Privacy-preserving statistical and machine learning methods under fully homomorphic encryption



Pedro M. Esperança

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2016

Abstract

Doctor of Philosophy

University of Oxford, Trinity 2016

Privacy-preserving statistical and machine learning methods under fully homomorphic encryption

by Pedro M. Esperança, Exeter College

Advances in technology have now made it possible to monitor heart rate, body temperature and sleep patterns; continuously track movement; record brain activity; and sequence DNA in the jungle — all using devices that fit in the palm of a hand. These and other recent developments have sparked interest in privacy-preserving methods: computational approaches which are able to utilise the data without leaking subjects' personal information.

Classical encryption techniques have been used very successfully to protect data in transit and in storage. However, the process of encrypting data also renders it unusable in computation. Recently developed fully homomorphic encryption (FHE) techniques improve on this substantially. Unlike classical methods, which require the data to be decrypted prior to computation, homomorphic methods allow data to be simultaneously stored or transfered securely, and used in computation. However, FHE imposes serious constraints on computation, both arithmetic (e.g., no divisions can be performed) and computational (e.g., multiplications become much slower), rendering traditional statistical algorithms inadequate.

In this thesis we develop statistical and machine learning methods for outsourced, privacy-preserving analysis of sensitive information under FHE. Specifically, we tackle two problems: (i) classification, using a semiparametric approach based on the naive Bayes assumption and modeling the class decision boundary directly using an approximation to univariate logistic regression; (ii) regression, using two approaches; an accelerated method for least squares estimation based on gradient descent, and a cooperative framework for Bayesian regression based on recursive Bayesian updating in a multi-party setting. Taking into account the constraints imposed by FHE, we analyse the potential of different algorithmic approaches to provide tractable solutions to these problems and give details on several computational costs and performance trade-offs.

Acknowledgements

I thank my supervisors, Chris Holmes and Michael Bonsall, for all the guidance and support throughout this research and during my DTC projects; for sharing their vast knowledge; and for their excitement about this research.

I thank Louis Aslett, collaborator and invaluable guide in all matters, for being kind and patient to share his immense expertise in virtually all subjects relevant to this work.

I thank a number of people for additional and important comments on early versions of this work: Arnaud Doucet, Gil McVean, Geoff Nicholls, Christina Palmou.

I also thank the Life Sciences Interface Doctoral Training Centre (DTC:LSI) and the Engineering and Physical Sciences Research Council (EPSRC) for funding. EPSRC grant reference number EP/F500394/1.

Contents

Abstract i

Acknowledgements ii

Contents ii

Main Contributions vii

Abbreviations viii

Notation ix

1 Introduction 1
- 1.1 What is data privacy? . . . 2
- 1.2 Why is data privacy important? . . . 3
- 1.3 How can we balance data privacy and data usability? . . . 6
- 1.4 The challenge: privacy-preserving statistical analysis . . . 7
- 1.5 Thesis outline . . . 9

2 Background on fully homomorphic encryption 10
- 2.1 Data privacy . . . 11
- 2.2 Classical encryption . . . 13
  - 2.2.1 Basic notions . . . 13
  - 2.2.2 Examples: Caesar cipher and RSA . . . 15
  - 2.2.3 Security: key sizes, noise and hard problems . . . 17
  - 2.2.4 Limitations . . . 20
- 2.3 Homomorphic encryption . . . 20
  - 2.3.1 Basic notions . . . 21
  - 2.3.2 The homomorphic principle . . . 22
  - 2.3.3 Bounding ciphertext noise . . . 24
  - 2.3.4 Theoretical possibilities and limitations . . . 25
  - 2.3.5 Application framework . . . 27
- 2.4 The `FV` homomorphic cryptosystem . . . 28
  - 2.4.1 Description . . . 29
  - 2.4.2 Security: the 'learning with errors' problem . . . 33
  - 2.4.3 Constraints . . . 34
  - 2.4.4 Computational aspects . . . 37
- 2.5 Alternative notions of privacy . . . 41

2.5.1 Anonymisation . . . 42
2.5.2 Differential privacy . . . 43
2.5.3 Secure multi-party computation . . . 45
2.6 Summary . . . 46

3 Semiparametric naive Bayes classification 48
3.1 Data representation and encoding . . . 49
3.1.1 Quantisation . . . 50
3.1.2 Expansion . . . 51
3.2 The naive Bayes framework . . . 52
3.3 Multinomial naive Bayes classifier . . . 54
3.3.1 Unnormalised probabilities . . . 54
3.3.2 Encrypted conditional counts . . . 54
3.3.3 Prediction . . . 55
3.3.4 Theoretical parameter requirements for the `FV` cryptosystem . . . 56
3.4 Semiparametric naive Bayes classifier . . . 56
3.4.1 Approximate logistic regression . . . 58
3.4.1.1 Iteratively reweighted least squares . . . 58
3.4.1.2 Paired optimisation . . . 59
3.4.1.3 Unpaired optimisation . . . 60
3.4.2 Shrinkage . . . 61
3.4.3 Generalisation error . . . 62
3.4.4 Theoretical parameter requirements for the `FV` cryptosystem . . . 64
3.5 Results . . . 64
3.5.1 Classification performance . . . 64
3.5.2 Computational costs . . . 66
3.6 Discussion . . . 67
3.7 Summary . . . 69

4 Accelerated least squares regression 71
4.1 Data representation and encoding . . . 72
4.2 The linear regression model . . . 72
4.2.1 Ordinary least squares . . . 73
4.2.2 Regularised least squares . . . 73
4.3 Least squares via descent methods . . . 75
4.3.1 Iterative methods . . . 76
4.3.1.1 Sequential updates via coordinate descent . . . 77
4.3.1.2 Simultaneous updates via gradient descent . . . 78
4.3.2 Prediction . . . 81
4.3.3 Inference . . . 82
4.3.4 Regularisation . . . 83
4.3.5 Theoretical parameter requirements for the `FV` cryptosystem . . . 83
4.4 Acceleration . . . 85
4.4.1 Preconditioning . . . 85
4.4.2 van Wijngaarden transformation . . . 86
4.4.3 Nesterov's accelerated gradient . . . 88
4.5 Results . . . 89

4.5.1 Simulations . . . 89
4.5.2 Applications . . . 93
4.5.2.1 Mood stability data . . . 93
4.5.2.2 Prostate cancer data . . . 94
4.6 Discussion . . . 98
4.7 Summary . . . 100

5 Cooperative Bayesian regression with distributed data 102
5.1 Data representation and encoding . . . 103
5.2 The Bayesian linear regression model . . . 104
5.3 Computational framework . . . 105
5.3.1 Distributed and horizontally partitioned data . . . 106
5.3.2 Multiple parties . . . 106
5.3.3 Recursive Bayesian updating . . . 106
5.4 Approximate recursive Bayesian updating . . . 108
5.4.1 Estimating the posterior variance . . . 108
5.4.2 Estimating the posterior mean . . . 109
5.4.3 Cooperative Bayesian linear regression . . . 111
5.4.4 Data standardisation . . . 112
5.4.5 Theoretical parameter requirements for the `FV` cryptosystem . . . 115
5.5 Results . . . 115
5.5.1 Performance . . . 115
5.5.2 Computational costs . . . 120
5.6 Discussion . . . 121
5.7 Summary . . . 123

6 Conclusion 125
6.1 Summary . . . 125
6.2 Developments . . . 128
6.3 Future work . . . 130

A Software and implementation 132
A.1 The `HomomorphicEncryption` package . . . 132
A.1.1 Initialisation and basic cryptographic functions . . . 132
A.1.1.1 Parameter selection . . . 132
A.1.1.2 Key generation . . . 135
A.1.1.3 Encryption . . . 135
A.1.1.4 Decryption . . . 136
A.1.2 Homomorphic operations . . . 136
A.2 The `EncryptedStats` package . . . 137
A.2.1 Semiparametric naive Bayes classifier . . . 137
A.2.1.1 Estimation . . . 137
A.2.1.2 Prediction . . . 139
A.3 Implementation details . . . 140
A.3.1 Hardware . . . 140
A.3.2 Software . . . 140

B Results **141**
B.1 Proof of Theorem 4.1 141
B.2 Transformed equations: `ELS-GD` 143
B.3 Cryptographic parameters: `ELS-GD` 144
B.4 Transformed equations: `ELS-NAG` 147
B.5 Transformed equations: `ECBLR-QN` 148
B.6 Cryptographic parameters: `ECBLR-QN` 150

Bibliography **153**

Glossary **160**

Main Contributions

- We review the broad area of privacy-preserving computation and, in particular, the field of fully homomorphic encryption, in a manner accessible to statisticians and applied scientists. We give details on the mechanics, possibilities and limitations of fully homomorphic encryption as relevant for the implementation and development of statistical and machine learning methods. We provide enough technical detail to make the workings and limitations of fully homomorphic encryption understandable from a theoretical perspective but abstain from presenting the technical aspects which are not essential for this understanding, thus striking a good balance between technicality and readability, aimed at non-cryptographers — *[Chapters 1–2]*

- We develop a set of tailored statical and machine learning methods for homomorphic computation, including classification and regression; provide details on the computational costs and limiting factors in these methods; study their computational trade-offs; and give practical advice on how to choose the cryptographic parameters required to initialise the encryption scheme used in order to achieve the desirable amount of computation under strong security and correctness guarantees — *[Chapters 3–5]*

- We provide an open-source software package, `EncryptedStats`, written for `R` and implementing the methods for encrypted statistical analysis developed in this thesis. The package follows the user-friendly philosophy of `R` and includes documentation and example code, which makes a good resources for practitioners in the field of homomorphic computation — *[Appendix A]*

Abbreviations

DP	**D**ifferential **P**rivacy
FHE	**F**ully **H**omomorphic **E**ncryption
FV	**F**an and **V**ercauteren (cryptosystem)
HE	**H**omomorphic **E**ncryption
MCS	**M**aximum **C**oefficient **S**ize
MMD	**M**aximum **M**ultiplicative **D**epth
MPC	**M**ulti-**P**arty **C**omputation
PHE	**P**artially **H**omomorphic **E**ncryption
SHE	**S**omewhat **H**omomorphic **E**ncryption
SMPC	**S**ecure **M**ulti-**P**arty **C**omputation

Notation

General:

$\mathbb{N}_{a:b}$	subset of the natural number from a to b, with $a, b \in \mathbb{N}$ and $a \leq b$, that is, $\mathbb{N}_{a:b} = \{a, a+1, \ldots, b-1, b\}$
$\|\| \cdot \|\|_p$	L_p norm

Data objects:

x	scalar
$\boldsymbol{x}$	vector
$\boldsymbol{X}$	matrix
$\boldsymbol{X}_{i\bullet}$	row i of $\boldsymbol{X}$
$\boldsymbol{X}_{\bullet j}$	column j of $\boldsymbol{X}$
$\mathbf{0}_P$	zero vector of length P
$\boldsymbol{I}_{PP}$	identity matrix of size $P \times P$

Cryptography:

`pk`	public key
`sk`	secret key
$\texttt{Enc}_{\texttt{pk}}(\cdot)$	encryption function
$\texttt{Dec}_{\texttt{sk}}(\cdot)$	decryption function
`L`	maximum multiplicative depth (MMD)
`max`	maximum coefficient size (MCS)

Chapter 1

Introduction

Fast-growing computational paradigms such as cloud computing and e-health — feeding on the recent increases in data collection capabilities brought about by smartphones and wearable devices — call for sophisticated privacy-preserving approaches that can balance two conflicting demands: data privacy and data usability. Data collection from these and other sources can be beneficial to understand, among others, rare clinical phenomena, assist in clinical decisions, and materialise the potential of personalised medicine (Sahoo et al., 2014).

Consider an illustrative scenario: data collected from different devices is uploaded to the cloud (*server*), which can monitor health indicators by computing statistical functions of these measurements, and then provide feedback to the patient (*client*) such as health status, advised medication dosage or whether to seek professional medical attention (Basilakis et al., 2015; Lauter et al., 2011). To give a concrete example: Saeb et al. (2015) show that daily-life behavioral markers extracted from mobile phone GPS can be used in identifying depressive symptom severity. In this scenario — and many others with shared characteristics — the services of data aggregation, storage, management and computation can be outsourced to the cloud. Unfortunately, privacy issues remain one of the biggest concerns and barriers to cloud use (Lagendijk et al., 2013). Privacy-preserving methods can be particularly valuable here, to eliminate privacy concerns and simultaneously allow data to be used in beneficial ways.

Given a good understanding of the issues surrounding data privacy and the cryptographic tools available to deal with these issues, we think that statisticians and other computational

scientists are very well positioned to make significant contributions to the field of secure computation, particularly in the subfield of secure statistical and machine learning analysis.

Roadmap. We start by introducing basic ideas surrounding data privacy (§1.1); then, we explain why privacy is important and explain the consequences of privacy breaches (§1.2); we then turn to potential solutions that guarantee data privacy and usability simultaneously, focusing on fully homomorphic encryption (§1.3); finally, we review the application areas of homomorphic encryption, set out our goals (§1.4), and give an outline of this thesis (§1.5).

1.1 What is data privacy?

In general, idiosyncrasies such as context-dependence make the task of defining privacy non-trivial. Nonetheless, it is possible to identify key dimensions that are fundamental to the concept of privacy and which can be used as guidance to assess security risks.

Let us start with some terminology: we denote by *server* an agent (individual or organisation) that collects, stores and uses data (e.g., a cloud) from a client, and who is typically not the subject of the data. We denote by *client* an agent that provides data (e.g., a patient or research institution) to a server, and who is typically the subject of the data, or acts on its behalf. Lastly we denote by *party* an agent that plays a role at some stage in the interaction process between servers and clients, typically one of the two, but possibly a third-party.

Exploring the concept of privacy, Barker et al. (2009) suggest a taxonomy with 4 key dimensions: "purpose": what is the motivation for the client to provide data to the server; "visibility": who is permitted to access or use the data; "granularity": how much detail does each party have access to; "retention": when must the data be destroyed. Along the same lines, Agrawal et al. (2002) define 10 database design principles to safeguard data privacy based on the Hippocratic Oath and comparable legislation, including "purpose specification", "consent", "limited use" and, naturally, "safety" (protection against theft and other misappropriations).

All these elements contribute to our perceived idea of privacy. They relate to data storage, access, management, usage, or a combination of these. In this thesis we focus on privacy during computation; a working definition will be presented in chapter 2.

1.2 Why is data privacy important?

Privacy and security issues arise in a vast number of scenarios with varying degree of severity depending on factors such as the nature of the data and the control that the data subjects have over it. Because infringements of privacy may carry catastrophic consequences, the topic has reached mainstream status. For instance, on 6 September 2016, the Science network aired an episode of the popular documentary series *Through the Wormhole* hosted by Morgan Freeman entitled "Is Privacy Dead?". The simplest reason why privacy is sought after is people's desires for personal privacy. This is a natural reaction in the face of the consequences of potential misuse of private information from an individual risk minimisation perspective. In addition, institutions holding patient data are often legally and ethically bound to safeguard patients' interests and rights in their use and storage of medical data, in which case privacy is in fact an obligation for data holders (Anderlik and Rothstein, 2001; Hazin et al., 2013).

Concerns. Most study participants show concerns about privacy, although perceptions of the risks of privacy violations are mixed. Confidentiality is valued as a safeguard against possible future discrimination, and although most people trust academic investigators with their personal information, a smaller percentage are comfortable with providing the same information to industry or government institutions (Jamal et al., 2014; Kaufman et al., 2009; Rogith et al., 2014; Sanderson et al., 2015). Also, while some people have doubts about the effectiveness of anonymisation — the practice of removing from databases information that could be used to uniquely identify subjects, such as the social security number — others seem less concerned (Audrey et al., 2016; Rogith et al., 2014). Importantly, consent seems to be understood as opt-in consent — where participants' approval is required before their data can be used for specific studies — and this raises issues of potential mismatches between expectations and reality (Audrey et al., 2016).

Risks. Privacy is particularly important, and has received a proportionate attention, in medical related areas of research such as genomics, personalised medicine, biobanks and biometric identification (Green et al., 2011; Shi and Wu, 2016; Waggett, 2016). Consider some of the reasons. First, the nature of biomedical data is especially sensitive; for instance, electronic health records contain highly personal information — such as social security number, disease

history and medical test results — the disclosure of which would constitute a violation of personal privacy (Hayden, 2013; Jensen et al., 2012). Second, knowledge about this type of sensitive information can be used by malicious agents with devastating consequences for the data subjects, such as stigmatisation and genetic-based discrimination (e.g., for insurance or employment). The problem is exacerbated in the case of genetic information, where it can be used not only against an individual but against his entire extended family (Enserink and Chin, 2015; Naveed et al., 2015). Third, data may hold commercial value for businesses or financial companies, for instance, strategic information or economic forecasts. As an illustrative example, consider the recently developed authentication systems based on brainwave scans, more precisely electroencephalograms (EEGs). The methods promise to outperform voice and face recognition based systems as well as the traditional use of fingerprints, which are susceptible to theft. There now exist wireless earphones that can also record your EEG and upload the data to your phone (Strickland, 2016). Despite of this spectacular achievement, EEGs hide a array of sensitive information that should be safeguarded from inappropriate use: analysing two EEG authentication systems, Matovu and Serwadda (2016) were able to identify 25% of the alcoholics in a dataset of EEG scans from a group of alcoholics and non-alcoholics — and that is potentially just 'the tip of the iceberg' of what EEGs can reveal about a person.

Breaches. Fears of privacy breaches, or data leaks, are not unfounded. Their history is long and we give here only some recent examples. The most common technique for releasing sensitive information is anonymisation. Unfortunately, it has become clear that removing only the type of information deemed 'personal' in this framework is insufficient to protect subjects' identity. On 2 October 2006, the online movie rental company Netflix launched the *Netflix Prize*. A 1 million dollar prize was to be awarded for the best algorithm to predict user ratings for films, and an anonymised training dataset with ratings from half a million users was publicly released. Two weeks later, Narayanan and Shmatikov (2006) showed that it was possible to re-identify user's records by matching them to movie ratings from the Internet Movie Database, an open movie ratings website. In this case the information leaked may be deemed unimportant, but it should be noted that with only basic knowledge of a subjects' preferences it is possible to determine their presence in the database and, therefore to identify their entire record and potentially uncover other personal, sensitive information such as political and religious views. Sweeney et al. (2013) were able to identify by name over 84%

of patients that had consented to participate in the Personal Genome Project (PGP) by cross-referencing metadata from that study (e.g., date of birth, gender, zip-code; all potentially valuable for research) with publicly available databases such as voter registration archives. This allowed them to access not just DNA data but also other types of highly sensitive data present in the PGP database, such as information on abortions, sexual abuse and illegal drug use. In a similar fashion, Sweeney (2013) was able to identify by name 43% of the patients in the Washington state health records by simply looking up information in 'hospitalisation' newspaper stories, which often contain the patient's name, residential neighbourhood and cause of hospitalisation. In another example, Gymrek et al. (2013) were able to recover surnames by profiling short tandem repeats in the Y-chromosome from personal genome datasets and cross-referencing this information with free, publicly accessible, recreational genetic genealogy databases. Many other examples have been published of compromised privacy as a result of simply analysing transaction records, social networks or telephone metadata (de Montjoye et al., 2013, 2015; Mayer et al., 2016; Narayanan and Shmatikov, 2008, 2009). In all these examples, anonymisation failed and private information was leaked.

Cloud computing and data sharing. Anonymisation techniques are well motivated at heart but lack privacy guarantees. Similar issues to those plaguing anonymised data release are faced in cloud computing and other data sharing projects. Cloud computing is a way to outsource storage or computation to a remote computer. The client uploads data to the server that then stores and computes with this data. Data synchronisation, wearable devices, mobile healthcare services and smartphones, all have increased the demand for cloud storage and computing services (Nepal et al., 2015; Tari et al., 2015). Clouds are attractive options for a number of reasons: they provide high storage and computing capacity (e.g., for large datasets), are efficient in their use of resources (minimisation of idle resources) and are economically rational (minimisation of maintenance costs). However, questions regarding data security arise: if data is being transfered to a potentially untrusted cloud server, how can we avoid privacy breaches? At the same time, the trade-off privacy versus usability of data emerges: if data is to be used in computation, how can we guarantee usability and security simultaneously? Data sharing faces similar issues; whether in public biobanks or private databases, privacy concerns often preclude data from being pooled from different

sources. With initiatives pushing for data sharing among scientists, a computational framework that satisfies both needs is in high demand (Beck, 2015; Bobrow, 2015; Greenbaum et al., 2011; Topol, 2015). Personalised medicine, with the help of wearable and electronic health monitoring devices, is another booming area of medical research where privacy guarantees are paramount. For instance, patients could upload medically relevant information and receive personalised feedback advising them on drug dosage (Austen, 2015; Basilakis et al., 2015; Bonsall et al., 2012; Holmes et al., 2016; Lauter et al., 2011; Paul and Irvine, 2014). Achieving this in a privacy-preserving manner would be a remarkable accomplishment and, we speculate, would make patients more willing to donate their data to research biobanks.

