \bigcup_{j \neq i} G_i\left(F_j\left(\mathcal{D}_j\right)\right) \tag{6.27}$$

and the predictors/classifiers $f_1; \cdots; f_M$ are then learned in a separate phase on these augmented datasets. This phase could use an arbitrary convex or neural network architecture and perform training in a completely completely standard way.

6.6 First results of FL

This section relates some of our preliminary results on the topic. It is important to remark the following ideas: federated learning in health scenarios has to be focused from a very particular perspective due to the

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

heterogeneity of different nodes with unequal number of available samples and many samples can be missed. If we have in one node very few samples we have two options:

- Train the model in other nodes with 'more ideal conditions' and copy it. However, if we have different nodes with very different set of features, the architectures to be used will be quite different. In other words, just applying one common structure probably will not work.
- Use virtual patients models to be communicated to other nodes and create long enough number of effective samples. The basic idea is that at local bad nodes we use these good samples to impute the local missing features and train with the virtual patients and the remote enhanced samples. In some cases, this process can be understood as a denoising procedure.

And after this federated processing, locally use the desired architecture for prediction, diagnosis, survival analysis, clustering... These considerations deserve two additional discussions:

- How are we going to evaluate the performance of the federated implementation in order to verify that there is a real improvement?
- Choose a simple setup with a suitable database that permits to choose the number of samples and the number of missing attributes per node.

Next scheme shows the main idea:

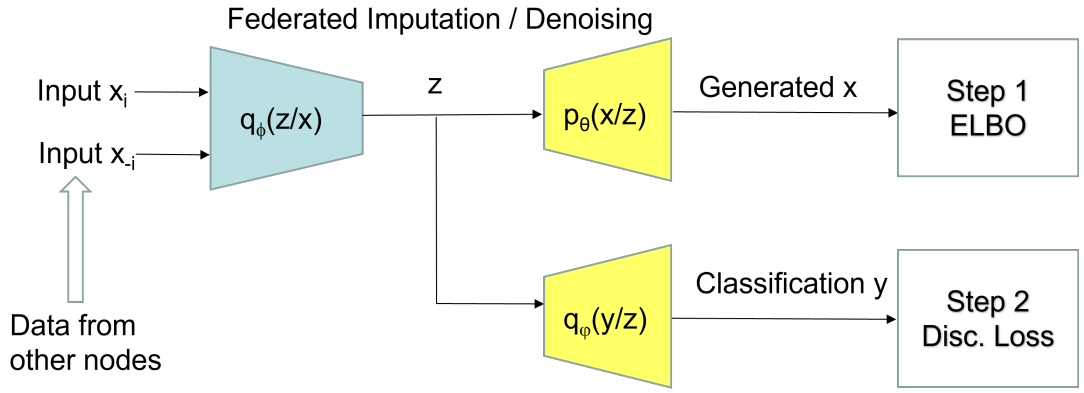


Figure 6.12: General scheme federated data augmentation.

6.6.1 How to evaluate the performance

We must pay attention to the evaluation procedure of the effectiveness of the federated implementation. This means that we should take under consideration the following ideas:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

- If nodes have similar number of samples with similar number of features we can be misled by apparent satisfactory performance. In this case the federated implementation could be useless without any practical effect.
- To avoid this misleading situation, it might be recommendable to consider very unbalanced scenarios to test the procedures.
  - Isolated good nodes with ideal conditions will give a performance upperbound.
  - Isolated bad nodes with very poor conditions will give a performance lowerbound
  - Bad nodes with the federated implementation must provide a performance in between and desirably closer to the upperbound.

6.6.2 Preliminary tests

6.6.2.1 MNIST results

In a first step we decided not to use the MDS database because it is very ad-hoc with not too many patients and with many missing entries. Also, the application of survival analysis is a complicated problem whose performance will depend on many aspects and can mask specific issues related to the federated implementation. Therefore, in order to evaluate the performance of the federated implementation using VAEs we have selected the MNIST data base with 10 handwritten digits while the purpose of the task is to implement a classification problem.

We have considered only two nodes, one (good one) with 500 digits with complete features. The second (bad one) only with 50 digits and with a percentage of missing pixels representing missing attributes that are replaced by binary noise (this way we control the imputation effect on the performance).

The federated implementation is based on the hypothesis that we have created an arbitrary large number of samples at the bad node using the ideal model of the good node. As the radial GAN architecture is not yet completely operative we have used the real samples.

For different imputation levels we have implemented the imputation algorithm described in [61] and we have plot the federated result in front of the two bounds: upper bound related to the performance of the good node isolated and lower bound the same for the bad node.

The following results deserve some explanations:

- Solid line represent the probability of detection using the latent space while the dashed-line represent the same result using the recovered samples. You can notice that using the latent space provide better results.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

- The numbers in first columns represent the original numbers, the reconstructed number in the ideal node, the same two results for the isolated bad node and the last two columns represent the case where the federated solution has been implemented.
- Next curves are related to different percentage of noisy (random imputed samples).

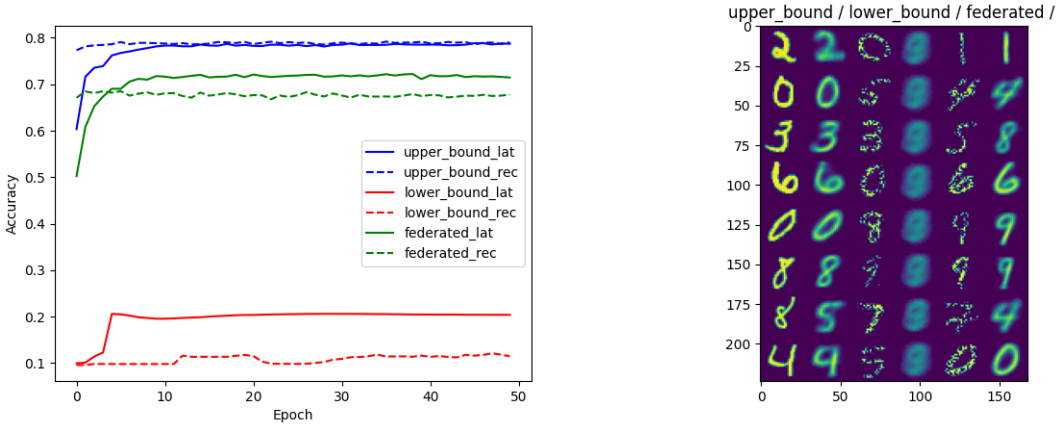


Figure 6.13: Imputation rate 0.5.

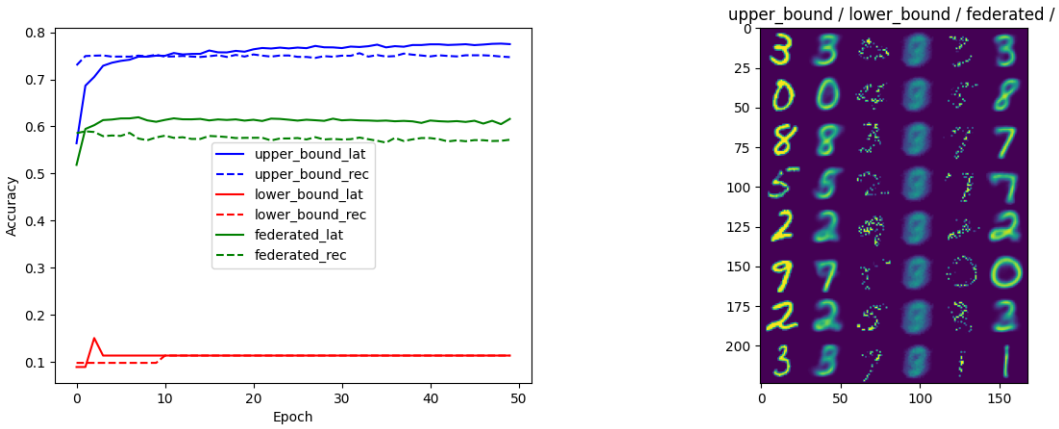


Figure 6.14: Imputation rate 0.7.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

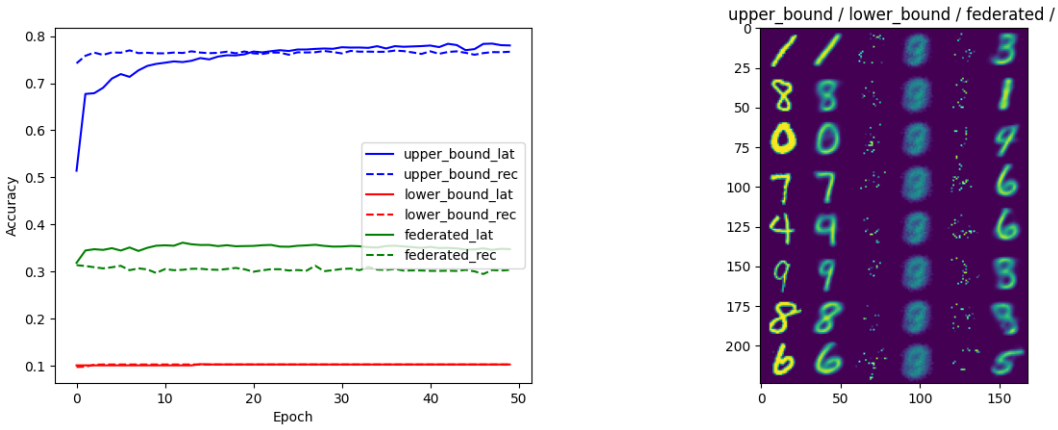


Figure 6.15: Imputation rate 0.9.

6.6.2.2 MDS results

Once we have achieved an implementation of the federated VAE that obtains considerable results for MNIST, we have proceeded to adapt the problem to the MDS database. In this case, we have also considered a classification problem targeting one gene, although of course other analyses can be done, but these are preliminary results. This gene is the SF3B1 gene, as it has a very frequent genetic mutation, although in this case the distinction between SF3B1 only mutated and unmutated SF3B1 (as they are completely identical if SF3B1 is excluded) will be completely lost.

We have followed the same process as in the previous case with only two nodes, a good one and a bad one, the latter with less information and a percentage of missing data. Unlike the MNIST database, this database contains very few samples (2048) so it makes sense to think that training it using this algorithm could lead to overfitting. This can be observed in the VAE training, as the training part of the VAE shows a significantly better performance than the testing part, clearly affecting the reconstruction part, as can be seen in the following accuracy curves 6.16. Even though the reconstruction phase shows worse results than the previous dataset, we have to focus in the latent part, where the federated dataset has an accuracy lower than the upper-bound one but higher than the lower-bound one. However, as it has been mentioned before, these are just preliminary ideas which have to be analyzed in depth.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

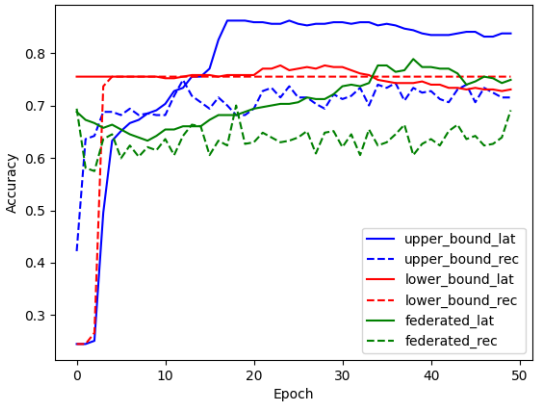


Figure 6.16: Imputation rate 0.5.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

7 UniBO contribution

Clustering is a technique that nowadays is widely used for patient stratification. In general, patients can be stratified according to some features obtained by multi-omics measurements (genomic, Imaging, etc) and, after this step, try to predict some clinical outcome. This approach is what has been applied in several studies concerning haematological diseases [65]. One of these approaches has been used for the Myelodysplastic syndrome (MDS) [66]. In this report, the authors used a particular type of clustering: the hierarchical Dirichlet process (HDP), a non parametric Bayesian approach to clustering grouped data. Due to the high number of studies that used this method, and also on the basis of the numerous clinical validations, the HDP has become a sort of ”gold standard” for clustering of hematological malignancies. It has to be noted that the choice of HDP is motivated also by the nature of the data set. As will be specified later, the data set format is, roughly speaking, formed by (for each patient) a set of genomic measurements (obtained by a gene panel), a set of cytogenetic measurements and by a third set of clinical variables. For the sake of simplicity, we will indicate the genomic and cytogenetic measurements with the common term of ”mutations”. According to this simplification, the first two sets of measurements can be coded by using a 0 (absence of mutation) or a 1 (presence of mutation). In this way, each row (each patient) will be represented as a realization of a multinomial distribution (the conjugate prior of the Dirichlet distribution). Another pros. of the HDP is that the number of clusters is determined automatically and there is no needing of assigning it a priori. Unfortunately, at our current knowledge, there is no a federated implementation for such algorithm, and also a ”de novo” federated implementation is not easy to develop. The current state of the art in the federated implementation of similar algorithms is the realization of a federated framework for the so called Latent Dirichlet Allocation (LDA) [67], a generative statistical model that has been used also for application in hematological malignancies. We observe that the HDP is the non-parametric Bayesian ”natural extension” of the LDA, and that the number of clusters can be learned from the data. For these reasons, all the federated implementations has to be compared not only with a centralized implementation, but also with the HDP results, in order to have a clinical interpretation and translation.