1.3 How can we balance data privacy and data usability?

The natural question that follows is then: *can we take steps to prevent exposure of sensitive information?* The answer is a sound *yes*. Cryptography is currently the gold-standard for protecting private information. Other alternative notions of privacy exist but they are either unsatisfactory, as in the case of anonymisation; or their precise potential is still uncertain, as in the case of differential privacy (we discuss alternative notions of privacy in §2.5).

Classical encryption. The classical approach to protect sensitive information x is to encrypt it, that is, transform it into some complex object $x' = \texttt{Enc}_{\texttt{k}}(x)$ which is indecipherable to untrusted parties. A key $\texttt{k}$ is used to encrypt and decrypt the data so that only the key holder has access to the original message $x = \texttt{Dec}_{\texttt{k}}(x')$. The specific form of x' depends on the encryption scheme used, but they are typically large objects such as 1024-bit integers (Kessler, 2016). Classical encryption methods enjoy strong security guarantees and are widely used to protect data in transit (e.g., online banking) and in storage (e.g., health records). The practical limitation of classical encryption methods is that encrypted data is rendered unusable by the same mechanism that ensures its privacy. If the data is to be used in any computation it must first be decrypted, thus compromising its confidentiality. Consequently, in the classical encryption framework we face the dilemma of choosing between data privacy and data usability: "having the cake and eating it too" is an impossibility.

Homomorphic encryption. Recent developments in cryptography are changing this fact, making it possible to have the *full* cake and eat *some* of it too. Homomorphic encryption shares many of the features of classical cryptography but goes a step further in allowing computation directly with encrypted data (Gentry, 2010). For instance, let $x' \leftarrow \texttt{Enc}_{\texttt{k}}(x)$ and $y' \leftarrow \texttt{Enc}_{\texttt{k}}(y)$ represent encryptions of x and y, respectively. An additively homomorphic encryption scheme has the following property: $\texttt{Dec}_{\texttt{k}}(x' \oplus y') = x + y$. This means that it is possible to add two encrypted values $(x' \oplus y')$ with the guarantee that the decrypted result will match that of the addition of the corresponding unencrypted values $(x + y)$. Put more dramatically, it means that we can *compute without seeing the data* — i.e., seeing only an encrypted version, which by definition leaks no information. The objects x' and y' are typically complex and high-dimensional (e.g., matrices or polynomials), and so the operation $\oplus$ which adds encrypted values is similarly more complex than regular addition. A fully homomorphic encryption (FHE) scheme is one which allows both addition $(\oplus)$ and multiplication $(\otimes)$ of encrypted values and provides the same correctness guarantees as above. Even though this may not seem impressive — after all, most statistics of interest require other algebraic operations besides addition and multiplication — given only these two basic operations and a binary (base-2) representation of the data, it is theoretically possible to perform *any* computation.

1.4 The challenge: privacy-preserving statistical analysis

FHE Restrictions. The beauty of FHE is that it allows data to be *simultaneously* stored securely and used in computation. Its possibilities for secure computation are immense, but achieving both data privacy and data usability is not a free lunch. To accommodate both of these demands, homomorphically encrypted data is considerably inflated in size, which leads to high computational costs and memory requirements relative to traditional (non-homomorphic) computation. Additionally, FHE schemes that work in binary spaces are prohibitively inefficient. Schemes that work directly with integers are currently the most viable solution, but these impose severe restrictions on the type and even the number of algebraic operations that can be performed homomorphically. For instance, division and comparison operations have no homomorphic counterpart, and so any computation to be performed on

encrypted data must avoid or work around these restrictions. The computational trade-offs implied by FHE can be described via an *impossible trinity* between "security", "size" and "usability". Any two of these are possible but not all three, and so we can achieve either (i) small data size and high security at the cost of usability (small amount of homomorphic computation); (ii) high security and usability at the cost of large data size (high homomorphic computational cost); (iii) high usability and small data size at the cost of low security (weak privacy guarantees).

Applications. Privacy-preserving methods, in particular those based on homomorphic encryption, already have applications in virtually all areas of data analysis where sensitive information is present. We highlight applications in the biomedical sciences and related areas, such as medical diagnosis (Ayday et al., 2013b; Kocabaş et al., 2013; Saeb et al., 2015), genomics (Ayday et al., 2013a; Graepel et al., 2013; Lauter et al., 2014), biometric identification (Barni et al., 2015; Yasuda et al., 2013), electronic health monitoring (Abbas and Khan, 2014; Kocabaş and Soyata, 2015), clinical decision support (Basilakis et al., 2015; Liu et al., 2016) and electronic health records management (Benaloh et al., 2009; Fernández-Alemán et al., 2013). Applications to other problems have also been put forward, for instance privacy-preserving web search (Çetin et al., 2016), smart metering (Busom et al., 2016), electronic voting (Mursi et al., 2013), image processing (Ziad et al., 2016), wireless sensor networks and the internet of things (Malina et al., 2016), as well as cloud computing (Atayero and Feyisetan, 2011; Oppermann et al., 2016; Smith, 2014).

Our goal. The restrictions imposed by homomorphic encryption create fascinating challenges in designing algorithms for secure computation that comply with these requirements. Although the number of privacy-preserving methods based on homomorphic encryption has been growing steadily, these typically involve simple algorithms that can be straightforwardly applied in the context of homomorphic computation without considerable modification (Bos et al., 2014; Kim and Lauter, 2015; Wu and Haven, 2012). In contrast, the vast areas of statistics and machine learning remain, for the most part, still poorly researched in that respect. Specifically, traditional methods for solving regression and classification problems are commonly computationally impractical under this new framework, so that computational

trade-offs between alternative methods must be carefully studied. Tailored methods for statistical analysis under FHE may have to be developed in some cases. Our goal is precisely that; to design and implement such algorithms and to provide answers to the question:

how can we do statistics privately under fully homomorphic encryption?

1.5 Thesis outline

After this **INTRODUCTORY CHAPTER** (chapter 1) where we informally introduced the reader to the issues of data privacy and solutions via homomorphic computation, we provide a **BACKGROUND CHAPTER** (chapter 2) where we formalise notions of privacy, cryptography and secure computation, with emphasis on homomorphic encryption; explain the homomorphic principle that is the foundation of homomorphic computation; give a detailed account of the constraints of homomorphic encryption; describe a modern fully homomorphic encryption scheme; and discuss alternative notions of privacy such as anonymisation, differential privacy and secure multi-party computation. In the following three **MAIN CHAPTERS** (chapters 3–5) we develop statistical and machine learning methods tailored for homomorphic computation, namely classification via multinomial and semiparametric naive Bayes algorithms, and regression via accelerated least squares and cooperative, recursive Bayesian updating. In all cases, the solutions proposed take into account the restrictions imposed by fully homomorphic encryption and details are given on trade-offs and computational costs. We finish with a **CONCLUDING CHAPTER** (chapter 6) where we discuss the relevance of this work and give directions for future research. We also provide a set of **APPENDICES** with details on software and implementation, as well as additional results. In particular, we describe the `HomomorphicEncryption` package which implements the `FV` cryptosystem, and the `EncryptedStats` package which implements the statical and machine learning methods developed for homomorphic computation. To aid the reader, we include a **GLOSSARY** of cryptographic terms at the end of this thesis. This contains a short definition of commonly used terms such as 'ciphertext' and 'public-key cryptography'.

Chapter 2

Background on fully homomorphic encryption

In this chapter we give an overview of cryptography, with emphasis on fully homomorphic encryption, in a manner accessible to statisticians and applied scientists. We provide details on its mechanics, possibilities and limitations as relevant for the development and implementation of statistical and machine learning methods. To understand the subsequent chapters, a good knowledge of the limitations imposed by fully homomorphic encryption is essential. They dictate which algorithmic strategies are viable and which ones are impractical in the context of homomorphic computation and, therefore, determine the strategies we will suggest to tackle the statistical problems considered in the main chapters of this thesis. For other high-level introductions see Aslett, Esperança, and Holmes (2015a); Armknecht et al. (2015); Gentry (2010); Micciancio (2010).

Homomorphic Encryption (HE) — from the ancient Greek 'homos' meaning 'same' and 'morphe' meaning 'form' — is a modern encryption technique which allows computation to take place directly with encrypted data, that is, without the need to first decrypt it, and with the guarantee that the result of the encrypted operations, once decrypted, matches the result of the same operations performed using the original, unencrypted data. This means, effectively, that we can perform algebraic operations without seeing the (original) data, which is unlike

classical encryption techniques, where the result of operations involving encrypted data has no meaningful interpretation.

The idea of constructing a *Fully Homomorphic Encryption* (FHE) scheme — one that supports unlimited addition and multiplication over encrypted data — was first proposed by Rivest et al. (1978a), but such a scheme was to remain elusive for the following three decades. Several partially homomorphic schemes appeared in the literature, allowing *either* homomorphic addition *or* multiplication but not both. Boneh (1999) came closest but although this scheme allows unlimited homomorphic additions, it allows only 1 homomorphic multiplication. Finally, in 2009 Craig Gentry constructed the first FHE scheme enjoying strong security guarantees for unlimited homomorphic addition *and* multiplication operations (Gentry, 2009a,b). Since then, a wave of research in FHE ensued and a number of other FHE schemes were proposed (Brakerski, 2012; Brakerski et al., 2012; Gentry et al., 2013; van Dijk et al., 2010). Implementations and optimisations followed suit (Gentry et al., 2012a,b; Halevi and Shoup, 2014, 2015) and applications quickly emerged in virtually all areas where data privacy and secure computation are desirable, as overviewed in chapter 1.

Roadmap. We start by introducing some basic notions of privacy and cryptography (§2.1) and presenting the classical cryptography framework, discussing its limitations for secure computation (§2.2); next, we review the recent field of fully homomorphic encryption, explain the homomorphic principle — which allows computation to be carried out directly with encrypted data — and present the limitations of homomorphic encryption as relevant to encrypted statistical analysis (§2.3); we then describe a modern homomorphic cryptosystem, giving details on its security and the computational costs (§2.4); and finally, we briefly discuss some alternative notions of privacy (§2.5).

2.1 Data privacy

The terms *data privacy* and *data security* (which we will use interchangeably) are often employed loosely as 'umbrella' terms. The difficulty in defining privacy stems from the multitude of considerations that make up the idea of privacy, including people's desires and

expectations, legislation and subjective ethical considerations — all of which are subject to changes over time (Smith, 2014).

A working definition of privacy. In order to make progress, we must lay down definition of data privacy. In this thesis we on focus secure computation and adopt a cryptographic definition of privacy:

Data is said to be secure when it is encrypted and the probability that an attacker with finite resources can break the encryption is negligible.

Encryption is a process which transforms, or *encrypts*, a message into 'something' indecipherable by untrusted parties. This 'something' is typically a noisy, complex object in a high-dimensional space; a scrambled message, virtually impossible to read. To encrypted data we can apply the reverse transformation, which reveals, or *decrypts*, the hidden message. These transformations require a *key* and so, once encrypted, a message can only be read by the key holder. The three algorithms (key generation, encryption and decryption) form the core of any *cryptosystem* or *encryption scheme*.

Most modern cryptosystems base their security on hard problems in number theory which are theoretically possible to solve but computationally too expensive. Security is then based on the high computational cost of breaking the cryptosystem and on the assumption that the attacker has large but finite computational resources. This is the case with virtually all modern encryption schemes, such as the RSA scheme (Rivest et al., 1978b; discussed in §2.2.2), which is the most widely used scheme for transferring messages over the internet. The same holds for homomorphic encryption schemes.

There are several issues important to privacy that this definition ignores, such as integrity (can the message be altered by untrusted parties?), authentication (can the identities of the trusted parties be verified?) and key management (are the keys safely stored?). Although practically important, these aspects are not directly connected with computation, which is the focus of the present work, and will not be discussed extensively here (see, for instance, Heatherly, 2016; van Dijk and Juels, 2010).

The immediate advantages of such a criterion for privacy are twofold: first, data that is secure according to this definition can be stored in untrusted remote locations, for instance in a cloud server; second, and most importantly for statistical analysis, data can be securely shared with untrusted parties for pooling and cooperative computation in a privacy-preserving manner.

2.2 Classical encryption

Although secure computation under the definition of privacy given above requires homomorphic methods, many of the key concepts are shared between homomorphic and non-homomorphic (classical) encryption. This section introduces these concepts using simple examples in classical encryption, which make for a simpler exposition as they are arguably simpler in nature and, therefore, more appropriate to illustrate fundamental principles.

We start with some basic definitions (§2.2.1); give illustrative examples of early and modern-days cryptosystems (§2.2.2); consider security under different types of attacks (§2.2.3); and discuss the limitations of classical cryptography for secure computation (§2.2.4).

2.2.1 Basic notions

By *classical encryption* we mean the set of tools which allow data to be stored and transferred securely according to the privacy definition given in §2.1. We exclude the possibility of computing directly with encrypted data, a task which we associate exclusively with homomorphic encryption.

There are two main classes of cryptosystems (Kessler, 2016): *symmetric* or *private-key* cryptosystems use the same key for encryption and decryption, while *asymmetric* or *public-key* cryptosystems use different keys for encryption and decryption. The obvious advantage of public-key cryptography is that the public key, used only for encryption, can be widely distributed without compromising the security of the system, since the public key provides no information about the secret key, which is required for decryption. In this work we will deal exclusively with public-key cryptosystems as all modern homomorphic encryption schemes belong to this class. The widely used RSA scheme is also a public-key cryptosystem.

The primary objective of a cryptosystem is to hide a *message*. In the cryptography literature, the message to be encrypted is also called a *plaintext*, while the encrypted message is called a *ciphertext*.

For security reasons, a ciphertext should have a number of desirable properties, namely: (i) it should have a noise component, so that several encryptions of the same plaintext produce different ciphertexts with high probability (semantic security); (ii) it should 'look' like an encryption of a random plaintext; in particular, its size should be independent of the plaintext it encrypts (indistinguishability). We give more details on security in §2.2.3.

Cryptosystems can be defined by a few essential components, including a set of parameters $\boldsymbol{\pi}$ and a number of cryptographic functions as follows:

- $\texttt{KeyGen}(\boldsymbol{\pi})$ generates the key pair $(\texttt{pk}, \texttt{sk})$, public and private key respectively, for a given set of cryptographic parameters $\boldsymbol{\pi}$.
- $\texttt{Enc}_{\texttt{pk}}(m)$ encrypts a message (plaintext) m under the public key $\texttt{pk}$.
- $\texttt{Dec}_{\texttt{sk}}(c)$ decrypts an encrypted message (ciphertext) c using the secret key $\texttt{sk}$.

Typically, the cryptographic parameters define a cryptosystem's security and computational properties. Encryption transforms a plaintext $m \in \mathcal{P}$ into a ciphertext $c \in \mathcal{C}$, while decryption reverses this operation. $\mathcal{P}$ and $\mathcal{C}$ denote the plaintext and ciphertext spaces, respectively. The specific form of these cryptographic functions and spaces depends on the encryption scheme used, although for simplicity we omit this dependence from the notation. An illustration of the encryption and decryption processes is shown in Figure 2.1.

The fundamental identity of classical cryptography can then be stated as follows: for any plaintext $m \in \mathcal{P}$ and corresponding ciphertext $c \leftarrow \texttt{Enc}_{\texttt{pk}}(m) \in \mathcal{C}$ we have:

$$\texttt{Dec}_{\texttt{sk}}(c) = \texttt{Dec}_{\texttt{sk}}(\texttt{Enc}_{\texttt{pk}}(m)) = m \tag{2.1}$$

that is, we can recover the original plaintext from its corresponding ciphertext as long as $(\texttt{pk}, \texttt{sk})$ form a valid pair of cryptographic keys.

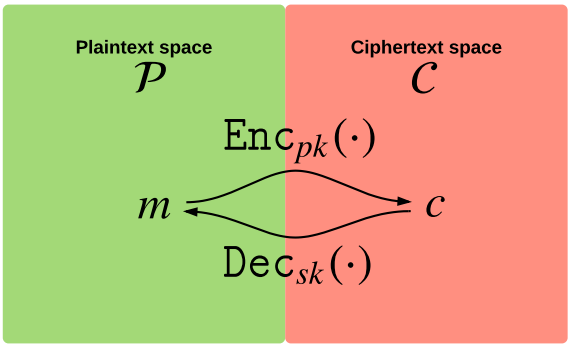


Figure 2.1: Diagram representing the encryption and decryption processes. Encryption, using the public key `pk`, transforms a plaintext m in plaintext space $\mathcal{P}$ (e.g., $\mathbb{Z}_t$, the integer ring modulo t) into a ciphertext $c \leftarrow \mathtt{Enc}_{\mathtt{pk}}(m)$ in ciphertext space $\mathcal{C}$ (e.g., $\mathbb{Z}_q$). Decryption, using the secret key `sk`, reverses this process.

2.2.2 Examples: Caesar cipher and RSA

We give here two examples of classical encryption schemes which will serve to exemplify properties and expose limitations of different types of cryptosystems.

The early days: Caesar cipher

The Caesar cipher — famously used by the Roman dictator Julius Caesar to protect confidential messages in the first century BC — is one of the simplest cryptosystems (Reinke, 1962). It is a type of *substitution cryptosystem*: each letter in a message is encrypted individually by substituting it with another from the same alphabet. If substitution is done by rotating the alphabet a fixed number of positions, it is also called a *rotation cryptosystem* (see Figure 2.2). Plaintext and ciphertext spaces are the same and correspond to the alphabet used.

Key generation. The Caesar cipher is an example of a symmetric or private-key cryptosystem, where public and secret keys are identical and are given by the rotation number, r, which should, therefore, remain secret. A rotation can be chosen uniformly at random from $\{1, \ldots, |\mathcal{P}|\}$, where $|\mathcal{P}|$ is the size of the alphabet.

Encryption and Decryption. Rotation ciphers can be described mathematically using modular algebra and assuming a correspondence between the letters in the alphabet and the natural numbers (e.g., A $\mapsto$ 1, B $\mapsto$ 2, etc.). For a rotation of r letters over the English

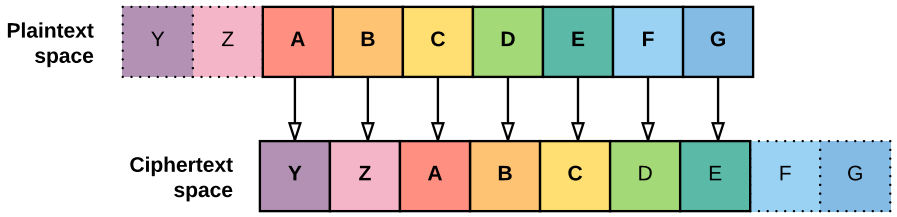


Figure 2.2: A Caesar cipher on the English alphabet (26 letters) with rotation equal to 24 letters forward or 2 letters backward. The top row gives the plaintext alphabet and the bottom row the corresponding ciphertext alphabet.

alphabet (26 letters), we can write the encryption and decryption functions as follows:

$$\texttt{Enc}_r(m) = m + r \pmod{26} \tag{2.2}$$

$$\texttt{Dec}_r(c) = c - r \pmod{26} \tag{2.3}$$

For a concrete example, consider a rotation equal to 24 letters forward, as in Figure 2.2. Encrypting a readable message element-wise leads to an apparently meaningless one,

$$\texttt{encryptedstatistics} \implies \texttt{clapwnrcbqryrgqrgaq}$$

In terms of security, this class of cryptosystems is weak and has, for this reason, been abandoned in practice (details in §2.2.3).

The modern days: RSA

Modern encryption schemes built their foundations on number theory. Their security is based on number theoretical problems which are difficult to solve in the sense that (i) brute force attacks are computationally impractical; and (ii) no practical solutions exploiting the cryptosystem's design are believed to exist. RSA is an example of such a cryptosystem (Rivest et al., 1978b). We note that the following exposition refers to the simplest version of RSA, called unpadded RSA, and is not used in practice for security reasons. The more secure and widely used version, padded RSA, is discussed in §2.2.3.