7.1 Clustering Methods

Clustering algorithms can be classified into the following categories, according to the method they use [68]. Partitioning methods: these methods attempts to directly decompose the data set into a set of disjoint clusters. Hierarchical clustering methods: these proceed successively by either merging smaller clusters into larger ones, or by splitting larger clusters and produce the so called dendrogram. Fuzzy clustering: these methods allow to the same object to be included into different clusters and with various degree of memberships. Density-based clustering: these methods group neighbouring objects of a data set into clusters based on density conditions. Moreover, clustering methods can be classified also on the basis of their membership

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

function; Soft and Hard clustering. A core aspect of clustering is dimensionality reduction, a key concept in machine learning and data science. Dimensionality reduction is important for feature extraction and data visualization, especially in the case of high-dimensional data. Among various methods, we can mention those based on nonlinear projection of high-dimensional data into a lower dimensional space, such as distributed stochastic neighbor embedding (t-SNE), and those based on manifold learning , such as Uniform manifold approximation and projection for dimension reduction (UMAP). Both these algorithms are, at our knowledge, not yet implemented in a federated way, but they can constitute a valuable benchmark.

7.2 MDS data description

The data on Myelodisplastic Syndrome (MDS) that are actually available between the GenoMed4All consortium are a combination of genomic, cytogenetic and clinical data. These data has been described in the deliverable D6.1, but we report here a brief description.The dataset is in a ”csv” format and we have that rows are patients while the columns are the variables (features) (for a complete description see the D6.1). These variables are organized as follows:

General and demographic variables

- Patient ID (EUROMDS patient labels)
- Gender (M/F)
- Age at data collection» (Age at diagnosis)

Clinical variables (prognostic scores)

- WHO 2016 subtype (WHO disease classification)
- IPSS risk group (disease classification according to IPSS)
- IPSSR score, IPSSR risk group (disease classification according to IPSSR)

Clinical-biological variables at diagnosis

These variables are composed of Hematochemical tests results on blood (cell counts and ferritin etc.) and bone marrow (count of bone marrow blasts an sideroblasts etc.) and a comorbidity score index.

Follow up and outcome variables
These variables are:

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

- Leukemia free survival (event, time to event)
- Overall survival (event, time to event)
- Acute Myeloid Leukemia (AML) adjusted Overall survival(event, time to event).

Cytogenetic variables

These data are grouped in 13 columns where is reported the presence/absence of a chromosomal alteration

Genomic variables - panel (yes=1/no=0)

The mutational status of 47 selected genes (gene panel for MDS)

HDP components

Patient specific Hierarchical Dirichlet Processes. The weights across 6 latent components are listed as reported in [66].

7.3 Federated Clustering

Clustering plays a central role in disease outcome prediction and intervention planning by bridging the gap between traditional, average based medicine, and fully personalized one. Clustering patients based on relevant attributes, but clinical, phenotypical, demographical and omics, allow to identify responses to therapy that might have been discounted as unpredictable variability before. To make full use of this ability it is necessary to employ extended clinical studies, with potentially tens of thousands of patients involved. This kind of studies are unfeasible for single centers and, even when possible, can't be generalized to the general population (for example at the European level) due to biases in socio-demographical and ethnicity characteristics of the population under study. This means that multi-center studies are fundamentals, but these kind of studies have issues with privacy and anonimization, when some of the information cannot be shared due to privacy concerns.

To circumvent this, federated methods can be applied, trying to identify cross-regional clusters without private information leaking between the centers. Sadly, most clustering methods cannot be directly extended in a federated setting, and even those than can might need to be changed substantially to accommodate for the different scenario.

Given the prominence of Deep Learning Approaches in federated approaches, we focused our exploration

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

on this kind of models. The model we developed is base on the Deep Embedding for Clustering (DEC) proposed by [69], implemented using the Flower framework[70] and extended for the MDS case. In the following a complete explanation on DEC model is provided, then the necessary modifications to make this model "federated" are proposed.

This model was inspired by parametric t-SNE and exploited the ability of autoencoders to project real data to the feature, or hidden, subspace, with a (commonly) strongly reduced dimensionality. This model, is able to simultaneously learn feature representations and cluster assignments . The main problem is clustering a set of n points $\{x_i \in X\}_{i=1}^{n}$ into k clusters, each represented by a centroid $\mu_j, j = 1, \ldots, k$. Instead of clustering directly in the *data space* X, the model first transform data with a non-linear mapping $f_\theta : X \longrightarrow Z$, where θ are the learnable parameters and Z is the latent *feature space*. The dimension of Z is typically much smaller than X in order to avoid the "curse of dimensionality". To parametrize f_θ, DNNs are a natural choice. In this data driven approach the optimization of the NN is crucial to obtain reliable results. DEC training is non trivial and subdivided in two main steps:

Step 1: parameters initialization and feature space identification by training a deep autoencoder.

Step 2: parameters optimization, i.e. clustering step, exploiting an auxiliary target distribution.

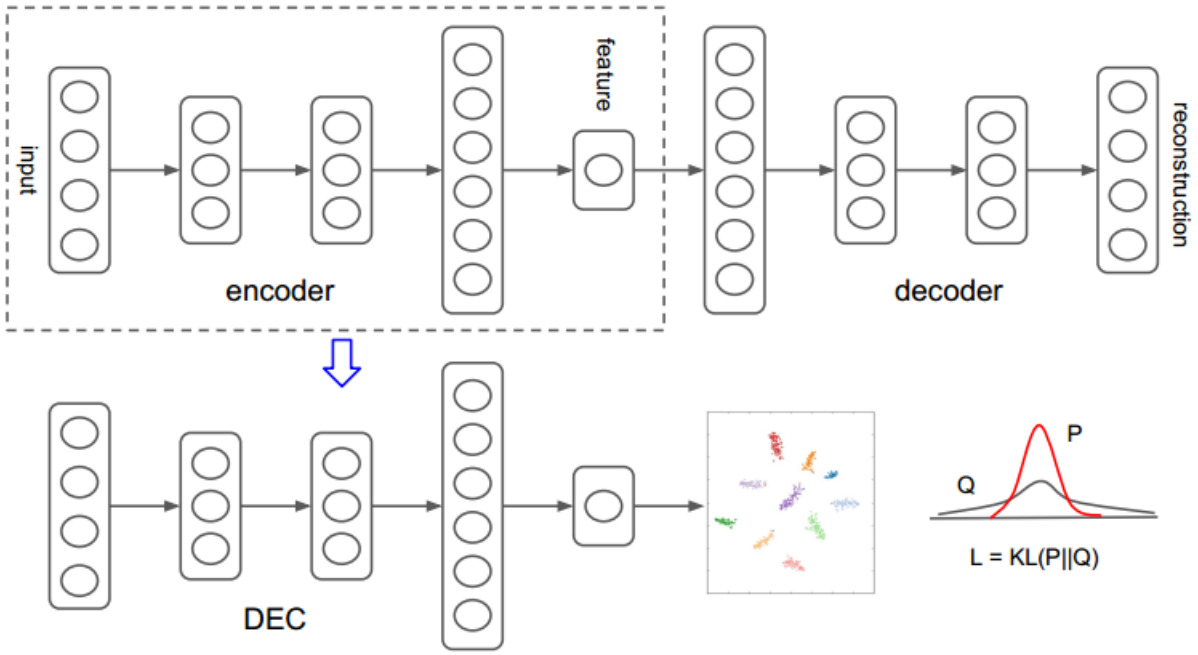


Figure 7.1: Diagram of the DEC model

During Step 1 the model tries to learn the feature representation of data, i.e. learns the non-linear mapping f_θ. A stacked autoencoder (SAE) is used because recent research has shown that they consistently produce semantically meaningful and well-separated representations on real-world datasets. Thus the unsupervised

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

representation learned by SAE facilitates in a natural way the clustering representation from DEC. In order to obtain the best representation possible, the SAE is initially pretrained as denoising autoencoder, that is a two layer neural network defined by:

$$\tilde{x} \sim Dropout(x) \tag{7.1}$$
$$h = g_1 W_1 \tilde{x} + b_1 \tag{7.2}$$
$$\tilde{h} \sim Dropout(h) \tag{7.3}$$
$$y = g_2 W_2 \tilde{h} + b_2 \tag{7.4}$$

where $Dropout(\cdot)$ is a stochastic mapping that randomly sets a portion of its input dimensions to 0, g_1 and g_2 are activation functions for encoding and decoding layer respectively, and $\theta = \{W_1, b_1, W_2, b_2\}$ are model parameters. Pretraining is performed by minimizing a reconstruction loss, usually Mean-Squared Error (MSE) $||x - y||_2^2$ is used for real-valued data, via the Stochastic Gradient Descent (SGD) optimizer. Then, the dropout layers are removed and the autoencoder is fine-tuned to optimize the reconstruction loss. The final result is a multilayer deep autoencoder with a bottleneck encoding layer in the middle. The decoder layers are then discarded and only encoder layers are used as initial mapping between the data space and the feature space.

Step 2 is then performed to refine the parameters found so far. The data is passed through the initialized DNN to get embedded data points and then k-means clustering is performed in the feature space Z to obtain k initial centroids $\mu_j, j = 1, \ldots, k$. Following this initialization the training proceed iterating between two tasks: computing the auxiliary target distribution and minimize the Kullback-Leibler (KL) divergence to the computed target distribution. The choice of the Student's t-distribution as a kernel to measure the similarity between embedded point and centroid seems natural (including the connection to the T-SNE encoding), and it is computed as:

$$q_{ij} = \frac{(1 + ||z_i + \mu_j||^2/\alpha)^{\frac{\alpha+1}{2}}}{\sum_{j'} (1 + ||z_i + \mu_{j'}||^2/\alpha)^{\frac{\alpha+1}{2}}} \tag{7.5}$$

where $z_i = f_\theta(x_i) \in Z$ corresponds to $x_i \in X$ after embedding, α are the degrees of freedom of the Student's t-distribution and q_{ij} can be interpreted as the probability of assigning sample i to cluster j (i.e., a soft assignment). They let $\alpha = 1$ since it is not possible to cross-validate α on a validation set in the unsupervised setting, and learning it is superfluous.

For an efficient convergence of the model, choosing the proper target distribution P is very important. It needs to possess the following properties:

1. Strengthen predictions (i.e., improve cluster purity).
2. Put more emphasis on data points assigned with high confidence.
3. Normalize loss contribution of each centroid to prevent large clusters from distorting the hidden feature space.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

During the KL divergence optimization, the model updates the deep mapping and refine the cluster centroids by learning from current high confidence assignments using an auxiliary target distribution. Specifically, the model is trained by matching the soft assignment to the target distribution. The objective is defined as a KL divergence loss between the soft assignments q_i and the auxiliary distribution p_i as:

$$L = \mathrm{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{7.6}$$

where p_{ij} is computed as:

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij}^2 / f_{j'}} \tag{7.7}$$

where $f_j = \sum_j q_{ij}$ are the soft cluster frequencies. The training strategy can be seen as a form of self-training. As in self-training, we take an initial classifier and an unlabeled dataset, then label the dataset with the classifier in order to train on its own high confidence predictions. Indeed, in the original paper the authors observed that DEC improves upon the initial estimate in each iteration by learning from high confidence predictions, which in turn helps to improve low confidence ones.

The method jointly optimizes cluster centers and DNN parameters θ using Stochastic Gradient Descent (SDG) with momentum.

$$\frac{\partial L}{\partial z_i} = \frac{\alpha + 1}{\alpha} \sum_j (1 + \frac{||z_i - \mu_j||^2}{\alpha})^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j), \tag{7.8}$$

$$\frac{\partial L}{\partial \mu_j} = -\frac{\alpha + 1}{\alpha} \sum_i (1 + \frac{||z_i - \mu_j||^2}{\alpha})^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j). \tag{7.9}$$

The gradients $\frac{\partial L}{\partial \theta}$ are then passed down to the DNN and used in standard backpropagation to compute the DNN's parameter gradient $\frac{\partial L}{\partial \theta}$. For the purpose of discovering cluster assignments, the procedure is stopped when less than a pre-defined number of points change cluster assignment between two consecutive iterations.

7.4 Results on MDS data

7.4.1 Federated training of the DEC models

Training DEC model in a federated fashion needs some modifications w.r.t the centralized version. Step 1 of training (initialization of parameters via SAE) is performed by exploiting FedAvg[71], one of the most popular algorithms for aggregating weights. Kullback-Leibler Divergence (KLD) minimization step also is performed using FedAvg. The initialization of the cluster centroids needs a deeper discussion, in fact it is not guaranteed that running k-means at clients place will undercover the same clusters centroids for every client. One have to keep in mind that clusters' inititialization does not need to be extremely precise because centroids will be optimized during KLD minimization step, but on the other hand they cannot be random

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

since DEC model works better if clusters are already enough separated in the feature space. In order to solve the problem of "federating" the clusters' initialization, the following algorithm is proposed.

Every client identify its clusters' centroids as in the centralized version. The initialization is done with 25 different random seeds, run for 300 epochs and looking to obtain all the k centroids. These centroids are then sent to the central node to be aggregated. To do the aggregation, at server place, the algorithm randomly chooses one of the clients centroids as the starting point, then add iteratively to the list of aggregated centroids the farthest centroid available from the list of all clients' centroids until k centroids are collected. This provide a full coverage of the features space from the initial centroids. The the final list of aggregated centroids is sent back to the clients to initialize the clustering layer.

The clustering layer treats them as weights to be optimized, and then trains for 10'000 epochs, updating every 100 epochs the auxiliary distribution and the soft assignments. The optimization is performed using SGD with a fixed learning rate of 0.1.

Other important modifications to the network architecture were performed to adapt DEC model to MDS data. The first dropout layer for noising data were substituted with a Random Flipping layer that randomly sets a portion of input data to 0 or 1 respecting the overall distribution of 0 and 1 in the entire dataset. The dimensions of the the fully connected layers that compose the autoencoder were reduced from {500,500,2000,10}, used in the paper for MNIST dataset, to {26, 26, 100, k}, where k is the number of cluster to identify. Moreover a Binary Cross-Entropy loss was chosen for reconstructing these binary data, as the outputs of the autoencoder could be interpreted as probability to assume values 0 or 1.

The entire dataset of 2043 sample patients is divided in equal number between clients in several different partitions, and every client is weighted in the average only by the number of samples:

- 2 clients set up, client 1 has 1021 samples, client 2 has 1022 samples
- 4 clients set up, client 1,2,3 have 510 samples, client 4 has 513 samples
- 6 clients set up, client 1,2,3,4,5 have 340 samples, client 6 has 343 samples
- 8 clients set up, client 1,2,3,4,5,6,7 have 255 samples, client 8 has 258 samples

Clients' local samples were subdivided in train-test set, respectively 80-20% of the samples, at clients' place.

To estimate the results of the clustering procedures we estimated the quality of the SAE and the Clustering quality. For the SAE the metrics were:

- Accuracy between original data and reconstructed data;
- Rounded accuracy between original data and rounded reconstructed data;
- G metric, ratio between train loss and evaluation loss.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

Identifying letter	AE epochs	Dropout rate	Random Flip rate	Unit Norm
a	2500	0.2	0.2	No
b	2500	0.05	0.05	No
c	2500	0.2	0.2	Yes
d	2500	0.1	0.1	Yes
e	2500	0.05	0.05	Yes
f	2500	0.01	0.01	Yes
g	5000	0.01	0.01	Yes

Table 7.1: Parameters configurations for the tested models.

For the clustering model the metrics were:

- Cycle accuracy of labels assignment, the accuracy between predicted assignment of real data and predicted assignment of reconstructed data;
- Number of samples whose label prediction change with regard to the previous epoch;
- G metric, the ratio between train loss and evaluation loss,

7.4.2 Effect of parameters on model performances

To assess the sensitivity of the results to the training parameters we tested different model configurations, identified by the letters a—g. For each model we changes the AE, the dropout rate, the random flip unit and the use of unit normalization. The individual models are summarized in Tab 7.1. Different features space dimension (k) were tested, from 2 to 12, for each model, and the results for this tests can be seen in Fig. 7.2, Fig. 7.3, Fig. 7.4. As one can see the only major difference can be observed in the G metric between the models with no normalization (a and b) and those with normalization (the others).

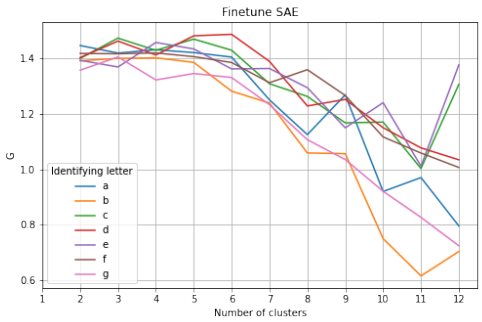


Figure 7.2: SAE for the different models configurations with varying numbers of clusters

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

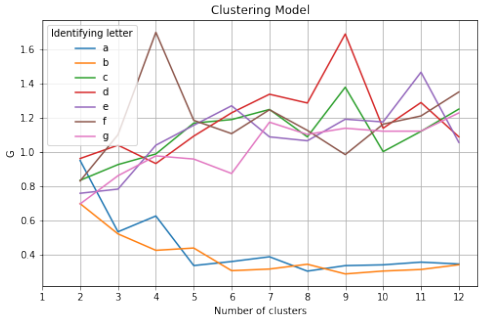


Figure 7.3: G metric for the different models configurations with varying numbers of clusters. One can observe the difference between the models with normalization (c, d, e, f, and g) and those without (a and b)

Figure 7.4: Cycle accuracy for the different models configurations with varying numbers of clusters

The results of the federation, by dividing the dataset in equal (or as equal as possible) splits, is detailed in Fig. 7.5, Fig. 7.6, Fig. 7.7 based on the number of cluster centroids selected. As one can see, there is no significant loss of clustering quality.

7.4.3 Comparison with the Hierarchical Dirichlet Process clustering

Currently the Hierarchical Dirichlet process can be considered the gold standard for cluster identification in a single core setting, but it is non trivial to distribute in a federated setting. We therefore used the previous results of the HDP as the ground truth to assess whether the new methods could perform in a comparable way.

We considered two cases, one with 6 and one with 8 clusters, being those the previously obtained results on the MDS dataset, with a varying number of nodes (1, 2, 4, 6, and 8) in the federation. Clearly with one node is intended a centralized version of the training. In one case the silhouette score was confronted between the two methods, while in the other 3 scores were considered to assess the agreement between the two different clustering. In Fig. 7.8 one can see the results with 6 clusters, while in Fig. 7.9 one can see the results with 8

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

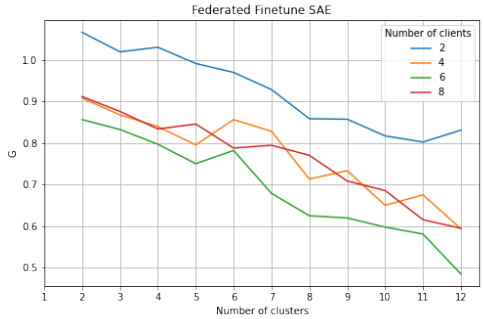


Figure 7.5: SAE for the different number of federated clients with varying numbers of clusters

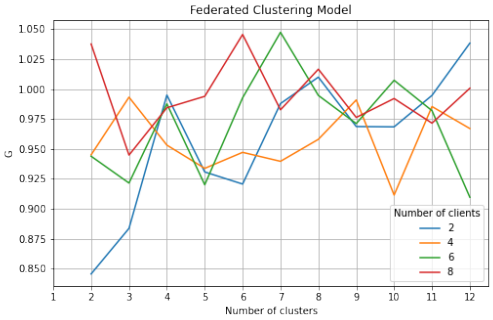


Figure 7.6: G metric for the different number of federated clients with varying numbers of clusters.

clusters.

One can observed that the Silhouette scores are significantly improved, even if is to be expected given that the DEC method is optimizing for that score, albeit indirectly. The agreement measures show that the agreement, even if not optimal, is consistent with different federation layouts, highlighting the robustness of the method to the federation procedure.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

Figure 7.7: Cycle accuracy for the different number of federated clients with varying numbers of clusters

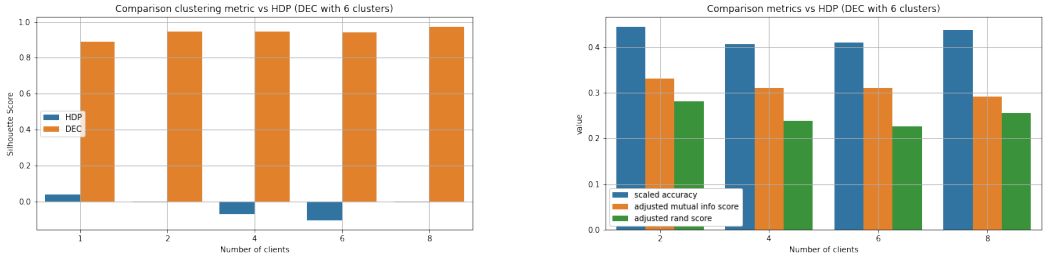


Figure 7.8: left: Silhouette score of both DEC and HDP for 6 cluster centroids with different federation divisions; right: Agreement scores between DEC and HDP for 6 cluster centroids with different federation divisions

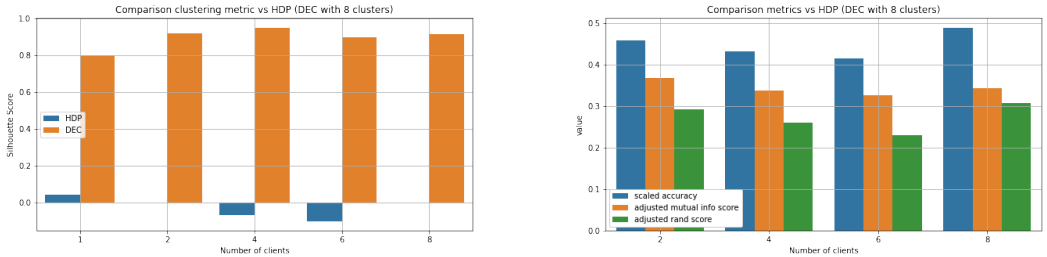


Figure 7.9: left: Silhouette score of both DEC and HDP for 8 cluster centroids with different federation divisions; right: Agreement scores between DEC and HDP for 8 cluster centroids with different federation divisions

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

8 UniTO contribution

The goal of survival analysis is to predict the occurrence time of some event of interest, which can be an adverse event for patients in a clinical setting. To this aim, the Cox proportional hazards model is routinely used by clinicians for prognostic purposes. However, clinical data are often limited in individual centres and aggregating local datasets is often impossible due to their sensitivity and strict privacy regulations, requiring federated approaches. Traditional survival models do not fit a federated learning framework, as their loss function is non-separable with respect to the samples. In addition, most of the state-of-the-art methods assume only linear relationships among covariates. Therefore, we explored the most recent Neural Network and Machine Learning techniques to develop nonlinear survival analysis models, implementing them through a federated approach.
This chapter is organized as follow: section 8.1 is dedicated to the description of the standard Cox Proportional Hazard model and its ”deep” version, using Neural Networks; section 8.2 introduces the proposed Federated Learning implementations of the algorithms; section 8.3 shows the obtained performance of the methods applied to Myelodysplastic syndrome (MDS) data and section 8.4 reports our conclusions and the directions for future works.