Key generation. RSA involves a triplet $\boldsymbol{\pi} = (e, d, n)$ representing, respectively, the public exponent, secret exponent and modulus of the scheme, which are defined as follows. Let p and q be two large prime numbers. Set the modulus to be the composite number $n = pq$ and compute $\phi(p, q) = (p - 1)(q - 1)$. Next, choose the public exponent $1 < e < \phi(p, q)$ to be a coprime of $\phi(p, q)$, i.e., $\gcd(e, \phi(p, q)) = 1$; and the secret exponent d to be the modular multiplicative inverse of $e \pmod{\phi(p, q)}$, i.e., $d = e^{-1} \pmod{\phi(p, q)}$. This gives the key pair $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{KeyGen}(p, q)$ where $\mathtt{pk} = (n, e)$ is and $\mathtt{sk} = (n, d)$. Because knowledge of $(p, q, \phi(p, q))$ can be used to compute d, these should also remain secret.

The security of RSA is based on the fact that even with the knowledge of e and n, finding d is extremely difficult. This is because computing the secret exponent d requires that we solve the integer factorisation problem of decomposing n into its prime factors p and q, an arguably hard problem in number theory (Boneh, 1999).

Encryption and Decryption. To encrypt an integer message $m \in \mathbb{Z}_n$ or decrypt a ciphertext $c \leftarrow \mathtt{Enc}_{\mathtt{pk}}(m)$, we compute

$$\mathtt{Enc}_{\mathtt{pk}}(m) = m^e \pmod{n} \tag{2.4}$$

$$\mathtt{Dec}_{\mathtt{sk}}(c) = c^d = (m^e)^d = m \pmod{n} \tag{2.5}$$

RSA turns out to be a partially homomorphic cryptosystem. It supports encrypted multiplication, which is easily verified:

$$\mathtt{Enc}_{\mathtt{pk}}(m_1) \cdot \mathtt{Enc}_{\mathtt{pk}}(m_2) = (m_1 m_2)^e \pmod{n} = \mathtt{Enc}_{\mathtt{pk}}(m_1 m_2). \tag{2.6}$$

2.2.3 Security: key sizes, noise and hard problems

A *cryptographic attack* is any attempt to break an encryption scheme, and a successful attack implies that the security of the data encrypted under this scheme has been compromised. There are many types of cryptographic attacks, and different encryption schemes are vulnerable to different attacks depending on their characteristics. Here we overview some of the

most common attacks and some important properties of cryptosystems which can help avoid security vulnerabilities.

Typically, it is assumed that both the encryption scheme and the public key are public knowledge, while the secret key and potentially some cryptographic parameters are secret.

Key sizes *versus* the brute force attack

A *brute force attack* (or *exhaustive key search*) is the simplest type of cryptographic attack. It works by systematically trying out all possible keys until the correct one is found. Brute force attacks are usually very slow as they require the evaluation of potentially all possible keys. Cryptosystems are designed to make this task virtually impossible by defining a key space so large as to render any attempt hopeless, even in the presence of extraordinary (but finite) computational resources.

Consider RSA with secret keys of 1024 bits. There are $2^{1024} \approx 1.8 \times 10^{308}$ possible such keys. Trying even a fraction of these would take billions of years with all the world's computer resources, and so the answer to the question *how long would it take to break RSA-1024 by brute force?* is, for all practical purposes, *forever*. On the other hand, the Caesar cipher is very weak against this type of attack as even for large alphabets (hundreds of elements) one can try all possible keys (i.e., rotations) very quickly. Note that in the case of the Caesar cipher $\mathcal{P} = \mathcal{C}$, and so observing ciphertexts tells us something about the size of the alphabet.

Thus, the size of the secret key provides security guarantee against brute force attacks, which is considered a basic requirement for security.

Semantic security *versus* the frequency analysis and dictionary attacks

Other attacks may be designed that exploit structural weaknesses of particular cryptosystems, using that information to create more efficient attacks. A *frequency analysis attack* exploits a fault in *semantic security* — the absence of randomness during the encryption process which translates into the same ciphertext being produced when a plaintext is encrypted multiple

times, under the same public key. Semantically secure schemes, on the other hand, will output a different encryption of the same message on multiple trials with high probability.

This means that if a scheme is not semantically secure, with basic information about the message space (e.g., language), an eavesdropper can match the frequencies of ciphertexts to the frequencies of plaintext space elements. For instance, the relative frequency of the letter "e" in the English language is about 12.702%. A ciphertext that matches this frequency in a sample will likely correspond to an encryption of the letter "e", and so information about the underlying plaintext is leaked.

An even simpler type of attack, called the *dictionary attack*, can be launched against semantically insecure cryptosystems when an attacker has access to the public key and has some basic knowledge about the message space. By simply encrypting all potential messages under this public key, it is possible to create a dictionary unambiguously mapping plaintexts to ciphertexts.

Noise is then an essential part of the encryption process to safeguard against these types of attacks. Neither the Caesar cipher nor the unpadded RSA cryptosystems presented earlier is semantically secure, as encryption is deterministic. In practice, RSA uses an additional reversible protocol before encryption, called "padding", which introduces randomness in the ciphertexts and protects them from attacks that exploit flaws in semantic security.

Number theory *versus* the design exploitation attack

A third type of threat to security, the *design exploitation attack*, uses knowledge of the design of the cryptosystem (which is assumed to be public) to build a more sophisticated attack. For example, in the case of RSA we known that the secret key is generated from two prime numbers p and q (unknown), which compose the scheme modulo $n = pq$ (known). An obvious attack route — in fact, the most efficient — is to solve the factorisation problem of decomposing n into its prime factors p and q. For large primes p and q this is believed to be a hard problem in the sense that no efficient methods to solve it exist nor are expected to be found (Boneh, 1999).

Additionally, schemes whose security is based on number theory typically afford increases in security by setting the parameters of the scheme appropriately. For instance, if more efficient design exploitation attacks targeting RSA are found, it is possible to increase security by choosing larger cryptographic parameters, (p, q), thus making the factorisation problem harder to solve.

2.2.4 Limitations

Classical encryption in its modern form, e.g., RSA, has been tremendously successful in providing privacy for data both in transit and in storage due to its efficiency (lightweight) and provable security. Yet, the price to pay is that encrypted data is rendered unusable by virtue of the transformations required to ensure security. In other words, classically encrypted data is unusable for computation or, conversely, if computation is to be performed, the data must first be decrypted, putting its security in jeopardy.

Hence, in the context of cloud and distributed computing we are faced with the 'all-or-nothing' trade-off *security vs. usability*; that is, we can have either security or data usability, but not both simultaneously. The aim of homomorphic encryption is precisely to solve this problem. In practice, it substitutes the previous trade-off with one of the type *security vs. computational cost* — overall less restrictive on computation than the trade-off dictated by classical encryption.

2.3 Homomorphic encryption

To be considered fully homomorphic, an encryption scheme must allow an unlimited number of additions and multiplications to be performed on ciphertexts without losing the ability to correctly decrypt the result. In between non-homomorphic and fully homomorphic there is a range of *Partially Homomorphic Encryption* (PHE) schemes. Some schemes are homomorphic for a single operation only (addition or multiplication), such as Rivest et al. (1978b), homomorphic for multiplication; Paillier (1999), homomorphic for addition/subtraction and multiplication (by constant only); ElGamal (1985), homomorphic for multiplication and exponentiation (by constant only); Goldwasser and Micali (1982), homomorphic for exclusive-or

(addition modulo 2); Benaloh (1994), homomorphic for addition/subtraction. Others are homomorphic for both addition and multiplication but in a very restricted sense, for instance Boneh et al. (2005), which is homomorphic for addition and multiplication but allows only 1 multiplication to be performed.

A more interesting part of the homomorphic spectrum is found in the group of *Somewhat Homomorphic Encryption* (SHE) schemes. These schemes support homomorphic addition and multiplication but are limited in the number of operations that can be performed before ciphertext noise grows too large and correct decryption becomes impossible. The precise number depends on the set of cryptographic parameters $\boldsymbol{\pi}$. Fortunately, in most applications the algorithm is known beforehand and so $\boldsymbol{\pi}$ can be chosen accordingly so as to allow the required numbers of operations to be performed, giving rise to a *leveled FHE* scheme. In the remaining of this work we will use the term FHE in the sense of leveled FHE.

In fact, the FHE construction of Gentry (2009b) started from a SHE scheme and coupled this with a re-encryption operation which effectively refreshes ciphertexts, bringing its noise down to that of a freshly encrypted ciphertext (details in §2.3.3).

We start by giving some basic definitions (§2.3.1) and explaining the homomorphic principle (§2.3.2); we then show how the ciphertext noise can be kept in check by re-encrypting ciphertexts (§2.3.3) and discuss the significance of this operation for FHE (§2.3.4); and finally we describe some key application areas of FHE (§2.3.5).

2.3.1 Basic notions

Let us start with some basic definitions. These will be important to explain the significance of FHE as a landmark in secure computation and also its limitations.

Let $\mathbb{B} = \{0, 1\}$ denote the binary set. A *logic gate* is an implementation of a boolean function $f : \mathbb{B}^k \to \mathbb{B}$ which takes k bits as inputs, performs a set of logical operations, and produces a bit as the output. For instance, `AND` gates perform logical conjunction (outputting 1 only when both inputs are 1) while `XOR` gates perform logical exclusive disjunction (outputting 1 when only one of the inputs is 1). A *boolean circuit* is a series of logical gates that compute a certain function. The concept can be extended to the non-binary case, but the binary case

will be especially important to understand the theoretical implications of FHE (discussed in §2.3.4). Informally, for most purposes a circuit can be thought of as an algorithm.

Another important concept in homomorphic computation is the *multiplicative depth* of an algorithm. By definition, this is the number of consecutive multiplication operations that are used, which should not be confused with the total number of multiplications. The difference is best explained via an example: consider the following two algorithms, where all x_i are freshly-encrypted ciphertexts:

$$A_1 = \sum_{i=1}^{N} x_i^2 = x_1 \times x_1 + x_2 \times x_2 + \cdots + x_N \times x_N \tag{2.7}$$

$$A_2 = \prod_{i=1}^{N+1} x_i = x_1 \times x_2 \times \cdots \times x_{N+1} \tag{2.8}$$

Even though both algorithms use the same total number of multiplications, namely N, they have very different multiplicative depths, namely 1 and N, respectively. Because ciphertext noise grows with homomorphic operations, and especially so with multiplications (details in §2.3.3), the multiplicative depth of an algorithm determines the computational efficiency of homomorphic operations and should be taken into consideration when designing algorithms for homomorphically computation. To continue with the example above, all terms x_i^2 in A_1 have a comparable level of noise and their addition does not increase this level substantially, the reason being that noise is symmetric around 0, and so some of it cancels out during addition. In A_2 the situation is more severe; noise growth is strictly cumulative and so the level of noise increases more quickly with each successive multiplication.

2.3.2 The homomorphic principle

Additionally to the components described in §2.2.1 — key generator function outputting the key pair, $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{KeyGen}(\boldsymbol{\pi})$; encryption function outputting a ciphertext, $c \leftarrow \mathtt{Enc}_{\mathtt{pk}}(m)$; and decryption function outputting a plaintext, $m \leftarrow \mathtt{Dec}_{\mathtt{sk}}(c)$ — a FHE scheme also admits an evaluation function:

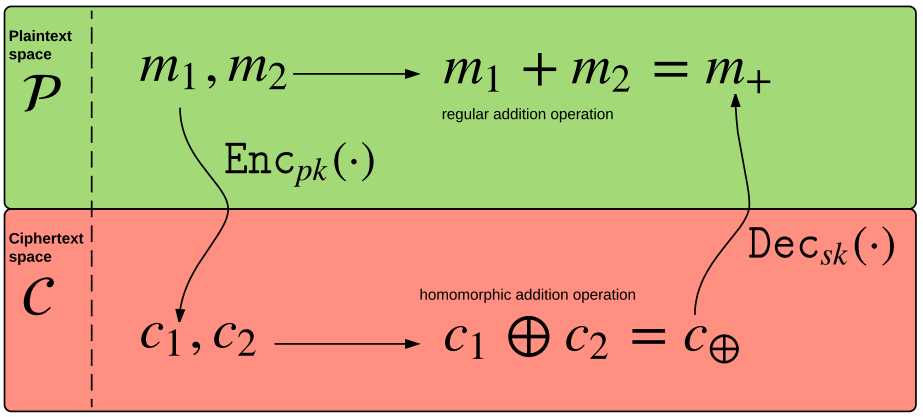


Figure 2.3: The homomorphic principle: the result of applying homomorphic addition ($\oplus$) to ciphertexts, $c_1 \oplus c_2 \equiv c_\oplus$, matches the result of applying regular addition ($+$) to plaintexts, $m_1 + m_2 \equiv m_+$, that is, $m_+ = \mathtt{Dec}_{\mathtt{sk}}(c_\oplus)$.

- $\mathtt{Eval}(f, \{c_1, \ldots, c_N\})$ evaluates a circuit f of the ciphertexts $c_i \leftarrow \mathtt{Enc}_{\mathtt{pk}}(m_i)$ for all $i \in \mathbb{N}_{1:N}$, such that the homomorphic correctness property holds:

$$\mathtt{Dec}_{\mathtt{sk}}(\mathtt{Eval}(f, \{c_1, \ldots, c_N\})) = f(\{m_1, \ldots, m_N\}). \tag{2.9}$$

In words, the *homomorphic correctness property* or *homomorphic principle* (2.9) means that we can perform any computation f directly on ciphertexts with the guarantee that the decrypted result will match that of the same computation applied to the corresponding plaintexts (see Figure 2.3 for an illustration). The only restriction imposed on f is that it can be efficiently represented as a circuit, for computational reasons.

As mentioned earlier, a FHE allows homomorphic addition and multiplication. An alternative way to state the homomorphic principle is as follows: let $(\mathtt{pk}, \mathtt{sk})$ be a valid key pair and $c_1, c_2 \in \mathcal{C}$ denote encryptions of $m_1, m_2 \in \mathcal{P}$ under $\mathtt{pk}$, respectively. A cryptosystem is said to be fully homomorphic if there exist operators $\oplus$ and $\otimes$ acting in ciphertext space such that

$$\mathtt{Dec}_{\mathtt{sk}}(c_1 \oplus c_2) = m_1 + m_2 \qquad \text{and} \qquad \mathtt{Dec}_{\mathtt{sk}}(c_1 \otimes c_2) = m_1 \times m_2 \tag{2.10}$$

that is, there exist operators in ciphertext space that correspond to addition and multiplication in plaintext space, such that the homomorphic principle holds. Using these two identities,

and composition, we can write arbitrary circuits (restricted to the class that uses only addition and multiplication) that can be evaluated homomorphically. The circuit f in (2.9) is, therefore, restricted to algorithms that use only addition and multiplication. Although this may sound unimpressive, as we will show in §2.3.4, these are sufficient, in theory, to implement any computation.

The operations $\oplus$ and $\otimes$ are typically more complicated than regular integer arithmetic operations and depend on the cryptosystem used. For instance, they may be polynomial addition and multiplication when ciphertexts are polynomials (Fan and Vercauteren, 2012), or matrix addition and multiplication when ciphertexts are matrices (Gentry, Sahai, and Waters, 2013).

A ciphertext that is the result of homomorphic operations should retain the semantic security and indistinguishability properties described earlier (§2.2.1), that is, it should be noisy, and difficult to distinguish from other ciphertexts, independently of the algorithm that they are a result of, a property known as circuit privacy (Gentry, 2009b).

2.3.3 Bounding ciphertext noise

As operations are performed with ciphertexts, the noise component grows: it is added in homomorphic addition and multiplied in homomorphic multiplication. If the noise of a freshly encrypted ciphertext is bounded by B, then evaluating a multiplication circuit with depth L leads to a level of noise B^L or even B^{2^L} (Fan and Vercauteren, 2012). Once the noise has crossed a certain threshold, a ciphertext can no longer be decrypted correctly and the message is lost. Therefore, a FHE scheme must also include a mechanism to keep the noise bounded (Vaikuntanathan, 2011).

The way Gentry (2009b) originally constructed his FHE scheme was by transforming a SHE scheme into a FHE scheme by means of an operation called *bootstrapping*[1] or *refreshing* which effectively re-encrypts a ciphertext homomorphically, that is, without exposing the underlying plaintext in the process.

[1] This should not be confused with the usage of the term "bootstrapping" in the context of statistics.

Let $f_{\mathtt{Dec}}$ denote the decryption circuit or algorithm. This is will be different for every encryption scheme, but an example is shown in (2.15) for the case of the `FV` scheme. Formally, the bootstrap operation can be described as follows:

- $\mathtt{Recrypt}_{\mathtt{rk}}(c') = \mathtt{Eval}_{\mathtt{rk}}(f_{\mathtt{Dec}}, c')$ evaluates the decryption circuit $f_{\mathtt{Dec}}$ under the re-encryption key $\mathtt{rk} \leftarrow \mathtt{Enc}_{\mathtt{pk}'}(\mathtt{sk})$, where $\mathtt{pk}'$ is a second public key and $c' \leftarrow \mathtt{Enc}_{\mathtt{pk}'}(c)$ is an encryption of c under $\mathtt{pk}'$.

We can break down the re-encryption operation as follows: Suppose we have a ciphertext $c \leftarrow \mathtt{Enc}_{\mathtt{pk}}(m)$, which may be the result of some homomorphic computation and whose noise we want to refresh. First, we encrypt c under a second public key $\mathtt{pk}'$ and obtain $c' \leftarrow \mathtt{Enc}_{\mathtt{pk}'}(c) = \mathtt{Enc}_{\mathtt{pk}'}(\mathtt{Enc}_{\mathtt{pk}}(m))$, which amounts to doubly encrypting the original message m. Then, we encrypt the secret key $\mathtt{sk}$ under $\mathtt{pk}'$ to obtain the re-encryption key $\mathtt{rk} \leftarrow \mathtt{Enc}_{\mathtt{pk}'}(\mathtt{sk})$. Finally, we evaluate the decryption circuit $f_{\mathtt{Dec}}$ under $\mathtt{rk}$ to obtain

$$c^\star \leftarrow \mathtt{Eval}_{\mathtt{rk}}(f_{\mathtt{Dec}}, c') = \underbrace{\mathtt{Enc}_{\mathtt{pk}'}(\mathtt{Dec}_{\mathtt{sk}}(c))}_{\dagger} = \mathtt{Enc}_{\mathtt{pk}'}(m) \tag{2.11}$$

which represents an encryption of m under $\mathtt{pk}'$, and whose noise level is comparable to that of a freshly encrypted ciphertext. Note that the expression marked with †, in particular the decryption of c, is never actually performed 'in the open' during the re-encryption process, which would compromise he privacy of m; the equality is guaranteed by the correctness properties of homomorphic encryption only.

Put simply, the second layer of encryption serves to hide the re-encryption of c. The entire process is effectively done homomorphically, thus preserving the privacy of both the encrypted message m and the secret key $\mathtt{sk}$. The procedure relies on *circular security*, which means that it is safe to encrypt the secret key under its own public key.

2.3.4 Theoretical possibilities and limitations

Recall the definitions of *circuit* and *logic gate* presented in §2.3.1 and let $\mathbb{B} = \{0, 1\}$ denote the boolean set. Crucially for homomorphic computation, boolean circuits can be constructed to

compute *any* arbitrary function $f : \mathbb{B}^k \to \mathbb{B}$ using only `AND` and `XOR` gates, corresponding to multiplication, subtraction and addition (modulo 2) in the boolean case. That is, if $x, y \in \mathbb{B}$ then these gates can be computed as $\texttt{AND}(x, y) = xy$ and $\texttt{XOR}(x, y) = x + y \pmod 2 = x + y - 2xy$, using only addition, subtraction and multiplication (Gentry, 2010). If we view the elements of $\mathbb{B}$ as bits, then using a homomorphic scheme that supports unlimited addition and multiplication (i.e., fully homomorphic) we can compute any arbitrary function in the encrypted domain (Gentry, 2010; Wu, 2015). The construction of such a scheme would constitute a major breakthrough in the field of cryptography, and that was precisely what Gentry (2009b) achieved by successfully designing the bootstrapping procedure presented in the previous section.

Although the theoretical possibility is real, in practice FHE schemes that operate on binary spaces are remarkably inefficient. Consider the simple addition of two integer messages: each message would have to be decomposed into its binary representation, each bit encrypted separately, and then operations performed at the level of the bits as in boolean circuits. More seriously, ciphertext noise is a function of the depth of the circuit evaluated, and so computing complex functions is not straightforward when we must also keep ciphertext noise bounded. This has been partly overcome in more recent encryption schemes that encrypt and operate directly on integers, although these are also subject to restrictions (discussed in §2.4.3). Implementations of bootstrapping are still considerably slow, taking around 30 minutes per ciphertext bootstrapped, for reasonable security levels (Gentry and Halevi, 2011; Halevi and Shoup, 2015).