8.1 Survival Analysis

Given a group of individuals susceptible to experiencing an event of interest, survival analysis techniques seek to infer the probability distribution for the time of that event for each individual. Obtaining these times-to-event requires patients to be followed in long observational studies, which typically last years. Due to this long duration, patients frequently drop out from studies before any event occurs. In this case, one can only know that the event did not occur before the dropping time, which is known. This partial knowledge is known in survival analysis as right censoring. An example is presented in Figure 8.1.
Here for patients A,C and E it is unknown whether they did or did not experience an event after termination of the study, so they are considered *censored*. The only available information is that they are event-free up to the last follow-up. Therefore, the exact time of an event could only be recorded for patients B and D, which are considered as *uncensored*. The goal of survival analysis is to model the probability distribution of the actual event time $T_i \in \mathbb{R}_+$ for each individual $i \in [N] = 1, ..., N$. The hazard function $h(t, x_i)$ represents this distribution through an approximate probability that an event, characterized by a vector $\mathbf{x}_i \in \mathbb{R}^p$ of features (e.g. clinical, demographics, genetic features, etc.), occurs in the time interval $[t; t + \Delta[$, under the condition that an individual would remain event-free up to time t:

$$h(t, \mathbf{x}_i) = \lim_{\Delta t \to 0} \frac{P(t \leq T_i < t + \Delta t \mid \mathbf{x}_i, T_i \geq t)}{\Delta t} \geq 0$$

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

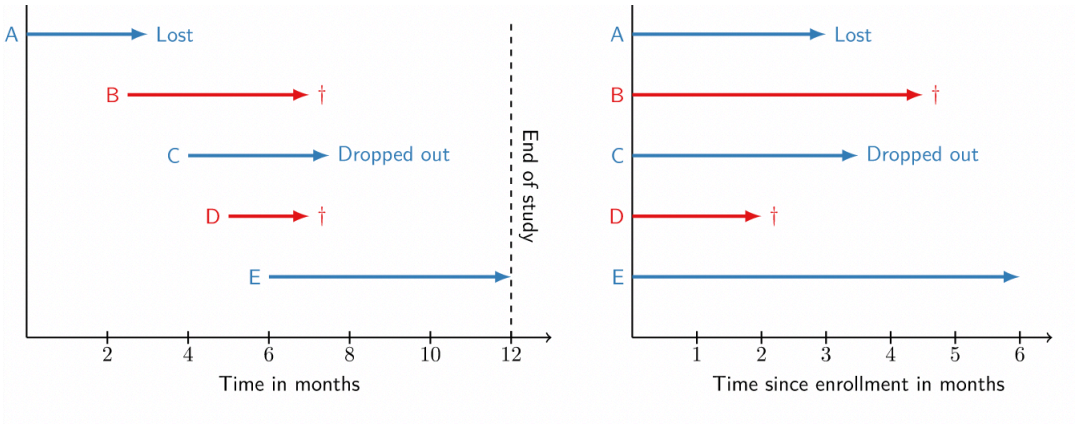


Figure 8.1: Examples of right censoring.

8.1.1 CoxPH model

Cox proportional hazards model is the most widely used technique to learn from censored survival data. It assumes that the hazard function can be factored as:

$$h\left(t \mid x_{i1}, \ldots, x_{ip}\right) = h_0(t) \exp\left(\sum_{j=1}^{p} x_{ij}\beta_j\right) \Leftrightarrow \log \frac{h\left(t \mid \mathbf{x}_i\right)}{h_0(t)} = \mathbf{x}_i^\top \beta \tag{8.1}$$

where $\beta \in \mathbb{R}^p$ are the coefficients associated with each of the p features, while $h_0(t)$ is the baseline hazard function, common to all the individuals in the group and dependent on time only. In addition, the model implies that the risk ratio between individuals does not depend on time. Therefore, considering only relative comparisons, the baseline hazard function does not need to be specified. The function $h_0(t)$ and the parameters β can be estimated independently from each other. For the latter, the minimization of the negative Cox partial log-likelihood function is performed, which only compares the relative risk ratios and it is based on the probability that the i-th individual experiences an event at time t_i, given that there is one event at time point t_i.

Let $\mathcal{R}_i = \{j \mid t_j \geq t_i\}$ be the risk set, i.e., the set of subjects who remained event-free shortly before time

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

point t_i, and $I(\cdot)$ the indicator function, then we have:

$$\begin{aligned}
&P\left(\text{subject experiences event at } t_i \mid \text{ one event at } t_i\right) \\
=&\frac{P\left(\text{ subject experiences event at } t_i \mid \text{ event-free up to } t_i\right)}{P\left(\text{ one event at } t_i \mid \text{ event-free up to } t_i\right)} \\
=&\frac{h\left(t_i \mid \mathbf{x}_i\right)}{\sum_{j=1}^{n} I\left(t_j \geq y_i\right) h\left(t_j \mid \mathbf{x}_j\right)} \\
=&\frac{h_0\left(t_i\right) \exp\left(\mathbf{x}_i^{\top} \beta\right)}{\sum_{j=1}^{n} I\left(t_j \geq t_i\right) h_0\left(t_j\right) \exp\left(\mathbf{x}_j^{\top} \beta\right)} \\
=&\frac{\exp\left(\mathbf{x}_i^{\top} \beta\right)}{\sum_{j \in \mathcal{R}_i} \exp\left(\mathbf{x}_j^{\top} \beta\right)}
\end{aligned}$$

By multiplying the conditional probability from above for all patients who experienced an event, and taking the logarithm, we obtain the partial likelihood function:

$$\widehat{\beta} = \arg\max_{\beta} \log PL(\beta) = \arg\max_{\beta} \sum_{i=1}^{n} \delta_i \left[\mathbf{x}_i^{\top}\beta - \log\left(\sum_{j \in \mathcal{R}_i} \exp\left(\mathbf{x}_j^{\top}\beta\right)\right)\right] \tag{8.2}$$

We also notice that with this formulation the baseline hazard function h_0 is canceled out so it does not need be defined for finding the coefficients β.

8.1.2 DeepCox

Cox proportional hazards model assumes that the log hazard is decomposed into and time-independent linear predictor of features vector $\mathbf{x}$ as $log(h(t \mid \mathbf{x})/h_0(t)) = \mathbf{x}^{\top}\beta$. However, the assumption is often too simplistic in real-world clinical data, as the patients' features could contribute through more complicated non-linear functions to the definition of risk scores. In order to capture this behaviour, the model can be easily extended to the non-linear case by replacing the linear predictor $\mathbf{x}^{\top}\beta$ with the output of a neural network with parameters $\boldsymbol{\Theta}$. Therefore, the partial log-likelihood expressed in (8.2) will be the loss function used to train the neural network. In particular, due to the event indicator δ_i, only uncensored data points are considered to compute the loss. A schematic representation of the model is presented in Figure 8.2.
This idea was originally proposed in the work of Faraggi and Simon [72] back in 1995, where they explored multilayer perceptrons, but the same loss can be used in combination with more advanced architectures such as convolutional neural networks or recurrent neural networks, thus extending the Cox model to different types of patients' data such as diagnostic images or temporal records. Therefore, it is natural to also use the same loss function in the era of deep learning.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

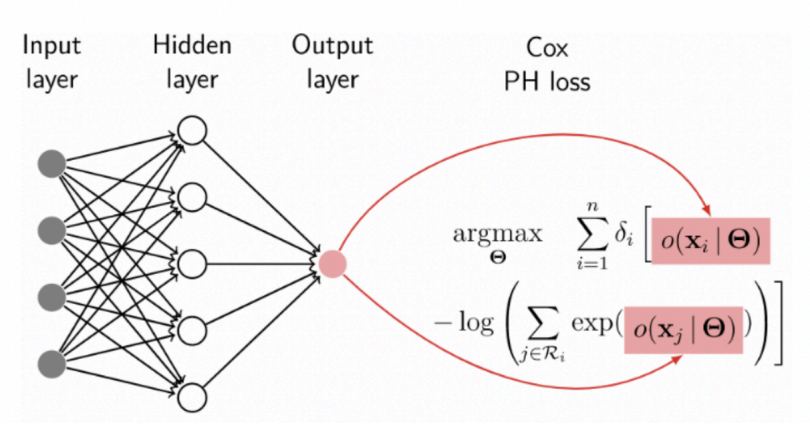


Figure 8.2: DeepCox neural network.

8.1.3 Evaluating Survival Models

The most frequently used evaluation metric for survival models is the Concordance Index (CI) and it will be considered as reference metrics to evaluate and compare the performances of the proposed models. CI is a measure of rank correlation between the predicted risk scores $\hat{f}$ and it can be considered as a generalization of the area under the ROC curve (AUC) that can take into account censored data. It is defined as the ratio of correctly ordered (concordant) pairs to all the comparable pairs. Two samples i and j are comparable if the sample with lower observed time experienced an event, i.e., if $t_j > t_i$ and $\delta_i = 1$ where δ_i is a binary event indicator $\delta \in \{0;1\}$ that defines the status of censored ($\delta = 0$) or uncensored ($\delta = 1$). A comparable pair (i, j) is concordant if the estimated risk $\hat{f}$ by a survival model is higher for subjects with lower survival time, i.e., $\hat{f}_i > \hat{f}_j \wedge t_j > t_i$, otherwise the pair is discordant.

8.2 Federated strategies

Designing Machine Learning algorithms in a federated privacy-preserving framework is challenging for both training and evaluating the model. In our application, the survival data from the patients are supposed to be available from different hospitals and cannot be moved across centres. This induces the development of algorithms able to jointly learn a predictive model from these data by sharing as little information as possible from the centres. This can be accomplished in multiple ways, depending on the kind of model one is concerned with. For neural networks, the internal weights can be shared without restrictions. However, it is not clear which is the most efficient strategy to aggregate and update the models at each federated step, since it depends on a wide range of factors such as the specific learning task, the data type or the statistical heterogeneity of the centres. To this purpose, we have investigated three possible federated learning architectures (*Sequential Federated*, *Weights Averaging* and *Ensemble Federated*), which have been tested on a survival dataset of MDS patients already described in the previous chapters. The implemented techniques are not limited to this specific application, but they can be also used for other regression or classification

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

tasks.