We emphasize that FHE hides the data used in computation, but does not hide the computation itself, i.e., the algorithm. *Functional Encryption* is a generalisation which allows a key holder to learn the value of a function of encrypted data without learning the data (as in FHE) nor the function. That is, given an encrypted value $x' \leftarrow \texttt{Enc}_{\texttt{pk}}(x)$ and a functional secret key $\texttt{sk}[f]$ associated with a function f, the holder of $\texttt{sk}[f]$ can obtain $f(x) \leftarrow \texttt{Dec}_{\texttt{sk}[f]}(x')$, in such as way that neither the data x nor the function f are exposed (Boneh et al., 2011, 2012). Though promising, the area is still in its early stages of development.

2.3.5 Application framework

The most important application of FHE is in situations where computation is to be outsourced to an untrusted server, for example, a cloud (Gentry, 2009b; Lauter et al., 2011). This can be either a two-party framework — where a server performs some computation on behalf of a client, who sees only the final result — or a multi-party framework — where several parties cooperate to compute a function of their joint data without any of the parties revealing information about their portions of the data to the remaining parties (Choudhury et al., 2013). There are many reasons why outsourcing computation is desirable, including convenience, economic and efficiency reasons. However, the main objections to the use of cloud computing still revolve around issues of privacy and security (Armbrust et al., 2010).

Secure cloud computing would enable computational services, such as prediction of disease risk (important to inform clinical decisions), to be delegated to a third party without compromising the privacy of patients. Let's consider this illustrative example in more detail. The process can broken down into 3 phases:

(1) DATA UPLOAD: user data collected via wearable devices or by medical institutes is uploaded to a cloud server after being encrypted. The upload can be done directly by the patients in the first instance, or by the medical institutes on behalf of the patient in the second. This step may also involve pre-processing such as especial encoding transformations or data standardisation (see §3.1).

(2) COMPUTATION: once the data has been uploaded, patients or authorised agencies can perform computation directly on encrypted data, potentially pooling this data with data from other patients. For instance, bipolar patients may be interested in receiving electronic, personalised feedback regarding medication changes following a self-assessment test of their mood fluctuations (see §4.5.2.1); or research agencies may be interested in finding new patterns of association between behaviour and disease in large-scale studies by pooling data from many wearable device users. Computation can be performed securely by virtue of homomorphic encryption.

(3) RESULT RETRIEVAL: after computation has taken place, the results can be retrieved by an authorised party in possession of the secret key. This step may also involve post-processing such as rescaling of parameters (see §4.3.1.2).

2.4 The `FV` homomorphic cryptosystem

The `FV` cryptosystem (Fan and Vercauteren, 2012) is a modern, fully homomorphic encryption scheme based on the scheme of Brakerski (2012). In this scheme, ciphertexts are polynomial ring elements with large integer coefficients, and all homomorphic operations consequently take place in this space. The security of the `FV` scheme is given by the ring version of the 'learning with errors' problem, described in §2.4.2. The scheme enjoys the important properties of semantic security and indistinguishability, meaning that (i) encryption introduces a random noise component in the ciphertexts, and that, on different attempts under the same public key, the same plaintext will be encrypted to different ciphertexts with high probability; (ii) any ciphertext 'looks like' any other, in particular has the same size as any other, even if it is the result of some homomorphic operation.

The `FV` scheme is implemented in the `R` package `HomomorphicEncryption`, developed by Louis Aslett and briefly described in §A.1 (see also Aslett, Esperança, and Holmes 2015a). All computational aspects of homomorphic encryption given in this thesis (e.g., processing times and memory requirements) relate to this implementation.

We start by describing the `FV` scheme, giving details on cryptographic parameters and functions, and homomorphic operations (§2.4.1); next, we explain the ring version of the 'learning with errors' problem (RLWE) and how it is used in the `FV` scheme to establish security (§2.4.2); we then give an extensive overview of the constraints of FHE in general and the `FV` scheme in particular, which are important when designing and implementing secure statistical algorithms (§2.4.3); and finally, we give details on the computational costs of all cryptographic and homomorphic operations as well as memory requirements, and their dependence on key cryptographic parameters (§2.4.4).

2.4.1 Description

Notation

- $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denotes the set of integers in $(-q/2, q/2]$, for some $q \in \mathbb{N}$.
- $\mathbb{Z}[x]$ and $\mathbb{Z}_q[x]$ denote polynomials in x over fields $\mathbb{Z}$ and $\mathbb{Z}_q$, that is, whose coefficients belong to $\mathbb{Z}$ and $\mathbb{Z}_q$, respectively. Concretely, $\mathbb{Z}_q[x] = \{a_r x^r + \cdots + a_0 : a_n \in \mathbb{Z}_q, \forall n \in \mathbb{N}_{0:r}\}$ and similarly for $\mathbb{Z}[x]$, where r is the degree of the polynomial.
- $\Phi_r(x)$ denotes the cyclotomic polynomial of degree r, the unique irreducible polynomial (it cannot be factorised) with integer coefficients which is a divisor of $x^r - 1$ and is not a divisor of $x^n - 1, \forall n < r$; for example, $\Phi_6(x) = x^2 - x + 1$. The degree of the cyclotomic polynomial is typically chosen to be of the form $r = 2^d$, so that $\Phi_{2^d}(x) = x^{2^{d-1}} + 1$ for all $d \in \mathbb{N}$. This choice is computationally convenient because it implies maximal sparsity such that polynomial arithmetic modulo $\Phi_{2^d}(x)$ can be performed very efficiently (Lyubashevsky et al., 2013).
- $R = \mathbb{Z}[x]/\Phi_{2^d}(x)$ and $R_q = \mathbb{Z}_q[x]/\Phi_{2^d}(x)$ denote polynomial quotient rings with coefficients over fields $\mathbb{Z}$ and $\mathbb{Z}_q$, respectively. In particular, these are polynomial rings modulo a cyclotomic polynomial, that is, the remainders of Euclidean division of elements of $\mathbb{Z}[x]$ or $\mathbb{Z}_q[x]$ by the cyclotomic polynomial $\Phi_{2^d}(x)$. More concretely, we can write $R_q = \{a_n x^n + \cdots + a_0 : a_n \in \mathbb{Z}_q, \forall n \in \mathbb{N}_{0:2^d}\}$ and similarly for R. For example, with $q = 5$ and $d = 3$ we have the cyclotomic polynomial $\Phi_8(x) = x^4 + 1$ and the ring $\mathbb{Z}_5[x]/(x^4 + 1) = \{a_8 x^8 + \cdots + a_0 : a_n \in \mathbb{Z}_5, \forall n \in \mathbb{N}_{0:8}\}$. Note that the parameter q determines the size of the field $\mathbb{Z}_q$, while the parameter d determines the degree of the polynomials in the ring R_q. We refer to these objects as *cyclotomic polynomial rings* (Ducas and Durmus, 2012).
- χ denotes the discrete Gaussian probability distribution bounded by $(-B, B)$ on R, where B is usually a function of a dispersion parameter, σ.
- $[a]_q$ denotes the modulo q reduction of a, such that if $a \in R$ then $[a]_q \in R_q$, and where the reduction is applied separately to each coefficient of a.

- $\sim$ denotes sampling, uniformly at random if only the space is given ($a \sim R_q$) or from a particular distribution if specified ($a \sim \chi$). By sampling a cyclotomic polynomial ring element a we mean sampling the coefficients of a independently, according to the specified distribution.

Preliminaries

The plaintext and ciphertext spaces. Both spaces are cyclotomic polynomial rings: the plaintext space is $\mathcal{P} = R_t$ and the ciphertext space is $\mathcal{C} = R_q \times R_q$. The values t and q are cryptographic parameters.

Cryptographic parameters. The `FV` scheme has a set of cryptographic parameters $\boldsymbol{\pi} = (d, t, q, \sigma)$ which regulate the security level: d determines the degree of the polynomial rings $\mathcal{P}$ and $\mathcal{C}$; t and q determine the magnitudes of the coefficients in $\mathcal{P}$ and $\mathcal{C}$; and σ controls the ciphertext noise. Combined, these parameters determine the level of security of the scheme.

Message encoding. To embed the integer message $m \in \mathbb{Z}$ in ciphertext space we first represent it (before encryption) as a binary polynomial $\mathring{m}(x)$ whose coefficients correspond to its bit representation, that is, $\mathring{m}(x) = \sum_{n=0}^{2^d} a_n x^n$ where a_n is the nth of b bits of m when written in binary form and $a_n = 0, \forall n > b$. Evaluating the polynomial $\mathring{m}(x)$ at $x = 2$ yields the original message, $m = \mathring{m}(2) = \sum_{n=0}^{2^d} a_n 2^n$. For example, the integer $m = 13$ can be represented in binary code as 1101 (least significant bit on the right), which would be encoded in polynomial form as $\mathring{m}(x) = x^3 + x^2 + 1$. This is a natural representation since all elements in the scheme are then polynomials, as will become clear shortly.

Key generation

Secret key. The *secret key*, represented by k_s, is simply an element sampled from the polynomial ring R_2 uniformly at random, that is, a polynomial of degree 2^d with binary coefficients,

$$k_s \sim R_2 \quad \text{uniformly at random.} \tag{2.12}$$

Public key. The public key, represented by k_p, is a pair of polynomials, each an element in the polynomial ring R_q,

$$k_p = (k_{p1}, k_{p2}) = ([-(a \cdot k_s + e)]_q, a) \in R_q \times R_q \tag{2.13}$$

where $a \sim R_q$, $e \sim \chi$ and '$\cdot$' denotes element-wise multiplication. The secret key is securely hidden inside the public key via the RLWE problem (details in §2.4.2).

Encryption

The *encryption function*, $\texttt{Enc}_{k_p}(m)$, encrypts a plaintext $m \in R_t$ under the public key k_p. It produces a ciphertext which is a pair of polynomials, each an element in R_q,

$$c = (c_1, c_2) \leftarrow \left(\left[k_{p1} \cdot u + e_1 + \left\lfloor \frac{q}{t} \right\rceil \mathring{m} \right]_q, [k_{p2} \cdot u + e_2]_q \right) \in R_q \times R_q \tag{2.14}$$

where $u, e_1, e_2 \sim \chi$. Note that all objects are polynomials, thus the importance of writing m as a polynomial $\mathring{m}(x)$ as well. This also allows for a larger plaintext space (see the discussion on polynomial coefficient size in §2.4.3). Note also the randomness introduced in the ciphertexts, which makes the scheme semantically secure.

Encryption is an element-wise operation: if $\boldsymbol{X}$ a data matrix, then encrypting $\boldsymbol{X}$ means encrypting each of its entries separately, so that each data point (plaintext) is transformed into a ciphertext. There are ways to pack multiple plaintexts into one ciphertext using the Chinese Remainder Theorem (see §6.2), but we will not use this technique in this thesis.

Decryption

The *decryption function*, $\texttt{Dec}_{k_s}(c)$, decrypts a ciphertext $c \in R_q \times R_q$ using the secret key k_s. It recovers the polynomial representation $\mathring{m}(x)$ of the original message m,

$$\mathring{m}(x) \leftarrow \left[\left\lfloor \frac{t[c_1 + c_2 \cdot k_s]_q}{q} \right\rceil \right]_t \in R_t \tag{2.15}$$

such that $\mathring{m}(2) = m$.

Decryption is correct as long as the noise remains below a certain threshold. To see this, write $[c_1 + c_2 \cdot k_s]_q = \lfloor q/t \rfloor \cdot \mathring{m} + v$, where $v = e \cdot u + e_1 + e_2 \cdot k_s$ is the noise contained in the ciphertext. Now, using the definition of modulo we can write $c_1 + c_2 \cdot k_s = \lfloor q/t \rfloor \cdot \mathring{m} + v + q \cdot r$ and multiplying this by t/q gives $c_1 + c_2 \cdot k_s = \mathring{m} + (t/q)(v - \epsilon \cdot \mathring{m}) + t \cdot r$ where $\epsilon = q/t - \lfloor q/t \rfloor < 1$. Correct decryption is then guaranteed for $(t/q)||v - \epsilon \cdot \mathring{m}|| < 1/2$, in which case the noise is eliminated by rounding (for more details see Fan and Vercauteren, 2012).

Homomorphic addition

Homomorphic addition, represented by $\oplus$, is achieved by regular polynomial addition followed by modulo q reduction. Letting $c_i = (c_{i1}, c_{i2}) \leftarrow \mathtt{Enc}_{\mathtt{pk}}(m_i)$ and $c_j = (c_{j1}, c_{j2}) \leftarrow \mathtt{Enc}_{\mathtt{pk}}(m_j)$ denote ciphertexts encrypted under the public key k_p,

$$c_i \oplus c_j = \left([c_{i1} + c_{j1}]_q , [c_{i2} + c_{j2}]_q \right) \tag{2.16a}$$

$$= \left(\left[k_{p1} \cdot u_\oplus + e_{\oplus 1} + \left\lfloor \frac{q}{t} \right\rceil (\mathring{m}_i + \mathring{m}_j) \right]_q , [k_{p2} \cdot u_\oplus + e_{\oplus 2}]_q \right) \tag{2.16b}$$

which according to (2.14) is a valid encryption of $m_i + m_j$ under the public key k_p. Notice that the noise terms are additive in this case, for instance $u_\oplus = u_i + u_j$, and similarly for the other composite noise terms, $e_{\oplus 1}$ and $e_{\oplus 2}$.

Homomorphic multiplication

Homomorphic multiplication, represented by $\otimes$, is considerably more complicated than addition because it increases the length of the ciphertext,

$$c_i \otimes c_j = \left(\left[\left\lfloor \frac{t}{q}(c_{i1} \cdot c_{j1}) \right\rceil \right]_q , \left[\left\lfloor \frac{t}{q}(c_{i1} \cdot c_{j2} + c_{i2} \cdot c_{j1}) \right\rceil \right]_q , \left[\left\lfloor \frac{t}{q}(c_{i2} \cdot c_{j2}) \right\rceil \right]_q \right). \tag{2.17}$$

To obtain a valid ciphertext of standard length we must apply a relinearisation operation (see Fan and Vercauteren, 2012 for details). Importantly, ciphertext noise grows multiplicatively in this case, which imposes constraints on the depth of the circuits that can be evaluated homomorphically, as discussed in §2.4.3.

By using the definition of ciphertext (2.14) in the equation above, we can see that the noise is multiplicative. As a result, ciphertext noise increases more rapidly with ciphertext multiplication than with ciphertext addition.

2.4.2 Security: the 'learning with errors' problem

The security of the `FV` cryptosystem is based on the *ring learning with errors* problem (RLWE; Lyubashevsky et al., 2010), which is the polynomial ring equivalent of the *learning with errors* problem (LWE; Regev, 2005, 2010). Here we describe briefly the RLWE problem and relate it to the security of the `FV` scheme.

Recall the polynomial ring notation from §2.4.1. Let q be a modulo parameter, and $s \sim R_q$ be a secret draw. Now, choose $a_i \in R_q$ independently and uniformly at random, and $e_i \in R$ independently according to χ. Finally compute $b_i = a_i \cdot s + e_i \pmod q \approx a_i \cdot s$ for 'small' noise e_i. Then, for $i \in \mathbb{N}$ we have a set of pairs $(a_i, b_i) \in R_q \times R_q$. The values a_i and b_i are known while e_i and s are unknown. The RLWE problem is then to find the polynomial s given arbitrarily many pairs (a_i, b_i). For a suitable distribution χ this can be shown to be at least as hard as the *short vector problem* in lattices, which is a well studied and difficult problem to solve, hence providing strong security guarantees (Lyubashevsky et al., 2010).

Note that the RLWE problem is embodied in the public key generation step (2.13), where we can identify the public key with a pair of polynomials (a_i, b_i) obtained as described above. Breaking the encryption scheme, that is, recovering the secret from the public key (the most promising approach), corresponds precisely to solving the RLWE problem. Intuitively, by construction it should be difficult to distinguish pairs (a_i, b_i) generated as above from truly uniformly random pairs $(a_i, u_i) \in R_q \times R_q$, implying that the secret key is safely embedded in the public key (Silverberg, 2013).

Additionally, lattice-based cryptography — to which RLWE-based schemes belong — enjoys very strong security proofs, based on worst-case hardness; are relatively simple to implement; and are believed to be secure against quantum attacks. These schemes are, therefore, good candidates for post-quantum cryptography (Bos et al., 2016; Micciancio and Regev, 2009; Peikert, 2014).

2.4.3 Constraints

All work presented in the main chapters (3–5) follows, more or less directly, from the constraints imposed by FHE, and so the importance of this section cannot be overstated. While limitations vary slightly between cryptosystems, a large number of these are common to virtually all modern FHE schemes. Here we give an overview of the most important constraints as relevant for the development and implementation of statistical and machine learning algorithms for homomorphic computation.

In general, designing such algorithms demands that computation be viewed from a very low-level perspective; thought must be put into what an algorithm requires and what can be achieved under FHE at the level of basic algebraic operations.

Plaintext space. All modern FHE schemes restrict the plaintext space to binary, $\mathcal{P} = \{0, 1\}$; integer, $\mathcal{P} = \mathbb{Z}$; or a more complicated ring structure, for instance an integer polynomial ring, $\mathcal{P} = \mathbb{Z}_q[x]/f(x)$, as in the case of `FV`. This means that direct encryption of real values is not supported. Current solutions include working directly with rational approximations by encrypting numerator and denominator separately and using the usual rules of arithmetic for fractions (very inefficient); or making use of the large plaintext space available to encode real values as large integers in a way that preserves relative distances between data points (one such method is introduced in §3.1).

No division. Homomorphic division is currently not supported, i.e., there is no FHE scheme for which there exists a homomorphic operation $\oslash$ such that $\texttt{Dec}_{\texttt{sk}}(c_1 \oslash c_2) = m_1/m_2$. The consequent setbacks for homomorphic computation vary in degree of severity. In some cases it is possible to avoid division altogether by rescaling the algorithms appropriately. Examples of this approach are given in chapters 4 and 5, and in Aslett, Esperança, and Holmes, 2015b, where the algorithms can be written to give results proportional to the true answer. To obtain the correct answer, the decrypted result must be rescaled by a factor which is independent of the data and consequently leaks no information.

No comparison. Comparison (i.e., inequality checking: `<`, `>`, `==`) is another key operation which is unavailable in ciphertext space. The most significant setback here is in algorithmic control flow: algorithms like decision trees and Markov chain monte carlo methods require

the evaluation of data-dependent logical conditions to decide what action to perform next (e.g., setting $y = a$ if $x < c$, or $y = b$ otherwise). Algorithms of this nature cannot be straightforwardly implemented when x is encrypted. One mitigating proposal is working with partitioned data with small numbers of partitions (e.g., quintiles), and evaluating logical conditions using only homomorphic addition and multiplication as follows: let the value $x \in \mathbb{N}_{1:5}$ be represented by a binary vector $\check{\boldsymbol{x}} = (\check{x}_1, \check{x}_2, \check{x}_3, \check{x}_4, \check{x}_5)$ with $\check{x}_i \in \{0, 1\}$ and $\sum_{i=1}^{5} \check{x}_i = 1$, which encodes the quintile information of x, that is, $\check{x}_i = 1 \Rightarrow x = i$. If x is known only in encrypted, extended form $\check{\boldsymbol{x}}$, then computing $\check{x}_3 \cdot \prod_{i \neq 3}(1 - \check{x}_i)$ gives an encryption of 1 if and only if $\check{x}_3 = 1$, which is equivalent to testing whether $x = 3$ using only operations that can be performed homomorphically. For arbitrary datasets, this requires the special encoding method briefly described above and presented in full detail in §3.1.1, which also has limitations of its own. Examples are given in §3.3 in the context of a multinomial naive Bayes classifier; and in Aslett, Esperança, and Holmes (2015b) where the authors build a classifier akin to random forests but where splits are done completely at random. While allowing some form of rudimentary encrypted comparison to take place, it does not solve the algorithmic control flow problem, for which no solution currently exists using only homomorphic methods.

Non-polynomial function evaluation. Evaluating non-polynomial functions, e.g., e^x or $\sqrt{x}$, although theoretically possible, is problematic in practice when x is an encrypted value. Simple methods such as polynomial approximations (e.g., Taylor series expansions) are unreliable as the approximation must be done in the neighborhood of x for accurate results, which is simply impossible if x is a ciphertext. Other methods, such as iterative methods, lookup tables and smart tricks generally require inaccessible operations like division and equality checking. One of the challenges in secure statistical computation is, therefore, to design algorithms that can avoid complex function evaluations.