8.2.1 Federated CoxPH

Before going deeper into the analysis of the federated strategies, it is worth making some observations about a common shortcoming when building Federated CoxPH models. The loss function (8.2) differs from traditional losses since it is non-separable at a single patient level. Indeed, the internal sums of exponential terms are performed over the risk sets of each patient i, thus supposing that their data are always available. However, this is not the case for the federated learning paradigm, where we can only access the patients' belonging to the same centre as i when computing the loss. Hence, the resulting loss function will be incomplete and, in principle, it could deviate significantly from the true value. For practical purposes, computing the CoxPH loss over a subset of the whole dataset is usually fine as the subset contains several uncensored samples, otherwise the outer sum in the partial likelihood function would be over an empty set. Specifically, if we represent all the risk sets with a squared boolean matrix where the i-th row denotes the risk set of the i-th instance (i.e., at the observed time $t_j > t_i$), computing the federated risk sets for each centre is like applying a mask to the whole dataset considering only the respective diagonal blocks of the matrix as represented in the figure below. In particular, there are two possible scenarios (Figure 8.3): *balanced* or *unbalanced* centres. One balanced which refers to the case in which all the hospitals have the same sizes in terms of number of patients (left panel of the figure), the other one unbalanced representing a non-uniform distribution of patients among the centres (right panel of the figure).

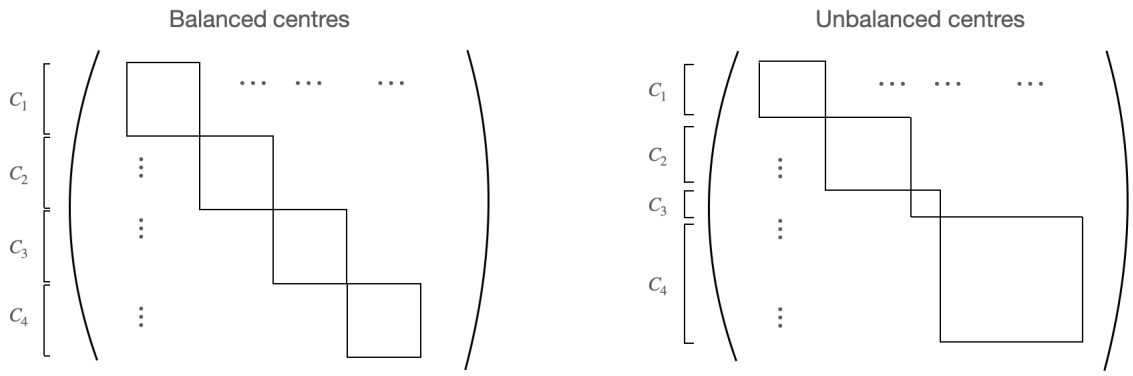


Figure 8.3: Example of masked risk set matrices in the balanced and unbalanced federated scenarios.

8.2.2 Sequential Federated

This architecture is inspired by the *FedAvg* algorithm, proposed by Google's researchers in their seminal work in 2017 [1]. The general idea is to create a single model that moves sequentially across the centres during the training phase; each centre is treated as a mini-batch with the only difference that the samples

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

of each batch are fixed and not randomly shuffled at each epoch. A schematic representation of the model is presented below: First, we randomly initialize the model's parameters M_0, e.g. the weights of a neural

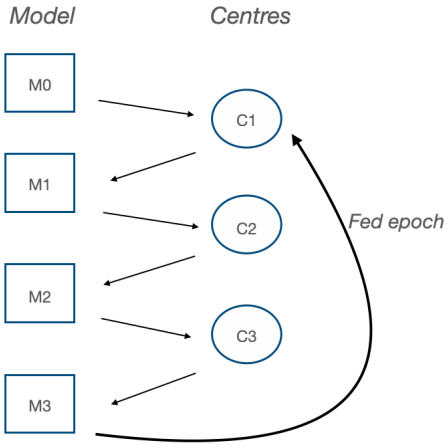


Figure 8.4: Sequential Federated architecture.

network, in a central server. M_0 is sent to the first centre C_1 where an internal training is performed for a fixed number of epochs, using only its own patients' data. Then, the updated model M_1 is sent back to the central server, which directly shares it with another centre without making any operation. The process is repeated until the model has seen all the participating nodes and the whole procedure is also repeated for a certain number of *federated epochs*, randomly shuffling the order of the centres each time.

8.2.3 Weights Averaging

While in the previous strategy a single model is sequentially trained across the centres, in this architecture a common model is trained simultaneously through all the K centres, by performing a weights aggregation. Therefore, the same model is separately trained through all the centres at the same time, with a fixed number of internal epochs. Then, all the local updates are sent to a central server, which perform a weighted average of the parameters, giving more importance to the nodes with a higher number of patients. This final global update is then used as a new starting point and the procedure is repeated for each federated epoch, as represented in Figure 8.5.
We highlight that *Sequential Federated* and *Weights Averaging* strategies are the limit cases of the *FedAvg* algorithm, respectively setting the fraction of selected nodes for each step to $C = 0$ (the updates are performed one centre at a time) and $C = 1$ (all the centres are used for each update).

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

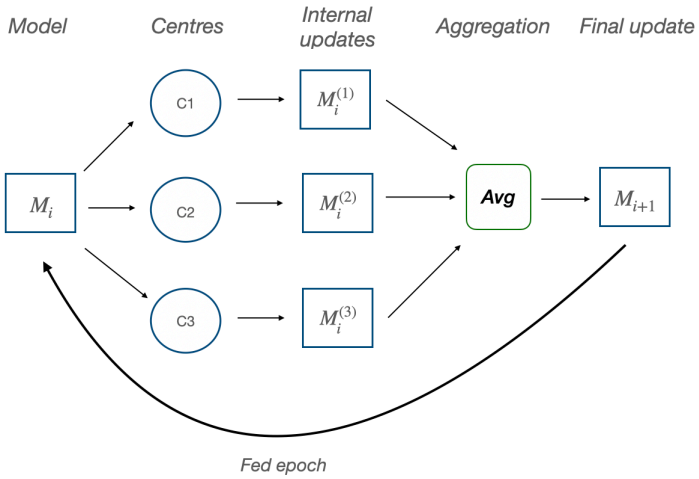


Figure 8.5: Weights Averaging Federated architecture.

8.2.4 Ensemble Federated

The last investigated method is based on a simple ensemble technique, where model outputs from different centres are aggregated to provide a final prediction. Therefore, K models are generated, with K corresponding to the number of centres, and they are trained separately at the same time. All local outputs y_k are then shared to a central server and aggregated together through a weighted average, to provide a final prediction $\hat{y}$ (Figure 8.6).

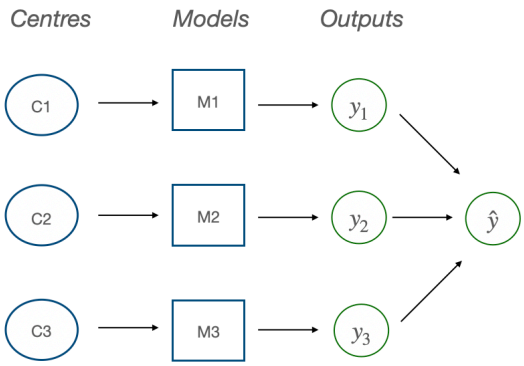


Figure 8.6: Ensemble Federated architecture.

As we can notice, this ensemble technique is different from the other federated strategies since there are no federated epochs and K local models are created instead of a global one. However, since data are never shared across the nodes to compute the final outputs, this method still deserves to be analysed.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

8.2.5 Centre-based KFold Cross-Validation

To test the models' performance it is necessary to design a validation strategy that satisfies all the privacy-preserving requirements. To this purpose, we propose a Federated KFold Cross-Validation. It is based on the same principles of the standard KFold Cross-Validation and it can be used to find the optimal hyperparameters of the model. In particular, for each fold, the data of a selected centre are excluded from the training and used as an internal test. To evaluate the methods we will use the CI, which is a relative score since it depends on the ratio of concordant pairs to comparable ones. Thus, this metric needs to be computed not at a single-centre level, but over the whole dataset in order to be reliable. Therefore, during the cross validation, the predictions obtained for each internal test will be concatenated to compute the CI for the entire population, which represents the final cross-validation score for a specific combination of hyperparameters.

8.3 Experimental setup and results

8.3.1 Model architecture

The baseline model is a Feed-Forward Neural Network with 2 hidden layers composed of 8 and 4 nodes respectively. To facilitate the optimization process during the training, we have set a learning rate that decades if $Loss^{(i+1)} > Loss^{(i)}$, such that $lr^{(i+1)} = \max\left(lr^{(i)}/\sqrt{i+1},\ 0.001\right)$. In addition, we also added $L1$ and $L2$ regularizations on the weights of the first hidden layer in order to prevent overfitting. The hyperparameters selection was performed through the cross-validation described above. In particular, our grid search included: federated epochs, internal epochs, number of internal batches and initial learning rate.

8.3.2 Federated settings and validation

To test our models we used the MDS data from Bersanelli et al. [66], which includes 2,043 patients and a total of 70 covariates of different types (demographics, clinical, genetic and cytogenetic features). For the time-to-event prediction task we considered the Overall Survival, i.e., the death of the patient. 28.9% of the patients are uncensored, while for the others the time of the last follow-up visit is reported.

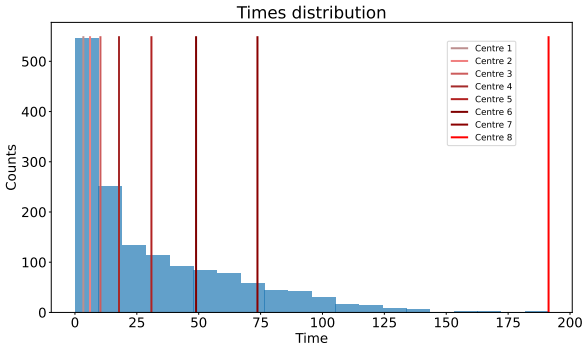
Data were standardized, putting all values in the $[0;1]$ interval, before being fed into the neural network.
For the models' evaluation, we divided the dataset into training and test sets (with a $75\% - 25\%$ split). Then, to simulate a federated framework, we distributed the training data across 8 artificial centres, but still storing them in the same memory unit of a single machine. For the simulation we explored different federated settings in order to test the robustness of the proposed architectures to real-world data distributions. In particular, we simulated 4 scenarios in the creation of the fictitious centres:

- **Balanced IID nodes:** Centres have the same number of patients and IID distributions of covariates

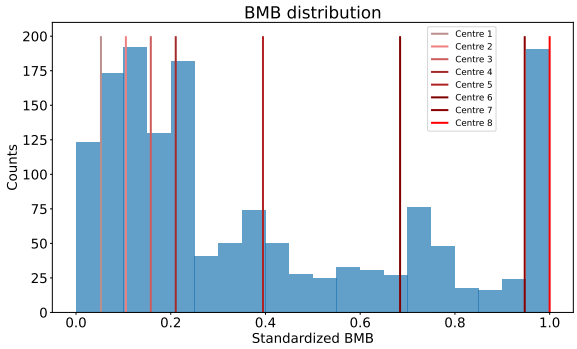
This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

and survival times;

- **Unbalanced IID nodes:** Centres have different number of patients (with a hub node containing 40% of patients) but maintain the same distributions of covariates and survival times;
- **Balanced Non-IID survival times:** Centres have the same number of patients but different non-overlapping survival times distributions (see Figure 8.7a);
- **Balanced Non-IID feature (BMB):** Centres have the same number of patients but different non-overlapping distribution of a single feature (see Figure 8.7b). In this case we performed a feature importance analysis over the whole training set and chose the feature with the higher predictive value for the Neural Network, which was the Bone Marrow Blasts percentage (BMB).



(a) Non-IID survival times distributions across the 8 centres.



(b) Non-IID BMB distributions across the 8 centres.

Note that for the Non-IID scenarios we decided to set hard thresholds to the distributions of survival times and BMB, when creating the centres. These situations should be intended as the worst case scenarios for the Non-IID settings.

Finally, we applied the Centre-based KFold Cross-Validation for hyperparameters selection only on the training data, while we used the test set for the evaluation of the selected model (*blind* test). The test set can be considered as an isolated centre with an unbiased distribution of the features. The evaluation process has been repeated 20 times, applying different train-test splits.