Data inflation. Data inflation is related to semantic security, which is necessary for security reasons (see §2.2.3). In modern FHE schemes data inflation is typically of the order of thousands. For instance, in the `FV` scheme, ciphertexts are polynomials of high degree, typically 4096, with large integer coefficients, typically 128 bits. Storing one single encrypted

element is, therefore, considerably more costly in terms of memory requirements than storing one integer or real value (see §2.4.4 for details).

Computational cost of homomorphic operations. While storing homomorphically encrypted data long-term is probably undesirable in most circumstances, the computational costs are a real problem. As a consequence of data inflation, homomorphic operations are typically orders of magnitude slower than their non-homomorphic counterparts, which imposes restrictions on computationally intensive methods. For instance, in the `FV` scheme, ciphertext addition corresponds to addition of polynomials with large integer coefficients, which are considerably more expensive than standard integer arithmetic (see §2.4.4 for details).

Circuit depth. The maximum multiplicative depth of a circuit (MMD; defined and illustrated in §2.3.1) is one of the most important parameters to consider when designing algorithms for homomorphic computation. Ciphertext noise grows with homomorphic operations, especially so with multiplication, and so in a leveled FHE scheme we are restricted in the depth of the circuit that we can evaluate before the noise grows too large and correct decryption becomes impossible. Although Gentry's bootstrapping procedure can theoretically refresh ciphertext noise at any point, it is a complicated and very expensive operation which remains impractical in most scenarios (around 30 minutes per ciphertext bootstrapped, for reasonable security levels; see §2.3.4). As mentioned in the introduction to this chapter, cryptographic parameters in leveled FHE schemes can be set according to the level of security and amount of computation desired. Therefore, the ciphertext size can be increased — higher degree polynomials and larger coefficients — to accommodate algorithms with higher MMD at the cost of considerably higher memory requirements and slower homomorphic operations (see §2.4.4 for details). By increasing the ciphertext size, it is possible for ciphertexts to contain more noise (implying a combination of higher security and more homomorphic computation), and yet still decrypt correctly.

A practical note on MMD: the implementation of `FV` takes a cryptographic parameter for MMD (denoted `L`; see appendix A). This refers to the depth in a simple multiplication circuit of the type $\prod_{i=1}^{N} x_i$, for which $\texttt{L} = N$ is sufficient to guarantee correct decryption. If other

intermediate operations are involved, for instance, $\sum_{j=1}^{P}\prod_{i=1}^{N} x_{ij}$, a higher value `L` may be necessary to guarantee correctness in decryption.

Polynomial coefficient size. As explained in §2.4.1, a plaintext m is first converted into a binary polynomial $\mathring{m}(x)$ before encryption. The coefficients of this polynomial must remain bounded throughout computation to ensure correct decryption. The default maximum coefficient size (MCS) in the `HomomorphicEncryption` package is arbitrarily set to $2^{14} = 16,384$, meaning that the coefficients of $\mathring{m}(x)$ must remain in the range $(-16,383, 16,384)$. Note that this restriction is not on the magnitude of m, which can be very large. As an example, we are guaranteed correct decryption after $16,383$ additions of fresh ciphertexts — the worst case scenario, when adding the same value $16,383$ times — although likely many more in real applications. Compare this with the alternative of working with $m \in \mathbb{Z}$ (equivalent to the polynomial of degree zero $\mathring{m}(x) = m$) where we would expect to hit the upper bound $16,383$ much earlier in the process of performing homomorphic operations. Increasing the MCS naturally increases the cost of homomorphic operations and memory requirements (see §2.4.4 for details).

These restrictions raise numerous difficulties for homomorphic computation — particularly in the implementation of statistical and machine learning algorithms — as will become clear in the subsequent chapters. Arguably, the most important of these constraints are the inexistence of division and comparison operations, and the low circuit depth restrictions.

2.4.4 Computational aspects

In this section, we present some important computational aspects of the `FV` scheme. We give processing times of homomorphic operations and memory requirements as a function of important cryptographic parameters which define the security and computational properties of the encryption scheme. For instance, a higher level of security and a higher multiplicative depth required both translate into a heavier encryption scheme and have an adverse impact on computational costs.

The results given below are relative to the implementation of the `FV` cryptosystem available in the `R` package `HomomorphicEncryption` (Aslett, Esperança, and Holmes, 2015a). Unless

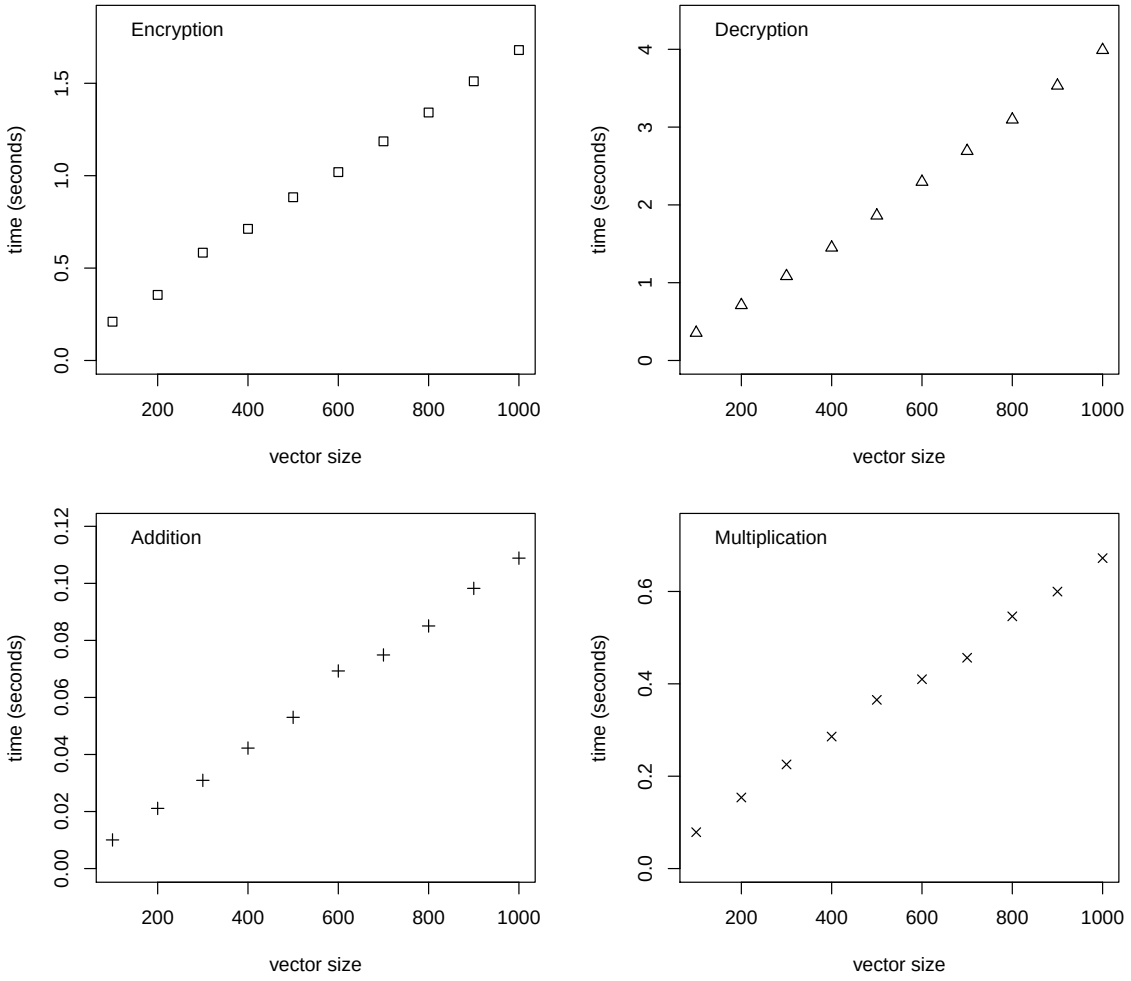


Figure 2.4: The computational cost of homomorphic operations is linear in the data size, a vector $\boldsymbol{v}$ in this example, for any set of fixed cryptographic parameters (default shown). **[top left]** encryption: $\boldsymbol{v}' \leftarrow \mathtt{Enc}_{\mathtt{pk}}(\boldsymbol{v})$; **[top right]** decryption: $\boldsymbol{v} \leftarrow \mathtt{Dec}_{\mathtt{sk}}(\boldsymbol{v}')$; **[bottom left]** addition: $\boldsymbol{v}' \oplus \boldsymbol{v}'$; **[bottom right]** multiplication: $\boldsymbol{v}' \otimes \boldsymbol{v}'$.

otherwise stated, we used the default cryptographic parameters of this implementation. Processing times are relative to a 48-core machine (for full details on hardware and software see appendix A.3).

Processing time

All basic homomorphic operations have linear processing time in the size of the data, for any fixed set of cryptographic parameters (Figure 2.4). Encryption and decryption, although expensive, are one-time operations and so carry relatively little weight overall. Homomorphic addition and multiplication are much slower than the corresponding regular integer arithmetic

operations. For instance, using the default cryptographic parameters we can perform close to 1500 homomorphic multiplications per second; for regular integer multiplications that number is in the billions.

In the `HomomorphicEncryption` package implementation, the user can set the maximum multiplicative depth (MMD, `L`), maximum coefficient size (MCS, `max`) and minimum security level (MSL, `lambda`) desired. Using bounds in the literature, these are used to derive a set of valid cryptographic parameters satisfying the required MMD, MCS and MSL (Lepoint and Naehrig, 2014; Lindner and Peikert, 2011; see also §2.4.3 and appendix A). Increasing any of these 3 parameters increases the computational costs associated with homomorphic operations — as expected, since they regulate different aspects of security (Figure 2.5). The growth pattern is step-like discontinuous due to non-smoothnesses in the way some cryptographic parameters are set. When a parameter is increased, to achieve a fixed level of security some parameters can be decreased, thus the non-smooth, and even non-monotonic, increasing schedules.

Memory requirements

The high computational costs are a reflex of the increase in size when data is encrypted, itself a consequence of semantic security. In general, the data inflation factor is of the order of thousands. Figure 2.6 gives a visual representation of the magnitude of data inflation. The exact expansion factor is a function of the cryptographic parameters regulating security. Figure 2.7 gives the ciphertext size as a function of two key cryptographic parameters MMD and MCS, with others remaining constant at default values, whenever possible. The growth pattern is step-like discontinuous due to non-smoothnesses in the way some cryptographic parameters are set; as an example, the current implementation doubles the size of the polynomial whenever any such increase is necessary, thus the steep rise in memory requirements at some points.

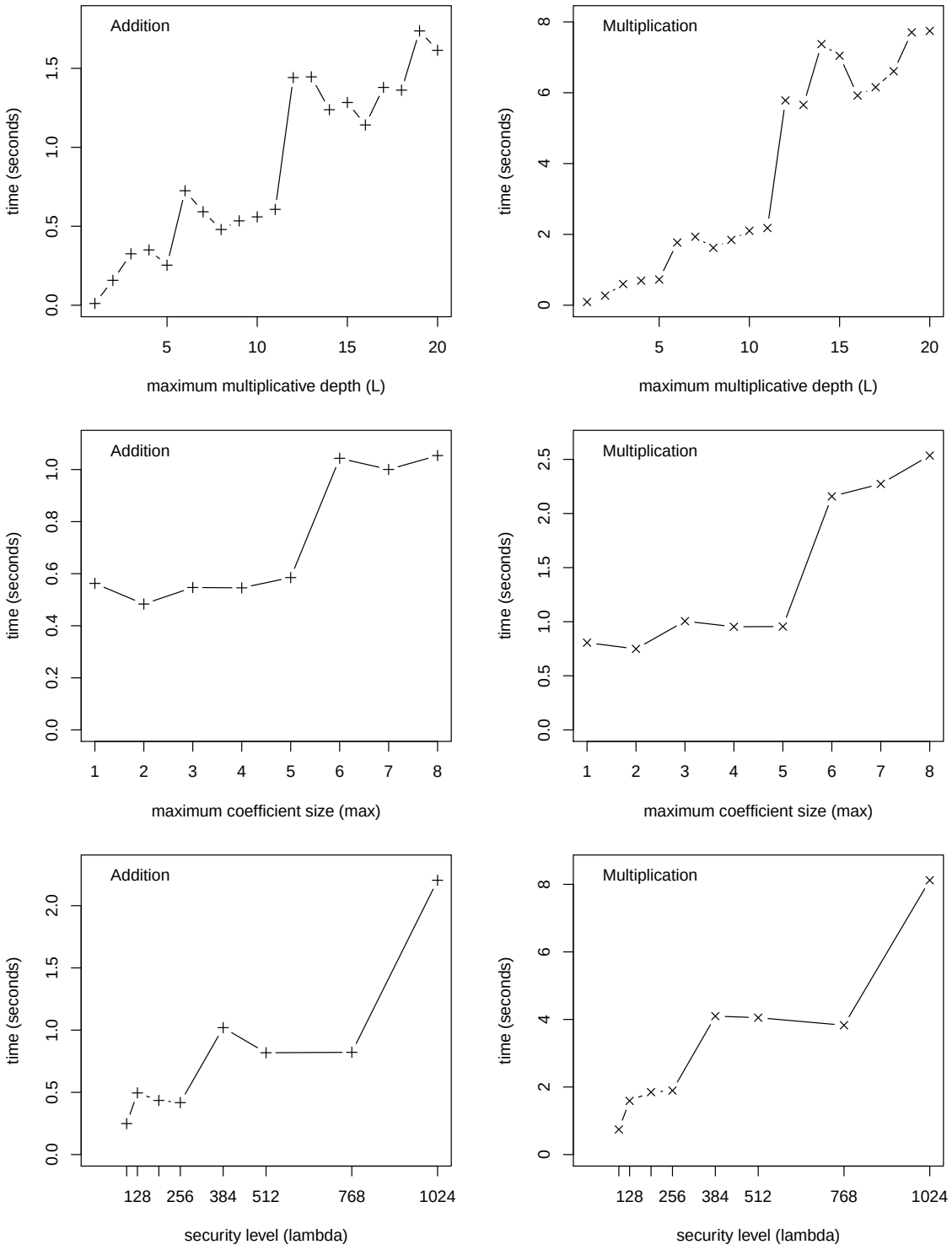


Figure 2.5: The computational cost of homomorphic operations — addition $\boldsymbol{v}' \oplus \boldsymbol{v}'$ (on the left) and multiplication $\boldsymbol{v}' \otimes \boldsymbol{v}'$ (on the right) — grows with all the main cryptographic parameters. The values correspond to runtime per 100 operations (i.e., $\boldsymbol{v}'$ is of length 100) averaged over 10 trials. **[top]** maximum multiplicative depth (`L`) **[middle]** maximum coefficient size (`max`), shown in $\log_{10}$ scale. **[bottom]** security level (`lambda`).

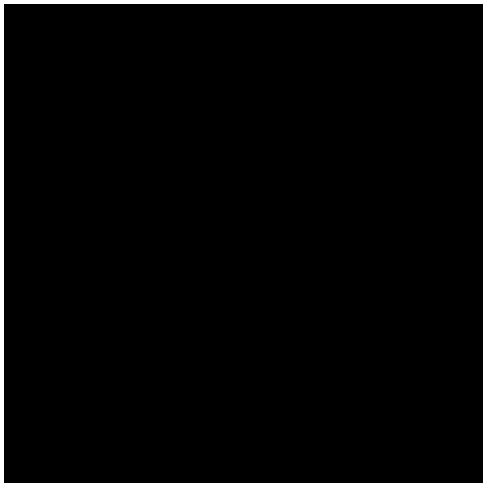


Figure 2.6: Representation of data inflation (ciphertext size in memory) when homomorphically encrypting a single value under the `FV` cryptosystem: **[left]** original message, 4 bytes (tiny dot at the bottom); **[right]** encrypted message, 131072 bytes (0.13 MB). [default cryptographic parameters from the `R` package `HomomorphicEncryption`]

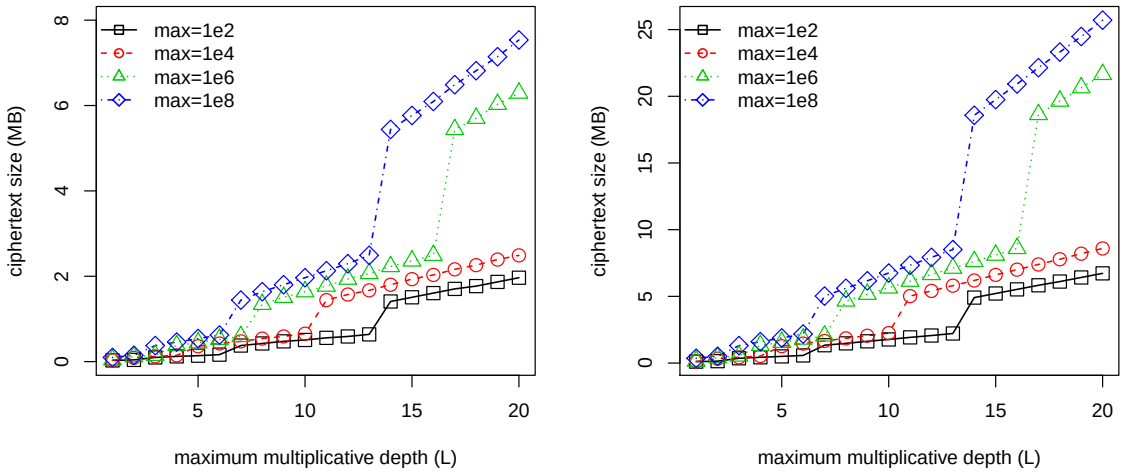


Figure 2.7: The ciphertext size grows with the two cryptographic parameters maximum multiplicative depth (`L`) and maximum coefficient size (`max`). The growth pattern is step-like discontinuous due to non-smoothnesses in the way cryptographic parameters are set, for security reasons. **[left]** size in memory, excluding overheads; **[right]** compressed size on disk, including overheads (e.g., relinearisation key).

2.5 Alternative notions of privacy

We focus, in the remaining of this work, on the strong, cryptography-based notion of privacy as embodied in homomorphic encryption. Yet, other notions of privacy exist that allow for secure computation, each with their idiosyncrasies. Here, we describe briefly the most influential of these, namely anonymisation (§2.5.1), differential privacy (§2.5.2) and secure multi-party computation (§2.5.3).

2.5.1 Anonymisation

Anonymisation or *de-identification* techniques have been widely used for disclosing sensitive information, for instance health data (El Emam et al., 2011), and deserve our attention for that reason.

Definition. Anonymisation is the practice of removing information from an individual's record which can potentially uniquely identify him or her. This includes personal information like name and social security number, which contain little information from a research perspective, but for security reasons may have to be extended to other types of information like area of residence or school attended, which may contain valuable information for research. Importantly, and unlike other notions of privacy such as cryptography and differential privacy, anonymisation techniques do not have provable security guarantees (El Emam et al., 2011; Narayanan and Shmatikov, 2010; Ohm, 2010).

Re-identification. Methods for *re-identification* of individual records, also known as *de-anonymisation* techniques, have been shown to be effective in re-identifying individuals from anonymised data, using the fact that each individual has a very particular signature materialised in his characteristics, interests and preferences. As a result, anonymisation techniques have generally performed below expectations. Examples of privacy breaches in the context of anonymised databases are numerous and can be achieved by simply cross-referencing public databases (Gymrek et al., 2013; Sweeney, 2013; Sweeney et al., 2013) or simply analysing transaction records, social networks or telephone metadata (de Montjoye et al., 2013, 2015; Mayer et al., 2016; Narayanan and Shmatikov, 2006, 2008, 2009). See the discussion on privacy breaches in chapter 1 for more details on some of these studies.

Limitations. In the face of these facts, anonymisation seems insufficient to guarantee data privacy at a satisfactory level. Generalisations of anonimisation methods such as k-anonimity — which requires each individual record to be indistinguishable from at least $k-1$ individuals also included in the database — have been proposed but retain many of the same problems, specifically the difficulty in balancing information loss and disclosure risk (Domingo-Ferrer and Torra, 2008; Sweeney, 2002).

2.5.2 Differential privacy

Differential Privacy (DP), originally proposed by Dwork (2006), is an alternative notion of privacy which works by perturbing the output of database queries so as to minimise the probability that individuals in the dataset can be identified (Dankar and El Emam, 2013; Dwork and Roth, 2014).

Definition. Let $\epsilon \in \mathbb{R}$ denote a security parameter; f denote an arbitrary query; and $\mathcal{A}_f$ denote a randomised algorithm to respond to query f. Then, $\mathcal{A}_f$ is said to be ϵ-differentially private if for all datasets D, D' differing on at most one record, and all $\mathcal{S} \subseteq \text{Range}(\mathcal{A}_f)$, we have:

$$\mathbb{P}[\mathcal{A}_f(D) \in \mathcal{S}] \leq e^{\epsilon} \cdot \mathbb{P}[\mathcal{A}_f(D') \in \mathcal{S}] \tag{2.18}$$

where the probabilities are over the randomness of $\mathcal{A}_f$. In words, this means that the presence or absence of a single individual (row) from the dataset does not considerably change the output distribution of the randomised algorithm $\mathcal{A}_f$, thus providing *probabilistic indistinguishability*. In guaranteeing individual privacy, the careful randomisation of the algorithm is the key: it provides security by effectively masking the effect on the algorithm output of including or excluding a record from the dataset. The parameter ϵ defines the maximum amount of information that is allowed to be leaked from the database.

Importantly, privacy in this framework is provided at the level of the algorithm and not at the level of the data. In practice, this means that the curator (who is in charge of $\mathcal{A}_f$) must keep the data secure from potential attackers, while simultaneously answering queries from interested parties. In particular, this notion is insufficient for outsourcing computation or storage.

Sensitivity. The *sensitivity* Δ_f of a query algorithm f, defined as

$$\Delta_f = \max_{D,D'} ||f(D) - f(D')||_1 \tag{2.19}$$

is a measure of how much the output of f can change as a result of the presence or absence of an individual record, in the worst case scenario. $||z||_1$ represents the L_1 norm of z. Note

that the sensitivity is calculated using the deterministic output f rather the corresponding randomised version $\mathcal{A}_f$.

To achieve DP, the level of noise injected into the query output, that is, in $\mathcal{A}_f$, must be proportional to the sensitivity of the query. Naturally, the larger the noise level, the less accurate the randomised algorithm output. As an example, the simplistic Laplace mechanism can be shown to be differentially private by injecting Laplace distributed noise with standard deviation equal to $\sqrt{2}(\Delta_f)/\epsilon$ to a query output (Dwork and Roth, 2014).

For low-sensitivity algorithms (e.g., counts and frequencies) the result can change only slightly, implying that only a low level of noise is required, and so we are able to achieve high accuracy and privacy simultaneously. For high-sensitivity algorithms (e.g., sum and maximum) the result can vary dramatically and, consequently, privacy and accuracy may not be simultaneously achievable.

Privacy budget and multiple queries. So far we have assumed that a single query is being performed. If multiple queries are to be performed, say q, the level of noise required to ensure DP over the q queries must scale linearly with q. Viewed from another perspective, a fixed ϵ implicitly defines a *privacy budget*, with each individual query having privacy parameter $\epsilon_i = \epsilon/q$. Therefore, to keep a constant privacy level per query ϵ_i when increasing q, the global privacy budget ϵ must be increased as well, implying larger noise and lower accuracy.

Limitations. This discussion demonstrates the two main limitations of DP (Dankar and El Emam, 2013; Kifer and Machanavajjhala, 2011). First, it may be difficult to achieve security for high-sensitivity queries because the large maximum (potential) change in the output due to the insertion or deletion of one record requires a higher level of noise to be injected into the algorithm output, leading to low accuracy. Second, it does not seem easy to achieve security in the case of multiple queries as the overall privacy budget must be divided by the total number of queries q, and a higher q will lead to lower accuracy per query (if ϵ is increased accordingly) or a lower level of security per query (if ϵ is kept fixed). Even if the number of queries per party is limited, the possibility of having colluding parties undermines the privacy mechanism. Finally, DP alone is insufficient to guarantee privacy in cases where outsourcing computation is desirable, because DP functions more like a secure data access mechanism,

run by trusted parties and targeted at untrusted users, as opposed to a mechanism for secure computation on untrusted servers, which is what FHE aims at.

2.5.3 Secure multi-party computation

Secure multi-party computation (SMPC) is another notion of privacy which allow parties to securely share information with each other.

Definition. In secure two-party computation, two parties, A and B (Alice and Bob), each holding a corresponding piece of data, d_A and d_B, wish to jointly compute a function $f(d_A, d_B)$ without revealing any information about their individual data to the other party. The function f is public while the data is (to be kept) private. A secure protocol for f is then any algorithm that can compute $f(d_A, d_B)$ without either party learning anything about the other's data, apart from the output $f(d_A, d_B)$ itself. That is, Alice does not learn d_B and Bob does not learn d_A. The SMPC approach is to design a protocol that lets Alice indirectly share d_A with Bob in such a way that Bob is incapable of learning d_A. The definition can naturally be extended to the multi-party case.

Protocols. Many protocols have been proposed that satisfy the definition above, for instance, oblivious transfer (OT; Du and Atallah, 2001; Lindell and Pinkas, 2009) and garbled circuits (GC; Snyder, 2014; Yao, 1982). A technical description of these protocols would be a digression here. In general, all SMPC protocols allow information to be shared among parties, and provide strong cryptographic guarantees that any party's private information is not learned by any of the remaining parties in the process.

Limitations. SMPC protocols require intensive communication between the parties involved, which can be very expensive when a strong cryptographic framework is also employed, thus raising issues of practicality (Du and Atallah, 2001; Hall et al., 2011). Additionally, there are also protocol-specific overhead costs or other implementation difficulties; for instance, Yao's GCs requires that the function being evaluated be expressed as a circuit, which can be extremely inefficient for complex problems, i.e., when the corresponding circuit representation is large; more specifically, the *communication complexity* is proportional to the size of the circuit (Gentry, 2009b; Lindell and Pinkas, 2009). SMPC is also unsuitable for outsourcing

computation, as it provides no security layer at the level of the data as standard cryptographic methods do.

FHE-based SMPC. One interesting area of research is the use of FHE and SMPC jointly. For instance, we can outsource computation using FHE but allow multiple rounds of communication between an untrusted server and a trusted (or at least honest-but-curious) client who can perform non-homomorphic computation (divisions, comparisons, etc) and refresh ciphertext noise. The cost here is essentially computational since transferring ciphertexts can be expensive, but in return we benefit from a larger set of algorithms that can be implemented (Choudhury et al., 2013; Damgård et al., 2012; Lagendijk et al., 2013; López-Alt, 2014). The fact that intermediate values are being decrypted may raise some concerns regarding security. In this case, DP can be used to perturb the intermediate outputs so that even an honest-but-curious party is unable to learn any sensitive information in the process.

2.6 Summary

In this chapter... we surveyed the field of cryptography, highlighting the recent subfield of fully homomorphic encryption (FHE). We presented basic notions of data privacy and classical cryptography, and provided examples to illustrate relevant concepts such as security and limitations of different types of encryption schemes. In particular, we discussed indistinguishability (ciphertexts look alike) and semantic security (multiple ciphertexts for each plaintext), two fundamental properties of modern encryption schemes. We then introduced FHE formally, describing the homomorphic principle (that computation can be performed directly on encrypted data, i.e., without first decrypting it, under strong security and correctness guarantees); explained the relevance of Gentry's 2009 breakthrough for secure computation (the construction of the first FHE scheme); considered the main application areas of FHE (cloud, distributed and cooperative computation); and discussed the main constraints of FHE for homomorphic computation as relevant to statistics and machine learning, namely: no division or comparison operations, no non-polynomial function evaluations and restricted circuit depth. Following this, we described a modern FHE scheme (`FV`) which is implemented in the `R` package `HomomorphicEncryption`. Finally we discussed some alternative notions of

privacy (anonymisation, differential privacy and secure multi-party computation) and their respective limitations for privacy-preserving computation.

In the next chapters... we take on the challenging task of designing and implementing statistical and machine learning algorithms in the encrypted domain. FHE imposes heavy penalties on complex algorithms and completely excludes some common statistical and machine learning methods from being implemented, at least without considerable modifications (Aslett, Esperança, and Holmes, 2015b, Graepel et al., 2013). In particular, we highlight the consequences of circuit depth: high computational costs and memory requirements, as well as potential need for bootstrapping. While high computational costs are to some extent unavoidable due to the nature of FHE, algorithms can be designed in order to avoid the very extreme costs of bootstrapping. In the remainder of this thesis we develop statistical and machine learning methods amenable to homomorphic computation that balance accuracy and computational costs, while at the same time avoiding bootstrapping.

Chapter 3

Semiparametric naive Bayes classification

In this chapter we propose a semiparametric naive Bayes (SNB) algorithm for homomorphic computation in the cloud which performs both training and prediction without requiring intermediate rounds of communication. The algorithm is tailored to homomorphic computation in the sense that approximations are taken which would be avoidable in an unrestricted computational scenario but become imperative in the restricted framework of FHE.

The naive Bayes (NB) classifier is a popular generative classification algorithm that models the joint probability of predictor variables independently for each response class, and then uses Bayes' rule and an independence assumption among predictors to construct a simple classifier (Hastie et al., 2009). The advantages and disadvantages have been extensively described in the literature. In particular, although the independence assumption underlying NB is often violated, the linear growth in complexity for large number of predictors and the simple closed-form expressions for the decision rules make the approach attractive in big-data situations (Ng and Jordan, 2002; Rennie et al., 2003). Moreover, because there is an important distinction between classification accuracy (predicting the correct class) and accurately estimating the class probability, NB can perform well in terms of classification error rate even when the independence assumption is violated by a wide margin (Domingos and Pazzani, 1997). Essentially, although it produces biased probability estimates, this does

not necessarily translate into a high classification error (Hand and Yu, 2001). Consequently, NB remains a well established and popular method.

Related work. In the context of secure computation, classification has received considerable attention. Graepel et al. (2013) propose secure binary classification methods for homomorphic computation in the cloud, namely linear means and linear discriminant classifiers. The authors offer division-free equations for training the algorithms. Encrypted prediction is also possible, subject to one offline comparison. In the same spirit, Bos et al. (2014) and Bost et al. (2014) propose additional classification methods by pairing HE and SMPC. While in the first work the model is known to all parties involved and in the second work (which includes NB classification) it remains private, in both cases only prediction is done in a privacy-preserving manner (in ciphertext space); the training step is assumed to have been completed offline (in plaintext space). These approaches, therefore, support encrypted prediction but not estimation and, additionally, require extensive communication between the parties involved, which is computationally expensive. Our approach can perform both estimation and prediction in the encrypted domain without requiring intermediate communication between server and client.

Roadmap. We start by defining important data representations and encoding techniques, which are crucial when encrypting non-integer data (§3.1), and describing the standard framework of the naive Bayes classifier (§3.2); next, we give a simple implementation of a secure multinomial naive Bayes classifier that exploits a specific encoding strategy (§3.3), and a general purpose, semiparametric classifier based on naive Bayes as well as an approximation to simple logistic regression tailored for homomorphic computation (§3.4); we then present some results on the performance of classification and computational costs (§3.5); and finally, discuss (§3.6) and summarise (§3.7) our findings.

3.1 Data representation and encoding

Recall from §2.4.3 that modern FHE schemes restrict the plaintext space to an integer ring $\mathbb{Z}_q = \{n : n \in \mathbb{Z} \wedge n \in (-q/2, q/2]\}$ or similar space. To deal with real-valued data, we must apply an appropriate data transformation and encoding before encryption. Here, we

discuss quantisation (also known as discretisation) and three possible encoding strategies, for categorical, ordinal and real-valued variables.

3.1.1 Quantisation

A simple quantisation strategy is to define non-overlapping partitions on the support of each variable. Let $\boldsymbol{X} = \{x_{ij}\}$ denote an $N \times P$ data matrix with N rows of observations and P columns of predictors. For each predictor $j \in \mathbb{N}_{1:P}$ we can define the partition $K^{(j)} = \{K_1^{(j)}, \ldots, K_{M_j}^{(j)}\}$ of predictor j's support such that for all $i \in \mathbb{N}_{1:N}$ we have $x_{ij} \in \bigcup_{m=1}^{M_j} K_m^{(j)}$ and $K_m^{(j)} \cap K_n^{(j)} = \emptyset, \forall m \neq n$. Given the partition $K^{(j)}$, we can then define a vector variable $\check{\boldsymbol{x}}_{ij}$ that maps the original values x_{ij} to a binary vector of length M_j defining the partition that x_{ij} belongs to:

$$x_{ij} \mapsto (\check{x}_{ij1}, \ldots, \check{x}_{ijM_j}) = (0, \ldots, 0, \underbrace{1}_{\text{position} = x_{ij}}, 0, \ldots, 0) = \check{\boldsymbol{x}}_{ij} \tag{3.1}$$

with the restrictions $\{\check{x}_{ijm}\}_{m \in \mathbb{N}_{1:M_j}} \in \{0, 1\}$ and $\sum_{m=1}^{M_j} \check{x}_{ijm} = 1$. Here, $\check{x}_{ijm}$ refers to the ith element of the column of $\check{\boldsymbol{X}}$ corresponding to bin m of predictor j.

As an example, for real-valued variables, a natural partition is that defined by the quintiles (percentiles 0%–20%, 20%–40%, etc). After partitioning the data, the encoding shown in (3.1) can be used. Discretisation into quintiles is a reasonable approach, widely used in the literature and can even be desirable for preventing overfitting (Liu et al., 2002).

For ordinal partitions on real-valued data we can simply use the partition index as the new representation. For instance, for two real-valued variables that have been discretised into quintiles, this can be represented as follows:

$$\begin{pmatrix} 55.2 & 1.71 \\ 72.5 & 1.92 \\ 61.7 & 1.63 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 3 \\ 5 & 5 \\ 4 & 1 \end{pmatrix}. \tag{3.2}$$

For categorical partitions — or ordinal partitions where the effect of different levels is not constant across the range of levels — we can use the encoding of (3.1), illustrated below,

where each of the two original variables is assumed to have been discretised into 5 different levels (e.g., quintiles):

$$\begin{pmatrix} 1 & 3 \\ 4 & 5 \\ 2 & 1 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.3}$$

This encoding method is a valuable addition to the homomorphic computation toolbox. It is very flexible, allowing for arbitrary partitions on real-valued data, and, moreover, allows for an elementary type of conditional counting in the encrypted domain, as follows. Suppose one wishes to count, say, the number of 2s in a vector $\boldsymbol{x} = (x_1, \ldots, x_N)$, and that $\boldsymbol{x}$ is to be encrypted. As a consequence of encryption, there is no direct way of achieving this. However, by encoding each element of $\boldsymbol{x}$ as in (3.1) before encryption, that is, $x_i \mapsto (\check{x}_{i1}, \ldots, \check{x}_{iM}) = \check{\boldsymbol{x}}_i$ for all $i \in \mathbb{N}_{1:N}$, we can perform this operation indirectly in ciphertext space by computing $\sum_{i=1}^{N} \check{x}_{i2}$, which gives an encryption of the number of 2s in the encrypted vector $\boldsymbol{x}$.

This encoding method is used in the multinomial naive Bayes classifier (§3.3). On the downside, this method substantially increases the size of the data (five-fold in the quintile example) and, consequently, makes homomorphic computation more costly.

3.1.2 Expansion

An alternative data representation that is compatible with homomorphic computation is through the use of a data expansion technique. Graepel et al. (2013) proposed an approximation method where real values are first approximated by rationals (two integers factors) and then denominators are cleared by multiplying the entire dataset by a pre-specified integer and rounding the results to the nearest integer.

We suggest a more straightforward alternative: choose a desired level of accuracy, say ϕ, which represents the number of decimal places to be retained; then multiply the data by 10^{ϕ} and round to the nearest integer. This avoids the need for rational approximations and the requirement for a double approximation caused by the denominator-clearing step. More precisely, for a given precision $\phi \in \mathbb{N}$, a real value $z \in \mathbb{R}$ is approximated by $\dot{z} \equiv \lfloor 10^{\phi} z \rceil$,

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. The transformation can be illustrated as follows:

$$\begin{pmatrix} 55.2 & 1.71 \\ 72.5 & 1.92 \\ 61.7 & 1.63 \end{pmatrix} \mapsto \begin{pmatrix} 552 & 171 \\ 725 & 192 \\ 617 & 163 \end{pmatrix}. \tag{3.4}$$

This transformation adequately represents real values in an integer space, in the sense that smooth relative distances (e.g., Euclidean distance) are approximately preserved. For data sets of finite precision (the typical case in real applications), no loss of precision is necessary if ϕ is selected to be equal to the accuracy (i.e., number of decimal places) of the most accurate value in the data set. Otherwise, in the cases where transformations are required (e.g., logarithms), precision is under the user's control. Note that the parameter ϕ regulates the accuracy of the input (data), not that of the output (result). To the extent that the output accuracy depends on the input accuracy and also on the complexity of the algorithm, the choice of ϕ should take both these factors into consideration.

This method is used in the semiparametric naive Bayes classifier (§3.4) and in the regression algorithms in chapters 4 and 5. The downside is that some algorithms may have to be rescaled following the rescaling of the data, and the final result rescaled back to obtain the right answer. This is typically not problematic since the rescaling is independent of the data and, therefore, no sensitive information is leaked in the process.

3.2 The naive Bayes framework

We distinguish between *observables* and *observations*. Observables are random variable names denoting stochastic processes, while observations are data objects (vectors, matrices, etc.) containing observed instances of the corresponding observables. We use uppercase letters for observables, either scalar or multidimensional, Y; lowercase letters for scalar observations, y; and bold letters for non-scalar observations, lowercase for vectors, $\boldsymbol{y}$, and uppercase for matrices, $\boldsymbol{Y}$. For example, we write $\mathbb{P}(Y = c)$ to denote the probability that the observable variable Y takes on the value c. The empirical estimate of this quantity is $\sum_{i=1}^{N} \mathbb{I}\{y_i = c\}/N$, where $\boldsymbol{y} = \{y_i\}_{i \in \mathbb{N}_{1:N}}$ denotes a vector of observations corresponding to N realisations of the

observable process Y, and $\mathbb{I}\{\mathtt{z}\}$ denotes the indicator function, equal to 1 if condition $\mathtt{z}$ is true and zero otherwise.

Consider the binary classification problem of predicting a class label $Y \in \{0, 1\}$ from a set of predictors $X = \{X_j\}_{j \in \mathbb{N}_{1:P}} \in \mathcal{X}$. Thus, Y is a univariate stochastic process while X is multivariate. The NB classifier uses Bayes' theorem,

$$\mathbb{P}(Y|X) \propto \mathbb{P}(X|Y)\mathbb{P}(Y), \tag{3.5}$$

for prediction, coupled with an independence assumption,

$$\mathbb{P}(X|Y) = \prod_{j=1}^{P} \mathbb{P}(X_j|Y), \tag{3.6}$$

which embodies a compromise between accuracy and tractability, to obtain a conditional class probability. This allows NB to separately model the conditional distributions of predictor variables, $\{\mathbb{P}(X|Y = 1), \mathbb{P}(X|Y = 0)\}$, and then construct the prediction probability via Bayes' theorem:

$$\mathbb{P}(Y = 1|X) = \frac{\mathbb{P}(X|Y = 1)\mathbb{P}(Y = 1)}{\mathbb{P}(X|Y = 1)\mathbb{P}(Y = 1) + \mathbb{P}(X|Y = 0)\mathbb{P}(Y = 0)}. \tag{3.7}$$

The most popular forms for the distributions $\mathbb{P}(X_j|Y)$ are multinomial for categorical predictors, and Gaussian for continuous predictors. While Gaussian distributions lay outside of the possibilities of FHE (as they require evaluation of non-polynomial functions), it is possible to work with multinomial distributions directly in ciphertext space. However, this requires the encoding strategy given in (3.1) and illustrated in (3.3) to be used, which leads to a considerable expansion in data size and, consequently, in computation costs (Aslett, Esperança, and Holmes, 2015b).

In the following two sections we present two NB algorithms for homomorphic computation: a standard multinomial naive Bayes classifier (MNB; §3.3) and a tailored, semiparametric naive Bayes classifier (SNB; §3.4). Crucially, this novel method (SNB) scales to an arbitrary number of predictors at no additional multiplicative depth, making it well suited to encrypted computation.

3.3 Multinomial naive Bayes classifier

The multinomial naive Bayes (MNB) classifier estimates $\mathbb{P}(X_j|Y)$ and $\mathbb{P}(Y)$ using the corresponding empirical probabilities:

$$\hat{\mathbb{P}}(X_j = k|Y = c) = \frac{\sum_{i=1}^{N} \mathbb{I}\{x_{ij} = k\} \cdot \mathbb{I}\{y_i = c\}}{\sum_{i=1}^{N} \{y_i = c\}} \qquad \text{and} \qquad \hat{\mathbb{P}}(Y = c) = \frac{\sum_{i=1}^{N} \{y_i = c\}}{N} \tag{3.8}$$

where $\boldsymbol{X} = \{x_{ij}\}_{i \in \mathbb{N}_{1:N}, j \in \mathbb{N}_{1:P}}$ and $\boldsymbol{y} = \{y_i\}_{i \in \mathbb{N}_{1:N}}$ are, respectively, a matrix of N observations on P predictors and a vector of N responses. The "multinomial" qualifier should be understood to mean precisely that x_{ij} take values in the discrete set $\mathbb{N}_{1:M_j}$ so that $\mathcal{X} = \mathbb{N}_{1:M_1} \times \cdots \times \mathbb{N}_{1:M_P}$.