8.3.3 Preliminary results

The performances of our models on the blind test set are reported in terms of CI, which is computed through the risk scores obtained for each patient with the trained neural networks.
To appreciate the advantage of the federated approaches we have also evaluated the performance of our models for the corresponding non-federated boundary cases, in order to establish an upper bound and a lower bound for the 4 federated settings. In particular, the lower bounds are represented by the situations in which centres are isolated and models are built using only their own data; on the other hand, the unique

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

upper bound is the centralized case, which refers to the case where all the data are simultaneously available, as if they could be shared to a central server without any restrictions.
These results are summarized in Tables 8.1, 8.2, 8.3.

	Concordance	SD
Centralized case (Upper bound)	**0.75**	0.02

Table 8.1: Non-Federated Centralized case.

IID centres	**Balanced sizes**		**Unbalanced sizes**	
	Concordance	SD	Concordance	SD
Isolated nodes (Lower bound)	**0.62**	0.06	**0.62**	0.07
Sequential	0.75	0.02	0.74	0.02
Weights Averaging	**0.75**	0.01	**0.75**	0.02
Ensemble	0.74	0.02	**0.75**	0.02

Table 8.2: Federated IID centres.

Non-IID centres	**Non-IID survival times**		**Non-IID BMB**	
	Concordance	SD	Concordance	SD
Isolated nodes (Lower bound)	**0.55**	0.06	**0.59**	0.05
Sequential	0.60	0.07	0.70	0.03
Weights Averaging	**0.68**	0.03	**0.73**	0.05
Ensemble	0.59	0.05	0.70	0.03

Table 8.3: Federated Non-IID centres.

As we can see from Table 8.2, in the IID-centres scenarios all the federated approaches achieve good scores with respect to the lower bounds, showing a significant improvement of the Concordance Index. This result also applies in the case of unbalanced centres, with no relevant differences.
On the other hand, there is a deterioration in the performance for the Non-IID scenarios, but still achieving scores significantly higher than the case of isolated nodes. In particular, Table 8.3 shows that the *Weights Averaging* federated strategy is more robust to both Non-IID times and Non-IID BMB settings with respect to *Sequential* and *Ensemble* strategies.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

8.4 Conclusions and further work

Here we have introduced a preliminary federated implementation of the survival analysis through a CoxPH model combined with a simple neural network and designing three possible federated strategies, which can be also used for different applications. The model, tested on MDS data, achieved promising results and we believe it is worth investigating the proposed techniques further through also other non-linear learning algorithms. Therefore, we aim at exploring in detail more complex neural network architectures, suitable also for different types of input data (such as medical images or temporal records). Moreover, we will also enrich the proposed federated strategies by considering more realistic scenarios, where some centres are unavailable for training in some federated epochs.

Regarding the models' evaluation, we will test also the performance with imputed data in order to include more variables and a greater cohort of patients, and considering other types of events like the Leukemia-Free Survival (LFS) and the Event-Free Survival (EFS), available from the MDS dataset.

Finally, we plan to analyze the performance of our FL strategies in other non-IID settings, i.e., with introduced heterogeneity in the local data distribution, since a performance degradation could be generated due to weight divergence of the local models resulting from non-IID scenarios, as pointed out by authors in [73, 74].

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

9 UCPH contribution

9.1 Motivation

Medical data sets are often small, heterogeneous, noisy, and have missing values. Additionally, when data come from different sources, they may have systematic differences due to differences in instrumentation and procedures at the different clinics. Therefore, the models used must be robust to these complicating factors. In the group at UCPH, we are working on methods and models that are well suited for these types of data.

Due to the limited size of some of the data sets, we will work on representation learning combined with transfer learning. The idea of representation learning is to learn a *low dimensional* representation of the high-dimensional data, which can facilitate subsequent training of a classifier or a survival model. Sometimes, it is possible to include additional data, for instance from the public domain, when learning representations, which potentially can lead to very robust representations. This has a significant potential for RNA-sequencing data.

Noise, uncertainty and missing data are best handled by probabilistic models. The generative neural networks described in the introduction are very expressive machine learning models that output probabilities rather than point estimates. Using such models, we can therefore better quantify the uncertainty.

We intend to apply and further develop a generative model, which combine representation learning and generative neural network, and which is also particularly well suited for handling of missing data. This model and the results obtained are briefly described below.

9.2 The deep generative decoder

The deep generative decoder (DGD) [75] is a generative neural network, which consists of a probability distribution over latent space (or representation space) and a neural network (the decoder), which maps representations to the feature (output) space. As the model described in Chapter 6, the distribution over representation space is parametrized and can have any differentiable form, such as a mixture of Gaussians, in which the mixture coefficients, component means and covariances are the parameters. The whole model is estimated by maximum aposteriori (MAP) estimation, including the representations. Please consult our paper for more details: `https://arxiv.org/abs/2110.06672`

The model is very similar to the variational autoencoder (VAE) [64] mentioned earlier, but does not have an encoder. The main difference from the VAE is the MAP estimation instead of the variational inference used in the VAE.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

Our experience with the model so far is that it is much easier to train than a corresponding VAE and that the results are also significantly better. In our recent paper [75], we used the CIFAR10 dataset to evaluate the performance of the DGD at various tasks such as model convergence and accuracy, the ability to reconstruct the original image; which we evaluated in various ways. As a baseline, we compared our model to a VAE.

We trained both the VAE and the DGD for 50 epochs on the CIFAR10 data, and we found that the DGD model search was faster than the VAE (Figure 9.1A). Despite the fact that the VAE converges faster during the initial iterations (0 to 20 epochs), the DGD model achieves a lower reconstruction error once convergence is achieved after 50 epochs (Figure 9.1B), suggesting that the DGD is a more stable and easier to train model.

Deep generative models, such as Generative Adversarial Networks (GANs) have recently attracted a lot of attention as an state-of-the-art method to generate random images. We therefore decided to compare the image generation performance of the DGD against the VAE and the popular, GAN based architecture DCGAN [76]. In order to contrast the different models, we investigated their abilities at generating random images and reconstructing the test images using the CIFAR10 dataset (Figure 9.1C). We found that the VAE performed poorly, mainly generating blurry images. In contrast, the DGD model generated sharper images, resembling those images that were generated by the DCGAN. We must acknowledge that the low performance of the VAE does probably not represent the optimal state this model can achieve provided we are not experts in the field of VAEs. Yet, our simple approach with the DGD method provides a robust starting point to train these type of models, which can be very useful for researchers without long-time experience in the field.

The Bayesian approach provides a natural approach to encode prior believes in a mathematically rigorous manner. Using MNIST as a case example, it seems plausible to consider a prior that assigns different clusters to the different types of digits, therefore allowing the data to get away from 0. To incorporate this idea (see [75] for details), we used a ”soft-ball” prior over the means of the Gaussian mixture, a Dirichlet prior over the mixture coefficients and a Gaussian prior on $\log(1/\sigma^2)$. Finally, as an evaluation, we visually tested the ability of our model to represent the MNIST data in a 2D space, using 20, 30 and 50 mixtures in the representation layer (Figure 9.1D-F). We found that the DGD approach with priors formed well-separated clusters in 2D for the different numbers. Furthermore, random digits generated with the model were very good.

One of the advantages of our model is that due to the lack of encoder neural network, the model design is also easier and the number of model parameters much smaller. This comes at the price that one has to run a MAP optimization for new samples in order to find the most probable representation and there is a danger that this process ends up in a local maximum rather than the global. For very large networks or in situations where speed is an issue, it may therefore be advisable to later introduce an encoder. An encoder can be trained entirely from the decoder without access to the real samples as described in [75].

There are a number of advantages to this model, which we believe matches the GenoMed4ALL project very well. Below we will describe some of them and some of the extensions that will be developed in the project.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

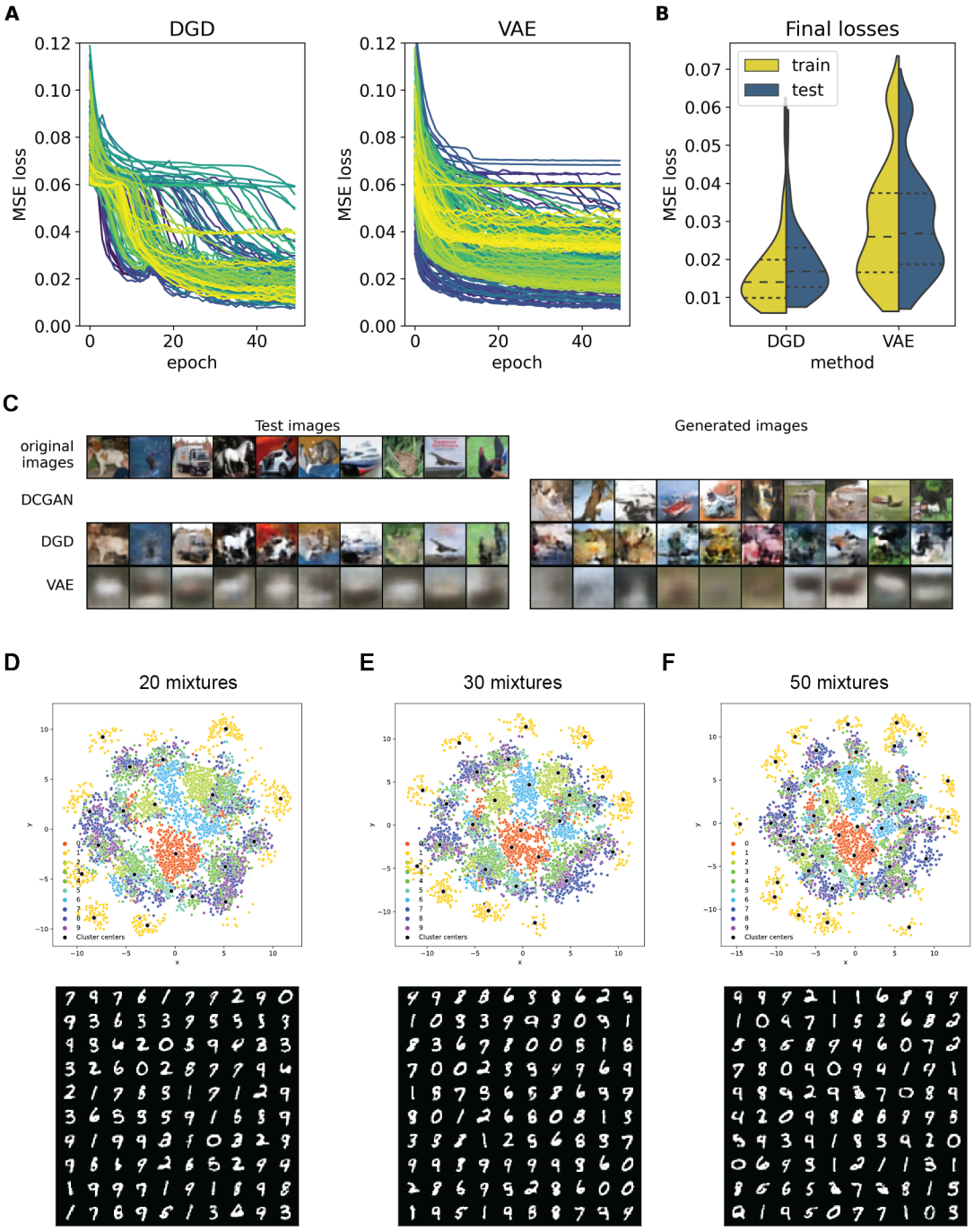


Figure 9.1: Performance of the Deep Generative Decoder and the VAE on CIFAR10 and MNIST.(**A**) Learning curves for the DGD and the VAE.(**B**) Test loss distributions for the VAE and the DGD. (**C**) Image generation and reconstruction performance of the DCGAN, DGD and VAE, tested on the CIFAR10 dataset. (**D-F**) 2D representation space and random samples from DGD models with priors. The models contain 20 mixtures (**D**), 30 mixtures (**E**) and 50 mixtures (**F**) on the representation layer. Figure modified with permission from [75]

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

9.3 Missing values and imputation

Many machine learning approaches that we would consider for medical data relies on a complete data vector. If some of the values are missing in a data vector, they are replaced by some guesses, such as the mean value across samples. In the GDG, however, there is no encoder, and we can calculate the probability of the model (corresponding to the likelihood) just from the values present. This property (first pointed out in [77]) means that one do not have to guess any values, but simply optimize the model from the values that are present.