3.3.1 Unnormalised probabilities

Because it is not possible to perform homomorphic divisions, the algorithms will be written in terms of unnormalised probabilities, or counts:

$$\hat{\mathbb{P}}^{(u)}(X_j = k|Y = c) = \sum_{i=1}^{N} \mathbb{I}\{x_{ij} = k\} \cdot \mathbb{I}\{y_i = c\} \qquad \text{and} \qquad \hat{\mathbb{P}}^{(u)}(Y = c) = \sum_{i=1}^{N} \{y_i = c\}. \tag{3.9}$$

Estimation, therefore, requires the computation of $\hat{\mathbb{P}}^{(u)}(X_j|Y)$ and $\hat{\mathbb{P}}^{(u)}(Y)$ for all possible values of Y and X_j, and for all $j \in \mathbb{N}_{1:P}$. This corresponds to a $2 \times M_j$ table of unnormalised conditional probabilities for each predictor; plus two unnormalised class probabilities. In total, this makes $2 + 2 \cdot \sum_{j=1}^{P} M_j$ parameters.

3.3.2 Encrypted conditional counts

A further computational problem arises from the fact that $\hat{\mathbb{P}}^{(u)}(X_j|Y)$ cannot be computed directly. Since the values x_{ij} are encrypted, this makes it impossible to count, say, the number of 2s in $\boldsymbol{X}_{\bullet j}$ directly. An indirect route follows from one of the data encoding methods proposed in §3.1.1, namely the one in (3.1). Let $\tilde{\boldsymbol{X}}$ denote the binary-expanded version of $\boldsymbol{X}$; and $\tilde{x}_{ijk}$ denote the kth bin of $\tilde{\boldsymbol{x}}_{ij} = (\tilde{x}_{ij1}, \ldots, \tilde{x}_{ijM_j})$. Then, the conditional counts can

be computed as

$$\hat{\mathbb{P}}^{(u)}(X_j = k|Y = 1) = \sum_{i=1}^{N} \tilde{x}_{ijk} y_i \qquad \text{and} \qquad \hat{\mathbb{P}}^{(u)}(X_j = k|Y = 0) = \sum_{i=1}^{N} \tilde{x}_{ijk}(1 - y_i) \quad (3.10)$$

and the class counts as

$$\hat{\mathbb{P}}^{(u)}(Y = 1) = \sum_{i=1}^{N} y_i \qquad \text{and} \qquad \hat{\mathbb{P}}^{(u)}(Y = 0) = N - \hat{\mathbb{P}}^{(u)}(Y = 1). \quad (3.11)$$

As standard practice, implementations should use Laplace-smoothed counts to avoid the problems arising from having zero counts for some values of k.

Homomorphic computation of the MNB classifier requires the use of binary-expanded data ($\tilde{\boldsymbol{X}}$) as opposed to integer data ($\boldsymbol{X}$). This has a direct impact on the computational cost as it leads to a larger number of homomorphic operations (recall (3.10) and (3.13)) and requires more memory. Importantly, these algorithms have low depth: the MMD of $\hat{\mathbb{P}}^{(u)}(Y)$ is 0, while that of $\hat{\mathbb{P}}^{(u)}(X_j|Y)$ is 1.

3.3.3 Prediction

Classification of a new observation $\boldsymbol{x}_\star = \{x_{\star j}\}_{j \in \mathbb{N}_{1:P}}$ requires the computation of the factors

$$\begin{aligned} F_c &= \mathbb{P}(Y = c) \prod_{j=1}^{P} \mathbb{P}(X_j = x_{\star j}|y = c) \\ &= N^{-1} \left(\hat{\mathbb{P}}^{(u)}(Y = c)\right)^{-(P-1)} \prod_{j=1}^{P} \hat{\mathbb{P}}^{(u)}(X_j = x_{\star j}|Y = c) \end{aligned} \quad (3.12)$$

for $c \in \{0, 1\}$. If $\boldsymbol{x}_\star$ is known only in encrypted form, then $\hat{\mathbb{P}}^{(u)}(X_j = x_{\star j}|Y = c)$ must be computed indirectly as

$$\hat{\mathbb{P}}^{(u)}(X_j = x_{\star j}|Y = c) = \sum_{k=1}^{M_j} \tilde{x}_{\star jk} \cdot \hat{\mathbb{P}}^{(u)}(X_j = k|Y = c) \quad (3.13)$$

where $\tilde{\boldsymbol{x}}_{\star j} = \{\tilde{x}_{\star jk}\}_{k \in \mathbb{N}_{1:M_j}}$ is the binary expanded version of $\boldsymbol{x}_{\star j}$. The MMD is equal to 2 and the predicted probability of $\boldsymbol{x}_\star$ belonging to class $Y = 1$ is then $\mathbb{P}(Y = 1|X = \boldsymbol{x}_\star) = F_1/(F_0 + F_1)$, a final step which must be computed offline.

3.3.4 Theoretical parameter requirements for the FV cryptosystem

For MNB, the elements to be returned are $\hat{\mathbb{P}}^{(u)}(Y=c)$ and $\hat{\mathbb{P}}^{(u)}(X_j = x_{\star j}|Y=c)$, for $j \in \mathbb{N}_{1:P}$ and $c \in \{0,1\}$, with the former having MMD equal to 0 and the later equal to 2. It is possible to reduce the number of elements to be returned to 4 by computing $(\hat{\mathbb{P}}^{(u)}(Y=c))^{P-1}$ and $\prod_{j=1}^{P} \hat{\mathbb{P}}^{(u)}(X_j = x_{\star j}|Y=c)$ homomorphically, but this comes at the cost of an increase in the MMD to $P-1$ and $3P-1$, respectively.

3.4 Semiparametric naive Bayes classifier

The NB classifier solves the classification task using a generative approach, i.e., by modelling the distributions of the predictors (Ng and Jordan, 2002). However, distributions such as the Gaussian cannot be directly implemented under FHE as they typically involve division and exponentiation. Moreover, other distributional forms such as the MNB classifier (§3.3) require data to be discretised and encoded using the transformation (3.1), which expands the size of the data considerably and, as a consequence, increases the computational costs.

Here, we show that it is possible to model the decision boundary between the two response classes more explicitly — without a parametric model for the distributions of the predictors — while still remaining in the NB framework. As will become clear, this corresponds to a discriminative approach to classification where the decision boundary of the conditional class probabilities is modelled semiparametrically.

To begin, note that the expression for the log-odds prediction from NB can be rearranged:

$$\begin{aligned}
\log\left(\frac{\mathbb{P}(Y=1|X)}{\mathbb{P}(Y=0|X)}\right) &= \log\left(\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)}\prod_{j=1}^{P}\frac{\mathbb{P}(X_j|Y=1)}{\mathbb{P}(X_j|Y=0)}\right) \\
&= \log\left(\left[\frac{\mathbb{P}(Y=0)}{\mathbb{P}(Y=1)}\right]^{(P-1)}\prod_{j=1}^{P}\left[\frac{\mathbb{P}(X_j|Y=1)}{\mathbb{P}(X_j|Y=0)}\cdot\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)}\right]\right) \\
&= \log\left(\left[\frac{\mathbb{P}(Y=0)}{\mathbb{P}(Y=1)}\right]^{(P-1)}\prod_{j=1}^{P}\frac{\mathbb{P}(Y=1|X_j)}{\mathbb{P}(Y=0|X_j)}\right) \\
&= (P-1)\log\left(\frac{\mathbb{P}(Y=0)}{\mathbb{P}(Y=1)}\right) + \sum_{j=1}^{P}\log\left(\frac{\mathbb{P}(Y=1|X_j)}{\mathbb{P}(Y=0|X_j)}\right).
\end{aligned} \tag{3.14}$$

Using this identity, we now propose to model the decision boundary $\mathbb{P}(Y|X_j)$ directly using the linear logistic form, rather than parameterise $\mathbb{P}(X_j|Y)$ via a distribution function. That is, we assume

$$\log\left(\frac{\mathbb{P}(Y=1|X_j=x_j)}{\mathbb{P}(Y=0|X_j=x_j)}\right) = f(x_j;\alpha_j,\beta_j) \qquad \text{and} \qquad \frac{\mathbb{P}(Y=0)}{\mathbb{P}(Y=1)} = \theta \tag{3.15}$$

where, in this work, $f(x_j;\alpha_j,\beta_j)$ is taken to be a linear predictor of the form $\alpha_j + \beta_j x_j$. The independence structure means that each term can be optimised independently by way of an approximation to logistic regression, amenable to homomorphic computation (presented ahead; §3.4.1). Optimisation of θ is done independently of $\{\alpha_j,\beta_j\}$:

$$\hat{\theta} = \frac{\sum_{i=1}^{N} y_i}{N - \sum_{i=1}^{N} y_i}. \tag{3.16}$$

The estimated log-odds are then

$$\begin{aligned}\hat{\psi}(\boldsymbol{x};\hat{\theta},\hat{\alpha},\hat{\beta}) &= \log\left(\frac{\mathbb{P}(Y=1|X=\boldsymbol{x};\hat{\theta},\hat{\alpha},\hat{\beta})}{\mathbb{P}(Y=0|X=\boldsymbol{x};\hat{\theta},\hat{\alpha},\hat{\beta})}\right) \\ &= (P-1)\log(\hat{\theta}) + \sum_{j=1}^{P} f(x_j;\hat{\alpha}_j,\hat{\beta}_j)\end{aligned} \tag{3.17}$$

or, equivalently, in terms of conditional class probabilities

$$\mathbb{P}(Y=1|X=\boldsymbol{x};\hat{\theta},\hat{\alpha},\hat{\beta}) = \frac{1}{1+\exp\{-\hat{\psi}(\boldsymbol{x};\hat{\theta},\hat{\alpha},\hat{\beta})\}}. \tag{3.18}$$

Hence equation (3.18) can be computed after decryption from the factors (numerator and denominator) that comprise $\{\hat{\theta},\hat{\alpha},\hat{\beta}\}$ to form the conditional class predictions.

This defines a semiparametric naive Bayes (SNB) classification model that assumes a logistic form for the decision boundary in each predictor variable, and where each logistic regression involves at most two parameters. As we will show in the next section, in this setting there is a suitable approximation to the maximum likelihood estimates which are traditionally computed using the iteratively reweighted least squares algorithm, so that they are computable homomorphically.

3.4.1 Approximate logistic regression

The SNB algorithm proposed in the previous section requires a homomorphic implementation of simple logistic regression involving an intercept and a single slope parameter. In this section, an approximation to logistic regression based on the first iteration of the iteratively reweighted least squares (IRLS) algorithm is proposed and some of its theoretical properties are analysed. Apart from its use in SNB, the approximation to logistic regression described in this section also stands on its own as a homomorphic classification method.

3.4.1.1 Iteratively reweighted least squares

Optimisation in logistic regression is typically achieved via IRLS based on Newton iterations. Starting from an initial guess $\boldsymbol{\beta}^{[0]}$, the first step involves updating the auxiliary variables:

$$\eta_i^{[0]} = \boldsymbol{X}_{i\bullet}\boldsymbol{\beta}^{[0]} \tag{3.19a}$$

$$\mu_i^{[0]} = \frac{e^{\eta_i^{[0]}}}{1 + e^{\eta_i^{[0]}}} \tag{3.19b}$$

$$w_{ii}^{[0]} = \mu_i^{[0]}(1 - \mu_i^{[0]}) \tag{3.19c}$$

$$z_i^{[0]} = \eta_i^{[0]} + (w_{ii}^{[0]})^{-1}(y_i - \mu_i^{[0]}) \tag{3.19d}$$

where $\boldsymbol{X}_{i\bullet}$ denotes the ith row of $\boldsymbol{X}$. By starting with an initial value $\boldsymbol{\beta}^{[0]} = (0, \ldots, 0)^T$ the initialisation step of the algorithm is simplified: for all $i \in \mathbb{N}_{1:N}$ we have $\eta_i^{[0]} = 0$, $\mu_i^{[0]} = 1/2$, $w_{i,i}^{[0]} = 1/4$ and $z_i^{[0]} = 4y_i - 2$. In the second step, the parameter estimates of $\boldsymbol{\beta}$ are updated using generalised least squares:

$$\boldsymbol{\beta}^{[1]} = \left(\boldsymbol{X}^T\boldsymbol{W}^{[0]}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{W}^{[0]}\boldsymbol{z}^{[0]} \tag{3.20}$$

and these two steps are repeated until convergence is achieved. In what follows, only the first-iteration update of $\boldsymbol{\beta}$ is considered, because it provides an approximation to logistic regression and, most importantly, one which is computationally feasible under homomorphic encryption; this will be termed the *one-step approximation*. Note that implementation of the

full IRLS algorithm is infeasible under FHE because the weights w_{ii} cannot be updated as this requires the evaluation of non-polynomial functions.

The particular form of these (*paired*) one-step approximating equations for an intercept and a single slope parameter is shown in the following section. We present an additional simplification (called *unpaired*) which improves encrypted computational characteristics further at the cost of greater approximation. We refer to the two resulting algorithms as `SNB-paired` and `SNB-unpaired`, respectively

3.4.1.2 Paired optimisation

In the SNB model the conditional log-odds are optimised for each variable separately, following the independence assumption. In this case, the design matrix $\boldsymbol{X}$ contains a column of ones for the intercept plus a single predictor variable, and the one-step approximation implies:

$$\{\hat{\alpha}_j, \hat{\beta}_j\} = \{a_j/d_j, b_j/d_j\}\,, \quad \forall j \in \mathbb{N}_{1:P} \tag{3.21}$$

where

$$a_j = \left(\sum_{i=1}^{N} x_{ij}^2\right)\left(\sum_{i=1}^{N} z_i^{[0]}\right) - \left(\sum_{i=1}^{N} x_{ij}\right)\left(\sum_{i=1}^{N} x_{ij} z_i^{[0]}\right) \tag{3.22a}$$

$$b_j = N\sum_{i=1}^{N} x_{ij} z_i^{[0]} - \left(\sum_{i=1}^{N} x_{ij}\right)\left(\sum_{i=1}^{N} z_i^{[0]}\right) \tag{3.22b}$$

$$d_j = N\sum_{i=1}^{N} x_{ij}^2 - \left(\sum_{i=1}^{N} x_{ij}\right)^2 \tag{3.22c}$$

To use the one-step approximation in combination with the SNB classifier, the quantity $\{\hat{\alpha}_j + \hat{\beta}_j x_{\star j}\}$ — required for the classification of a new observation $\boldsymbol{x}_\star = \{x_{\star j}\}_{j\in\mathbb{N}_{1:P}}$ — can then be estimated as e_j/d_j, where $e_j = a_j + b_j x_{\star j}$. This corresponds to a standard, *paired* optimisation strategy, that is, a strategy that optimises intercept and slope jointly, using an approximation targeting

$$\underset{\alpha_j,\beta_j}{\arg\sup}\, f(x_j; \alpha_j, \beta_j), \quad \forall j \in \mathbb{N}_{1:P}. \tag{3.23}$$

An alternative to this approach would be to optimise intercept and slope independently (i.e., in an *unpaired* fashion). We will see that this is computationally appealing due to the simplified estimating equations when using the same one-step approximation.

3.4.1.3 Unpaired optimisation

The *unpaired* optimisation approach targets

$$\arg\sup_{\alpha_j} f(x_j;\alpha_j,\beta_j=0) \quad \text{and} \quad \arg\sup_{\beta_j} f(x_j;\alpha_j=0,\beta_j), \quad \forall j\in\mathbb{N}_{1:P} \tag{3.24}$$

in which case all α_j take the same value and is therefore equivalent to estimating all β_j independently and including a global intercept, $P\alpha_j$ for some j.

In this case (or in the absence of an intercept term) the estimating equations for α_j and β_j are simpler. To distinguish between unpaired and paired estimates we denote the unpaired by $\tilde{\alpha}_j$ and $\tilde{\beta}_j$, respectively:

$$\tilde{\alpha}_j = \tilde{\alpha} = \frac{1}{N}\sum_{i=1}^{N} z_i^{[0]}, \quad \forall j\in\mathbb{N}_{1:P} \tag{3.25a}$$

$$\tilde{\beta}_j = \left(\sum_{i=1}^{N} x_{ij}^2\right)^{-1} \sum_{i=1}^{N} x_{ij} z_i^{[0]} \tag{3.25b}$$

Note that this unpaired formulation also arises in the case of centred predictors, $\mathbb{E}[X_j]=0$, so that the unpaired approach is exactly equivalent to the paired approach and introduces no additional approximation if it is possible for the data to be centred prior to encryption. This is trivially achievable for ordinal quintile data by representing each quintile by $\{-2,-1,0,1,2\}$, rather than $\{1,2,3,4,5\}$. Consequently, where it is possible to centre the data, it should be done to take advantage of these computational benefits that incur no approximation cost.

Furthermore, when the data are discretised into quintiles (§3.1), this can be rewritten as follows. Define the auxiliary variables:

$$n_{jk}^{[1]} = \sum_{i:y_i=1} \tilde{x}_{ijk} = \sum_{i:\tilde{x}_{ijk}=1} y_i, \qquad n_{jk} = \sum_{i=1}^{N} \tilde{x}_{ijk}, \qquad n_{jk}^{[0]} = n_{jk} - n_{jk}^{[1]} \tag{3.26}$$

In words, $n_{jk}^{[c]}$ counts the number of observations in the kth bin of the jth predictor — or, equivalently, the number of elements of $X_{\bullet j}$ (the jth column of X) which are equal to k — and for which the corresponding response is equal to c, for $c \in \{0, 1\}$. In this case, (3.25b) becomes

$$\tilde{\beta}_j = 2 \left(\sum_{k=1}^{M_j} k^2 \sum_{i=1}^{N} \tilde{x}_{ijk} \right)^{-1} \sum_{k=1}^{M_j} k \left(n_{jk}^{[1]} - n_{jk}^{[0]} \right) \tag{3.27}$$

so that for a binary predictor $x_{ij} \in \{0, 1\}$ we find

$$\tilde{\beta}_j = \frac{2}{n} \left(n_j^{[1]} - n_j^{[0]} \right). \tag{3.28}$$

where $n = \sum_{i=1}^{N} x_{ij}$. This simple expression aids in the derivation of some theoretical results regarding the bias of the estimator.

We note that the above derivation implies a discretisation into ordinal variables (e.g., applicable in the case where the original variables are real-valued). For categorical variables, or a mix of the two, a similar approach can be used to derive analogous results.

3.4.2 Shrinkage

Let β denote the true parameter and $\tilde{\beta}$ denote the one-step IRLS estimate from Equation (3.28). The coefficient error is then given by

$$\mathbb{E}[\tilde{\beta} - \beta] = \mathbb{E}\left[\frac{2\left(n^{[1]} - n^{[0]}\right)}{n} - \beta \right] = -2 + 4\left(\frac{e^{\beta}}{1 + e^{\beta}} \right) - \beta. \tag{3.29}$$

The one-step "early stopping" shrinks or regularises the estimate towards the origin. The shrinkage, as a function of the true parameter β, is shown in Figure 3.1; it is negligible when β is small, but increases linearly in magnitude with β for β outside the interval $(-2, 2)$. The reason for this is clear from the formula in (3.28): the range of $\tilde{\beta}$, the one-step estimate, being $(-2, 2)$.

In particular, note that this shrinkage is a highly desirable property in light of the independence assumptions made in SNB. Indeed, the one-step procedure empirically outperforms the

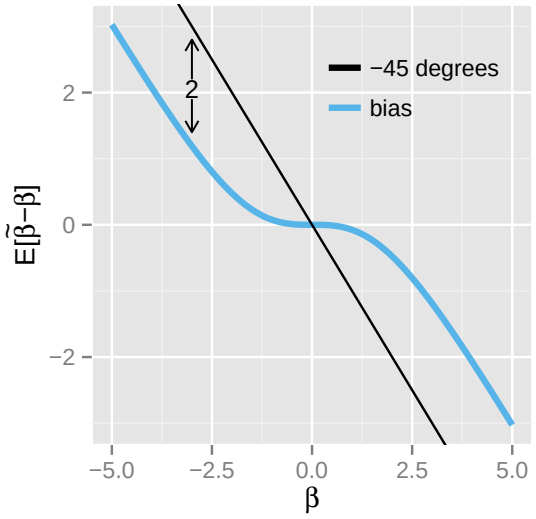


Figure 3.1: Shrinkage in $\tilde{\beta}$. Because the estimator $\tilde{\beta}$ is bounded between -2 and 2, the shrunk value is always equal to 2 for absolute values of β greater than ≈ 2.5.

full convergence IRLS when predictors are highly correlated and does not significantly underperform otherwise. Therefore, the one-step method is not only a computational necessity for homomorphic computation, but offers potential improvements in performance against the standard algorithm.