Once the model is estimated, it can be used for imputing missing values in arbitrary samples. This is done by first finding the most probably representation for the sample and then compute the probability distribution for the missing value. From this probability distribution, one can sample one or more values or simply choose the most probable as the imputed value. One can also sample representations and thereby obtain an even better distribution.

We will implement model estimation with missing data as well as the imputation method and test it on data in the GenoMed4ALL project.

9.4 Representation learning

When estimating the model, we obtain the most probable representations of the training data at the same time as we train the decoder. This is very similar to what happens in manifold learning, such as UMAP [78], that also returns a set of ”optimal” representations. The two methods differ in their objective functions, however, where the manifold learning methods seek to maximize a concordance based on the distances between points in representation and output space, the generative model treats the points as independent and seeks to obtain a good distribution of data.

We will work on reconciling these two model types. This could be done by adding terms to the loss of the DGD, which aims at making “good” representations. We have put “good” in quotes, because part of this work is to find ways of measuring the quality of a representation. One measure of quality that we will pursue is the performance of other machine learning tasks based on the representations. It means that a representation is judged on the performance in e.g. a classification or survival analysis task.

We will develop the representation learning and compare to UMAP. We will use data from the GenoMed4ALL project, on which UMAP is already applied.

This project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

9.5 Integration in a federated implementation

There are some advantages of the DGD when it comes to a Federated implementation. One is that the representations of samples (as well as the samples of course) can be kept private to the nodes and only the decoder and other parameters need to be communicated between nodes. Another advantage over the VAE is that the model is smaller and contains approximately half the number of parameters.

In collaboration with other partners in GenoMed4ALL, we will implement the GDG in a federated framework and test the different ingredients of the model (missing values, imputation, representation) on the tests developed for the project.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

10 FORTH contribution

10.1 Gradient Tree Boosting

Ensemble learning is an interesting meta-learning strategy where a large number of relatively weak simple models are combined in order to obtain a stronger ensemble prediction [79]. The most prominent examples of such machine-learning ensemble techniques are random forests [80] that use the "Bagging" technique (sampling with replacement from the data set and averaging of the sub-models) and have gained a lot of popularity in recent years. Instead of the simple model averaging technique that Random forests use, the so called "boosting" technique and the AdaBoost algorithm in ensemble learning [81] follows an iterative, "stage-wise" additive approach where a new model is added and trained based on the errors of the whole ensemble in the current iteration. A statistical view of the boosting techniques led to the introduction of the gradient boosting machines [82, 83] and gradient tree boosting when decision trees are used as the "weak" learners. Under this statistical framework, the new base-learners are selected according to their correlation with the negative gradient of the loss function, associated with the whole ensemble. Here, we are going to quickly present XGBoost which is one of most famous examples of gradient tree boosting algorithms [84].

In the Gradient Boosting Decision Trees (GBDT) setting a sequence of decision trees are trained. Formally, given a loss function l and a data set with n instances and d features $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, where $|\mathcal{D}| = n$, $\mathbf{x}_i \in \mathbb{R}^d$, and $y_i \in \mathbb{R}$, GBDT minimizes the following objective function [84]:

$$\tilde{\mathcal{L}} = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega\left(f_k\right)$$

where $\Omega(f) = \gamma T_l + \frac{1}{2}\lambda\|w\|^2$ is a regularization term to penalize the complexity of the model. Here γ and λ are hyper-parameters, T_l is the number of leaves and w is the leaf weight. Each f_k corresponds to a decision tree. Training the model in an additive manner, GBDT minimizes the following objective function at the t-th iteration.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}\left[g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2}h_i f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right)$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ are first and second order gradient statistics on the loss function. The decision tree is built from the root until reaching the restrictions such as the maximum depth. If I_L and I_R are the instance sets of left and right nodes after the split and letting $I = I_L \cup I_R$, then the "gain" of the split is given by

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \tag{10.1}$$

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

10.2 XGBoost on a Federated Learning setting

As described previously, during the process of construction of a tree, XGBoost (and other Gradient Boost algorithms) deal with the problem of locating an optimal split for the residuals of the data set. In XGBoost splitting is applied by locating a threshold that maximizes (10.1), which simplifies to the following since γ is constant:

$$\mathfrak{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right]$$

Where I_L, I_R are the instances (i.e. samples) of the data on the left and the right nodes respectively after the split, and I are all the instances before the split. Also, g_i and h_i are the first and second order gradient statistics of the loss function: $l(\hat{y}_i, y_i)$. In this function $\hat{y}_i$ is the predicted probability of sample i and y_i is the target of sample i. Finally, λ is a regularization parameter ($\lambda > 0$), the greater the λ the more pruning is applied from the algorithm.

Now if we assume that the loss function is "log-loss", one of the most common functions for binary classification:

$$l(\hat{y}_i, y_i) = -\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

then the first and second order gradients of this loss faction are $g_i = \hat{y}_i - yi$, or else the residuals and $h_i = \hat{y}_i(1 - \hat{y}_i)$. If we substitute these values in $\mathfrak{L}_{split}$ then the optimal split is the one that maximizes the expression:

$$L_{\text{split}} = \text{SimilarityScore}_{\text{left}} + \text{SimilarityScore}_{\text{right}} - \text{SimilarityScore}_{\text{root}}$$

where the $SimilarityScore$ for a collection of samples is:

$$SimilarityScore = \frac{\left(\sum_{i=1}^{n} g_i\right)^2}{\sum_{i=1}^{n} h_i + \lambda}$$

or else:

$$SimilarityScore = \frac{\left(\sum_{i=1}^{n} residual_i\right)^2}{\sum_{i=1}^{n} \hat{y}_i * (1 - \hat{y}_i) + \lambda}$$

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

So here we can see that in order to locate the optimal split of a leaf in the tree, we just need to find(1) sums of the residuals (for g_i) and (ii) the sum of the likelihoods of the predictions (for h_i). XGBoost locates this optimal split through an exact Greedy algorithm. Ultimately this means that it just applies all possible splits. A large part of the optimizations that are described in the XGBoost algorithm are dealing with substituting the greedy algorithm with heuristics that limit the number of possible splits. Here we present a split finding algorithm that is suitable for federated learning.

An ideal solution would be for all federated nodes to transmit all g_i and h_i values for each split to the master node and then the master node locates the optimal split as suggested in [85]. This method has the disadvantage of revealing too much information regarding the data in each node. For example in the first iteration the central node would get the residuals (g_i) for every instance for every node. Assuming an initial prediction probability of 0.5, it could simply check if the sample belongs to the positive or negative class by just checking if the residual is positive or negative. Furthermore, this requires a lot of communication with the central orchestrating node that will negatively affect the training time.

Here we suggest the following approach. Each node computes the S different splits that yield the top S $\mathfrak{L}_{split}$ values, where S is a configuration hyperparameter. For example if $S = 10$ then each node computes the splits that produce the top 10 $\mathfrak{L}_{split}$ values. Then each node returns to the central node three types of data: (i) the splits, (ii) their corresponding values of $\mathfrak{L}_{split}$ and (iii) the number of samples that belong to the root node of this split for this node. Assume that SP_s is the s-th split, C is the number of federated nodes, $\mathfrak{L}_{s,p}$ is the $\mathfrak{L}_{split}$ value returned from the p-th node when it applies the s-th split. Also assume that N_p is the number of samples that the root node contains in the p-th node (this is before the split, so it is irrelevant to the split). After receiving this data the central node computes the split that maximized the following expression:

$$\underset{SP_s}{argmax} \left\{ \sum_{p=1}^{C} N_p \mathfrak{L}_{s,p} \right\}$$

Or else it locates the split that contains a high number of samples over many nodes (N_p) and also high values of $\mathfrak{L}_{split}$.

After obtaining the optimal split the central node transmits that to the nodes so that they can continue building the tree. Again some information about the distribution of the data can be deduced, but important information like class values is not leaked.

The second part of the XGBoost algorithm is the computation of the output values of the leaves of the tree, or else the weight w_j^* of leaf j. According to the XGBoost publication this value is equal to:

$$w_j^* = \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_j + \lambda}$$

In contrast to the optimal split, here there isn't any "search" that needs to take place. The computation of the output value happens after the optimal tree has been constructed. Given the same loss function l and

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

substituting g_i and h_i in the formula above we have:

$$w_j^* = \frac{\sum_{i=1}^{n} residual_i}{\sum_{i=1}^{n} \hat{y}_i(1-\hat{y}_i) + \lambda}$$

Here we notice that the output value of a leaf can be computed from the sum of the residuals that this leaf contains and also from the sum of the likelihoods of the predicted probabilities that this leaf contains. Therefore after a split, each node can just transmit these two sums to the central node. The central node adds all sums and can compute the w_j^* for all leaves which transmits back to the clients. It is important to note that during this federated process all nodes 'work' on constructing the same set of trees.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

11 Conclusions

This report has played a remarkable role in the evolution of the project, as all partners involved have gained a lot of additional experience in federated architectures. In this period we have organised recurring meetings to clarify the scope and limitations of various federated implementations applicable to different challenges posed by the case studies ranging from supervised to unsupervised problems to very promising ideas. At this stage, all partners involved have a clearer perspective on the approaches they want to explore. In addition, some important missing issues have been identified that need to be addressed in the future:

- We need a single code chain that includes common procedures for generating data and analysing results. In addition, an agreed format of the different contributions is needed to be able to compare alternatives on fair terms.
- We need closer interaction with the leaders of other WPs to list the set of challenging problems defined by those responsible for the different case studies to make sure that AI initiatives are covering all demands.
- We need to expand our cooperation between the different partners to create synergies and promote a productive fusion of ideas. So far, we have identified several approaches with some common fundamental architectures that can probably be improved through joint efforts.
- We have also identified the need to bring our AI analysis closer to clinicians, while also trying to develop tools that improve the interpretability and explainability of the results of our mostly black-box algorithms.
- We have realized that we need an accurate and objective test procedure that should validate whether the federated implementation is really effective and distinguish the situation where performance has been obtained only on isolated algorithms and local data. This procedure will define good nodes with complete data whose isolated behaviour will define an upper bound of performance and very poor nodes with only a few samples and many missing features whose isolated performance should be an unsatisfactory lower bound but which improves significantly after the federated process. This improvement will quantify the effective gain of our implementations.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

References

[1] H. McMahan, Eider Moore, D. Ramage, S. Hampson, and Blaise Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[2] P. Kairouz, H. McMahan, B. Avent, Aurélien Bellet, Mehdi Bennis, A. Bhagoji, Keith Bonawitz, Z. Charles, Graham Cormode, R. Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, D. Evans, Josh Gardner, Zachary A. Garrett, A. Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Z. Harchaoui, Chaoyang He, Lie He, Z. Huo, B. Hutchinson, Justin Hsu, M. Jaggi, T. Javidi, Gauri Joshi, M. Khodak, Jakub Konecný, A. Korolova, F. Koushanfar, O. Koyejo, T. Lepoint, Yang Liu, P. Mittal, M. Mohri, R. Nock, Ayfer Özgür, R. Pagh, Mariana Raykova, Hang Qi, D. Ramage, R. Raskar, D. Song, Weikang Song, S. Stich, Ziteng Sun, A. T. Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, L. Xiong, Zheng Xu, Q. Yang, F. Yu, Han Yu, and Sen Zhao, "Advances and open problems in federated learning," *ArXiv*, vol. abs/1912.04977, 2019.

[3] "State of ml frameworks," 2019 (accessed Jan 31, 2021), `https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning/`.

[4] "Fate framework," (accessed Feb 2, 2021), `https://fate.fedai.org/`.

[5] "Fate," `https://github.com/FederatedAI/FATE`, 2019.

[6] "Paddlefl," (accessed Feb 2, 2021), `https://paddlefl.readthedocs.io/en/latest/introduction.html`.

[7] "Paddelpaddel," `https://github.com/PaddlePaddle/Paddle`, 2018.

[8] T. Ryffel, Andrew Trask, M. Dahl, Bobby Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *ArXiv*, vol. abs/1811.04017, 2018.

[9] "Pysyft," `https://github.com/OpenMined/PySyft`, 2017.

[10] "Tensorflow federated," (accessed Feb 2, 2021), `https://www.tensorflow.org/federated`.

[11] "Tensorflow federated," `https://github.com/tensorflow/federated`, 2018.

[12] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Y. Zhou, Ali Anwar, Shashank Rajamoni, Yuya Jeremy Ong, J. Radhakrishnan, A. Verma, M. Sinn, M. Purcell, Ambrish Rawat, T. Minh, N. Holohan, S. Chakraborty, Shalisha Whitherspoon, Dean Steuer, L. Wynter, Hifaz Hassan, Sean Laguna, Mikhail Yurochkin, M. Agarwal, Ebube Chuba, and Annie Abay, "Ibm federated learning: an enterprise framework white paper v0.1," *ArXiv*, vol. abs/2007.10987, 2020.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

[13] Nathalie Baracaldo Heiko Ludwig and Gegi Thomas, "Ibm federated learning – machine learning where the data is," August 21, 2020 (accessed February 3, 2021), `https://www.ibm.com/blogs/research/2020/08/ibm-federated-learning-machine-learning-where-the-data-is/`.

[14] Several contributors, "Ibm federated learning," 2020 (accessed February 3, 2021), `https://github.com/IBM/federated-learning-lib`.

[15] Khuyen Tran, "Introduction to ibm federated learning: A collaborative approach to train ml models on private data," July 20, 2020 (accessed February 3, 2021), `https://towardsdatascience.com/introduction-to-ibm-federated-learning-a-collaborative-approach-to-train-ml-models-on-pr`

[16] "Nvidia clara," (accessed Feb 11, 2021), `https://developer.nvidia.com/clara`.

[17] "Leaf," 2019 (accessed Feb 2, 2021), `https://leaf.cmu.edu/`.

[18] S. Caldas, Peter Wu, Tian Li, Jakub Konecný, H. McMahan, V. Smith, and Ameet Talwalkar, "Leaf: A benchmark for federated settings," *ArXiv*, vol. abs/1812.01097, 2018.

[19] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, pp. 50–60, 2020.

[20] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, M. Zaheer, Ameet Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *ArXiv*, vol. abs/1812.06127, 2018.

[21] Li Huang, Yifeng Yin, Z. Fu, S. Zhang, Hao Deng, and D. Liu, "Loadaboost: Loss-based adaboost federated machine learning on medical data," *ArXiv*, vol. abs/1811.12629, 2018.

[22] Nicholas Carlini, Chang Liu, J. Kos, Úlfar Erlingsson, and D. Song, "The secret sharer: Measuring unintended neural network memorization & extracting secrets," *ArXiv*, vol. abs/1802.08232, 2018.

[23] J. Nielsen, "Extending oblivious transfers efficiently - how to get robustness almost for free," *IACR Cryptol. ePrint Arch.*, vol. 2007, pp. 215, 2007.

[24] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC '09*, 2009.

[25] Zvika Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," *IACR Cryptol. ePrint Arch.*, vol. 2012, pp. 78, 2012.

[26] "Helib," `https://github.com/homenc/HElib`, 2014.

[27] Sai Sri Sathya, Praneeth Vepakomma, R. Raskar, R. Ramachandra, and Santanu Bhattacharya, "A review of homomorphic encryption libraries for secure computation," *ArXiv*, vol. abs/1812.02428, 2018.

[28] C. Dwork, "Differential privacy: A survey of results," in *TAMC*, 2008.

[29] L. Babai, L. Fortnow, L. Levin, and M. Szegedy, "Checking computations in polylogarithmic time," in *STOC '91*, 1991.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

[30] A. Francillon, Q. Nguyen, Kasper Bonne Rasmussen, and G. Tsudik, "A minimalist approach to remote attestation," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2014.

[31] Wei Dai, Abhimanu Kumar, Jinliang Wei, Q. Ho, Garth A. Gibson, and E. Xing, "High-performance distributed ml at scale through parameter server consistency models," *ArXiv*, vol. abs/1410.8043, 2015.

[32] Takayuki Nishio and Ryo Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2019.

[33] Zachary B. Charles and Dimitris Papailiopoulos, "Gradient coding via the stochastic block model," *ArXiv*, vol. abs/1805.10378, 2018.

[34] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, M. Zaheer, Ameet Talwalkar, and Virginia Smith, "Federated optimization in heterogeneous networks," *arXiv: Learning*, 2020.

[35] S. Caldas, Jakub Konecný, H. McMahan, and Ameet Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *ArXiv*, vol. abs/1812.07210, 2018.

[36] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, pp. 3400–3413, 2020.

[37] Xiangru Lian, Ce Zhang, Huan Zhang, C. Hsieh, Wei Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *NIPS*, 2017.

[38] Lie He, An Bian, and M. Jaggi, "Cola: Decentralized linear learning," in *NeurIPS*, 2018.

[39] Lumin Liu, J. Zhang, S. Song, and K. Letaief, "Edge-assisted hierarchical federated learning with non-iid data," *ArXiv*, vol. abs/1905.06641, 2019.

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

[41] Stephen P. Boyd, N. Parikh, E. Chu, B. Peleato, and Jonathan Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, pp. 1–122, 2011.

[42] Alex Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. 2011, vol. 24, Curran Associates, Inc.

[43] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra, "Weight uncertainty in neural networks," 2015.

[44] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

[45] Thang D. Bui, Cuong V. Nguyen, Siddharth Swaroop, and Richard E. Turner, "Partitioned variational inference: A unified framework encompassing federated and continual learning," 2018.

[46] Aki Vehtari, Andrew Gelman, Tuomas Sivula, Pasi Jylänki, Dustin Tran, Swupnil Sahai, Paul Blomstedt, John P. Cunningham, David Schiminovich, and Christian Robert, "Expectation propagation as a way of life: A framework for bayesian inference on partitioned data," 2019.

[47] Leonard Hasenclever, Stefan Webb, Thibaut Lienart, Sebastian Vollmer, Balaji Lakshminarayanan, Charles Blundell, and Yee Whye Teh, "Distributed bayesian learning with stochastic natural-gradient expectation propagation and the posterior server," 2017.

[48] Luca Corinzia, Ami Beuret, and Joachim M. Buhmann, "Variational federated multi-task learning," 2021.

[49] Jinsung Yoon, James Jordon, and Mihaela van der Schaar, "Radialgan: Leveraging multiple datasets to improve target-specific predictive models using generative adversarial networks," 2018.

[50] Jenna Wiens, John V. Guttag, and Eric Horvitz, "A study in transfer learning: leveraging data from multiple hospitals to enhance hospital-specific predictions," *Journal of the American Medical Informatics Association : JAMIA*, vol. 21 4, pp. 699–706, 2014.

[51] Chunyuan Li, Hao Liu, Changyou Chen, Yunchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin, "Alice: Towards understanding adversarial learning for joint distribution matching," 2017.

[52] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2020.

[53] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim, "Learning to discover cross-domain relations with generative adversarial networks," 2017.

[54] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," 2018.

[55] Jakub M. Tomczak and Max Welling, "Vae with a vampprior," 2018.

[56] Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi, "Variational autoencoder with implicit optimal priors," 2019.

[57] Chunsheng Guo, Jialuo Zhou, Huahua Chen, Na Ying, Jianwu Zhang, and Di Zhou, "Variational autoencoder with optimizing gaussian mixture model priors," *IEEE Access*, vol. 8, pp. 43992–44005, 2020.

[58] Vignesh Prasad, Dipanjan Das, and Brojeshwar Bhowmick, "Variational clustering: Leveraging variational autoencoders for image clustering," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–10.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

[59] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *arXiv preprint arXiv:1611.02648*, 2016.

[60] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," 2017.

[61] Alfredo Nazabal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera, "Handling incomplete heterogeneous data using vaes," 2020.

[62] Diederik P Kingma, Danilo J Rezende, Shakir Mohamed, and Max Welling, "Semi-supervised learning with deep generative models," *arXiv preprint arXiv:1406.5298*, 2014.

[63] Rajesh Ranganath, Adler Perotte, Noémie Elhadad, and David Blei, "Deep survival analysis," in *Machine Learning for Healthcare Conference*. PMLR, 2016, pp. 101–114.

[64] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv*, vol. 1312.6114 [stat.ML], Dec. 2013.

[65] Elli Papaemmanuil, Moritz Gerstung, Lars Bullinger, Verena I Gaidzik, Peter Paschka, Nicola D Roberts, Nicola E Potter, Michael Heuser, Felicitas Thol, Niccolo Bolli, et al., "Genomic classification and prognosis in acute myeloid leukemia," *New England Journal of Medicine*, vol. 374, no. 23, pp. 2209–2221, 2016.

[66] Matteo Bersanelli, Erica Travaglino, Manja Meggendorfer, Tommaso Matteuzzi, Claudia Sala, Ettore Mosca, Chiara Chiereghin, Noemi Di Nanni, Matteo Gnocchi, Matteo Zampini, et al., "Classification and personalized prognostic assessment on the basis of clinical and genomic features in myelodysplastic syndromes," *J Clin Oncol.*, 2021.

[67] Taro Matsutani and Michiaki Hamada, "Parallelized latent dirichlet allocation provides a novel interpretability of mutation signatures in cancer genomes," *Genes*, vol. 11, no. 10, 2020.

[68] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, Sept. 1999.

[69] Junyuan Xie, Ross B. Girshick, and Ali Farhadi, "Unsupervised deep embedding for clustering analysis," *CoRR*, vol. abs/1511.06335, 2015.

[70] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro P. B. de Gusmão, and Nicholas D. Lane, "Flower: A friendly federated learning research framework," 2021.

[71] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2017.

[72] David Faraggi and Richard Simon, "A neural network model for survival data," *Statistics in medicine*, vol. 14, no. 1, pp. 73–82, 1995.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

[73] 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.

[74] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin, "Federated learning on non-iid data: A survey," *arXiv preprint arXiv:2106.06843*, 2021.

[75] Viktoria Schuster and Anders Krogh, "The deep generative decoder: Using MAP estimates of representations," 2021.

[76] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[77] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu, "Alternating back-propagation for generator network," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, Satinder P. Singh and Shaul Markovitch, Eds. 2017, pp. 1976–1984, AAAI Press.

[78] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger, "UMAP: Uniform manifold approximation and projection," *The Journal of Open Source Software*, vol. 3, no. 29, pp. 861, 2018.

[79] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[80] Leo Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[81] Robert E Schapire, "The boosting approach to machine learning: An overview," *Nonlinear estimation and classification*, pp. 149–171, 2003.

[82] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[83] Jerome H Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[84] Tianqi Chen and Carlos Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco California USA, Aug. 2016, pp. 785–794, ACM.

[85] Mengwei Yang, Linqi Song, Jie Xu, Congduan Li, and Guozhen Tan, "The Tradeoff Between Privacy and Accuracy in Anomaly Detection Using Federated XGBoost," *arXiv:1907.07157 [cs, stat]*, Oct. 2019, arXiv: 1907.07157.

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **101017549**

GENOMED4ALL

genomed4all.eu

@genomed4all

/genomed4all

GENOMED4ALL receives funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. 101017549