3.4.3 Generalisation error

Define $p = \mathbb{P}[y = 1|x = 1, \beta]$ and $\tilde{p} = \mathbb{P}[y = 1|x = 1, \tilde{\beta}]$, where β is the true parameter and $\tilde{\beta}$ is as in (3.28). Then, the generalisation error for $x = 1$ can be written as

$$\mathbb{E}[\tilde{p} - p] = \mathbb{E}\left[\frac{e^{\tilde{\beta}}}{1 + e^{\tilde{\beta}}} - p\right] \tag{3.30}$$

and approximated at $x = 0$ by the polynomial

$$\mathbb{E}[\tilde{p} - p] \approx \mathbb{E}\left[\frac{1}{2} + \frac{1}{4} \cdot \tilde{\beta} - \frac{1}{48} \cdot \tilde{\beta}^3\right] - p. \tag{3.31}$$

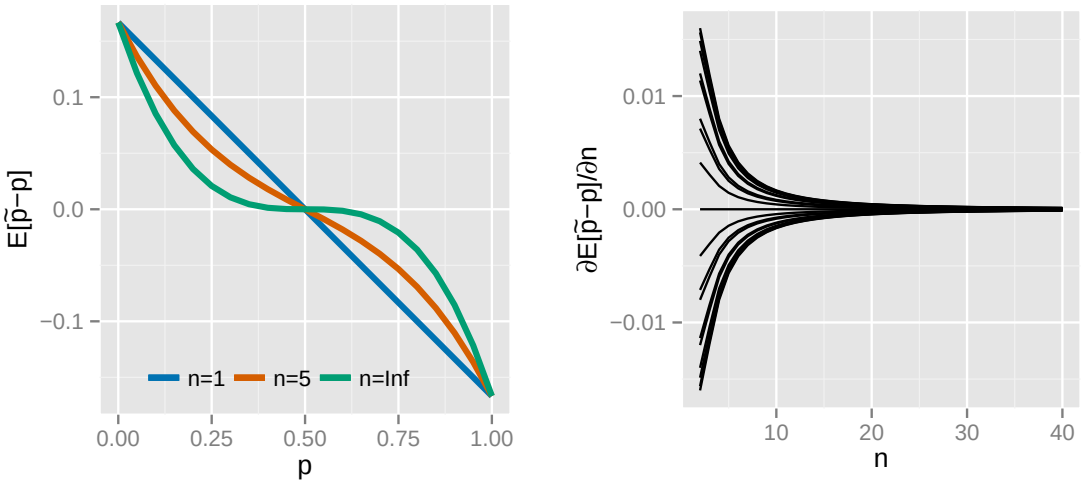


Figure 3.2: Generalisation error: **[left]** for all p, the generalisation error shrinks monotonically with increased n and converges to an asymptotic curve — in green; **[right]** the rate at which the generalisation error curves approach the asymptotic curve is decreasing with n for all values of p (several shown here) and stabilises at around $n = 20$.

Using the moment generating function for the Binomial distribution one can compute the higher-order expectations and thus arrive at a formula which depends only on n and p,

$$\begin{aligned}\mathbb{E}[\tilde{p} - p] \approx -\frac{1}{6n^2}\big(&8n^2p^3 - 12n^2p^2 + 6n^2p - n^2 - 24np^3 \\ &+36np^2 - 12np + 16p^3 - 24p^2 + 8p\big)\,.\end{aligned} \tag{3.32}$$

The expression is dominated by p in the sense that even for very small values of n, the generalisation error is close (according to our approximation) to the one obtained asymptotically,

$$\lim_{n\to\infty} \mathbb{E}[\tilde{p} - p] = -\frac{1}{6}(2p-1)^3. \tag{3.33}$$

Figure 3.2 (left) shows how the value of n affects the generalisation error; and Figure 3.2 (right) shows the speed at which the generalisation error converges to the one given by the (approximate) asymptotic curve, for several values of p.

The generalisation error results highlight that while the estimates of the true class conditional probabilities may be unstable, the classification error rate achieved by SNB may be low, as the classifier has to only get the prediction on the correct side of the decision boundary.

3.4.4 Theoretical parameter requirements for the `FV` cryptosystem

In order to fit `SNB-paired`, the MMD of the coefficients $\{e_j, d_j\}_{j \in \mathbb{N}_{1:P}}$ is 4 and 3, respectively, while the MMD of the empirical class counts is 3. For `SNB-unpaired` the MMD of the coefficients is equal to 2 and 1, respectively.

With respect to the growth in message coefficient size for the `FV` scheme (see §2.4.3), the biggest growth comes from computing d_j in (3.22c) in the case of `SNB-paired`. Initially, the polynomial representation of all x_{ij} will have maximum coefficient value of 1. By the multinomial theorem, the maximum coefficient size (MCS) of the polynomial representation of x_{ij}^2 will be 2. Thus, the MCS for the first term is $2N$. In contrast, the maximum coefficient of $\sum_i x_{ij}$ will be N, and so, by the multinomial theorem, the MCS of the polynomial representation of $(\sum_i x_{ij})^2$ will be $2N^2$. Therefore, to ensure correct decryption, the message space must support coefficient values up to size $2N^2$. For `SNB-unpaired` the maximum coefficient requirements come from (3.25b) and are also significantly relaxed, with the MCS being $2N$.

3.5 Results

In this section we apply the encrypted machine learning methods presented earlier to a number of benchmark learning tasks. We also include related methods which are used as a benchmark for performance comparisons. The SNB fitting and prediction algorithms are implemented in the open source `R` package `EncryptedStats` (described in appendix A) and can be run on unencrypted as well as encrypted data.

3.5.1 Classification performance

Data description. We tested the methods on 20 data sets of varying type and dimensions from the UCI machine learning data repository (Lichman, 2013). For the purposes of achieving many test replicates, the results in this subsection were generated from unencrypted runs, with checks to ensure that the unencrypted and encrypted versions give the same results. The code paths in the `EncryptedStats` package for unencrypted and encrypted values are identical.

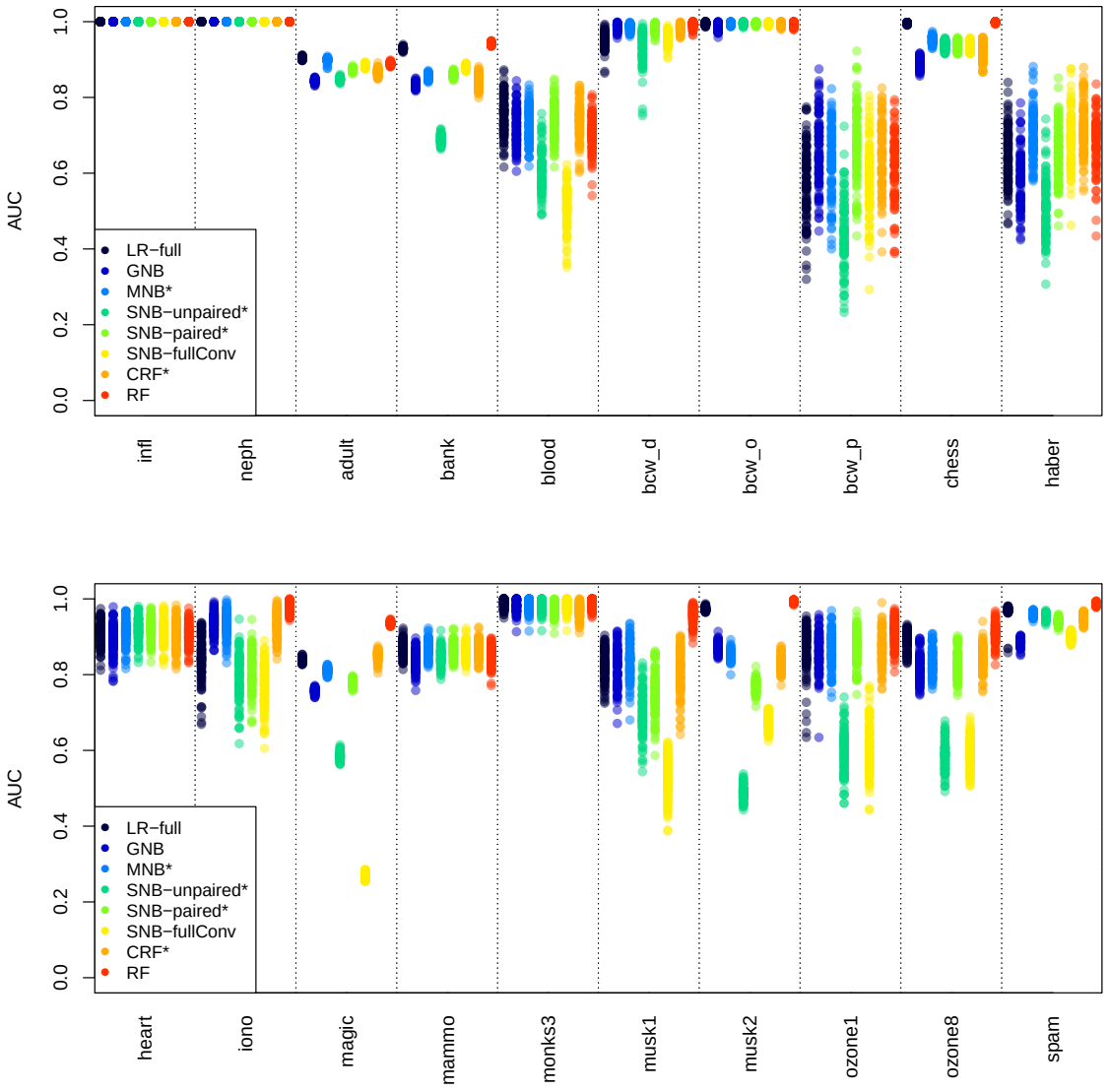


Figure 3.3: Performance of various methods. For each model and dataset, the AUC for 100 stratified randomisations of the training and testing sets. An asterisk indicates the method can be computed encrypted and the remaining methods serve as benchmarks.

Figure 3.3 shows the comparison of these novel methods with each other as well as with their traditional counterparts. The traditional methods included are: full logistic regression (`LR-full`), Gaussian naive Bayes (`GNB`) and random forests (`RF`), plus the proposed semi-parametric Naive Bayes with IRLS taken to convergence (`SNB-fullConv`) — none of which can be computed within the constraints of homomorphic encryption. The methods proposed in this paper and which can be computed homomorphically are: paired (`SNB-paired`) and unpaired (`SNB-unpaired`) semiparametric naive Bayes, and multinomial naive Bayes (`MNB`). A additional method described in Aslett, Esperança, and Holmes (2015b), completely random

forests (`CRF`), is also compared here. The CRFs are all 100 trees grown 3 levels deep, including stochastic fraction estimate ($M = 8$).

The performance measure used is the area under the ROC curve (AUC, ranging from 0 to 1). For each model and dataset, the algorithms were run with the same 100 stratified randomisations of the training and testing sets (split in the proportion 80%/20%, respectively), so that each point on the graph represents the AUC for one train/test split and one method.

The first two data sets (`infl` and `neph`) constitute very easy classification problems and the new techniques match the traditional techniques perfectly in this setting, keeping almost uniformly good pace in the other rather easy data sets as well (`bcw_d`, `bcw_o`, `monks3`), the unpaired SNB being the exception on `bcw_d`.

Unsurprisingly, the traditional random forest tends to perform best in the more challenging data sets (Fernández-Delgado et al., 2014), though only in 4 of the data sets it clearly outperforms all the other methods by the AUC metric. Indeed, in the most challenging data sets (`blood`, `bcw_p` and `haber`) the new methods proposed in this work exhibit slightly better average performance than their counterparts.

The results of the unpaired SNB were obtained without centring (which would be equivalent to paired) and affirms the observation of the previous section that centring or paired computation is always to be preferred where available.

The SNB method with IRLS run to convergence is not presented in the figure: as alluded to in the previous section, the natural shrinkage of the one-step estimator meant it performed equally well in most situations and in the `chess` and `musk1` cases the average ratio of full convergence over one-step AUCs was 0.76 and 0.67 respectively.

3.5.2 Computational costs

The runtimes shown here for the `SNB-paired` algorithm were measured on a 36-core server (see §A.3 for hardware and software details). The encrypted methods presented here scale linearly in the dataset size, so performance numbers can be provided per number of observations and per number of predictors.

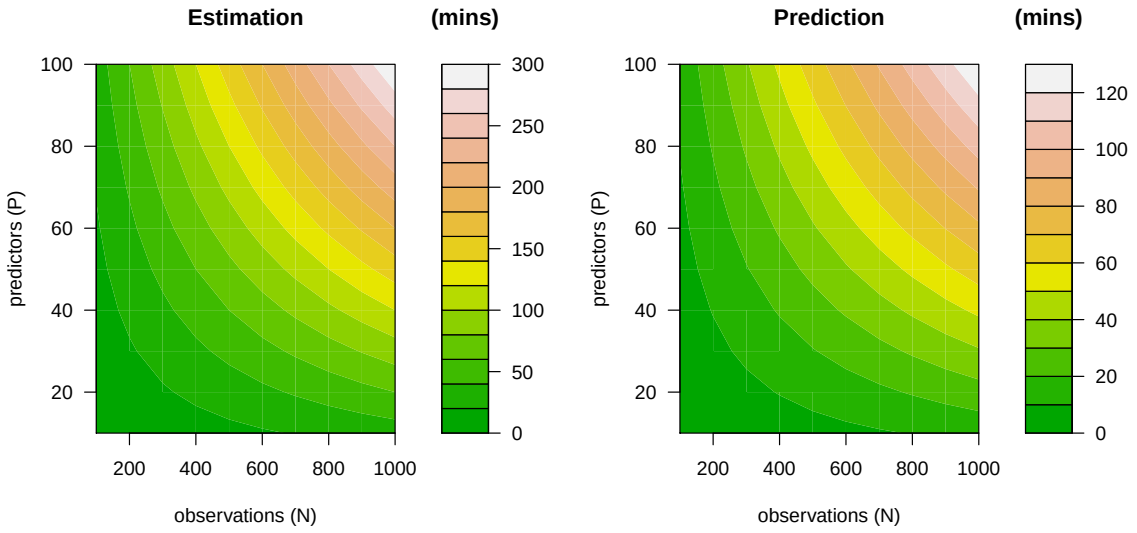


Figure 3.4: Projected runtimes (in minutes) for the `SNB-paired` classifier with encrypted data: **[left]** estimation step; **[right]** prediction step.

Per 100 observations per predictor, the estimation step takes approximately 18 seconds while the prediction step takes approximately 7.8 seconds. Figure 3.4 shows projected runtimes with varying problem size, N and P. The values are projections based on two facts: (i) the cryptographic parameters for this problem are independent of P and only theoretically dependent on N via the MCS, while in practice this dependence is weak, see §3.4.4; (ii) for a fixed set of cryptographic parameters, the costs of homomorphic operations are proportional to the size of the data, see §2.4.4.

Memory requirements are proportional to the dataset size at a rate of approximately 154KB per ciphertext. To this should be added the number of auxiliary variables that need to be stored during computation, e.g., regression coefficients. Notice that the algorithm can be easily scaled to large datasets because all computation can be done in batches of small numbers of predictors if memory bottlenecks exist.

3.6 Discussion

The methods presented in this chapter constitute a significant step forward in achieving practical, privacy-preserving statistical analysis in the cloud without any intermediate communication between server and client. Contrarily to other approaches in the literature, we are

capable of performing both estimation and prediction in the encrypted domain (Bos et al., 2014; Bost et al., 2014). In particular, estimation and prediction using the SNB classifier can be performed directly on encrypted data by computing a set of numerators and denominators which the secret key holder decrypts and assembles into the final result, as standard.

Differentially-private outputs

Because a number of statistics need to first be decrypted in order to be assembled into the final result (e.g., numerator and denominator of conditional class probabilities in the case of the MNB classifier), we may be concerned that these statistics could leak information about the original data; for example, if the secret key holder is not supposed to have access to the unencrypted data but only to the final result.

A simple solution would be to involve a third party, who has legitimate access to the data (i.e., also possesses the secret key), and who assembles all intermediate statistics into the final result and communicates only the later to the party who is interested in the final result. Alternatively, it is possible to obtain stronger privacy guarantees by applying differential privacy (DP) to these intermediate statistics, without requiring an additional party to be involved in the computation. DP has been applied to standard logistic regression (Chaudhuri and Monteleoni, 2009; Ji et al., 2014; Zhang et al., 2012). However, in the case of the proposed SNB classifier the details are less clear, since the method is a composition of several logistic regressions and, therefore, the most efficient DP mechanism may differ. In the case of the MNB classifier the task seems to easier, as the Laplace smoothed probabilities are closely linked to the Laplace mechanism for DP (Dwork and Roth, 2014; Vaidya et al., 2013).

Homomorphic-tailored algorithms for non-homomorphic computation

Because the most complex operations involved in SNB are addition and multiplication, the unencrypted version of this algorithm scales well to massive data sets. Contributing to this, is the fact that parallel computation is easy to implement: (i) due to the NB independence assumption, which allows computation to be broken down vertically (predictor-wise); (ii)

due to the simple vector algebra involved, which allows computation to be broken down horizontally (observation-wise).

3.7 Summary

In this chapter we presented algorithms for privacy-preserving classification in the context of fully homomorphic encryption. FHE schemes open up the prospect of outsourced computation for privacy-preserving machine learning applications. However, practical constraints of existing FHE schemes demand tailored approaches. We provided a scalable, semiparametric naive Bayes classification algorithm tailored for homomorphic computation. Additionally, a more trivial classifier for multinomial data was also provided which exploits a particular encoding strategy that allows for a rudimentary type of encrypted conditional counting.

The main methodological contribution of this chapter is the SNB classifier in its paired form. It enjoys several desirable properties: (i) it allows for both secure estimation and secure prediction; (ii) it has a low multiplicative depth, 3 for only estimation and 4 for estimation and prediction, which is crucially independent of the problem size and so makes it possible to carry out classification on large scale datasets; (iii) due to the naive Bayes independence assumption, both estimation and prediction can be easily implemented in parallel, which allows the method to scale to large-P datasets by processing predictors independently, while large-N datasets are also trivially parallelisable by processing horizontally partitioned blocks of data and then aggregating the results; (iv) it performs very well empirically, where it is competitive with standard logistic regression.

As a result of its combined efficiency and performance, the SNB algorithm stands as a good candidate for regular (non-homomorphic) classification. Because it uses only very simple operations — addition and multiplication — it can be especially useful in high-dimensional settings.

Some classification algorithms have been adapted for homomorphic computation (Bos et al., 2014; Bost et al., 2014; Dowlin et al., 2015), but few provide both estimation and prediction in the encrypted domain (Graepel et al., 2013). We have shown that it is indeed possible

to construct alternative classifiers that include both of these capabilities, and that are well suited for homomorphic computation in the cloud.

Chapter 4

Accelerated least squares regression

In this chapter we propose an accelerated least squares algorithm for privacy-preserving linear regression. Our approach enjoys several notable properties: it scales linearly with the size of the problem (number of predictors); admits the option of L_2 (ridge) regularisation shrinkage of regression coefficients; allows both estimation and prediction to be performed in the encrypted domain; and can avoid bootstrapping for moderately large problems.

Related work. The topic of linear regression in the context of FHE has not been covered systematically in the literature thus far. Hall et al. (2011) propose protocols for regression analysis based on PHE and SMPC. The Paillier cryptosystem supports homomorphic addition and multiplication by a constant, and so extensive rounds of communication between several parties are required to compute additive shares of key quantities in the traditional ordinary least squares estimator, namely $\{\boldsymbol{X}^T\boldsymbol{X},\ \boldsymbol{X}^T\boldsymbol{y},\ (\boldsymbol{X}^T\boldsymbol{X})^{-1}\}$, and finally shares of the target quantity $\hat{\boldsymbol{\beta}}_{\text{ols}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. The communication overhead in this approach is substantial: it takes two days to complete a problem of size $N = 51,016$ and $P = 23$. Wu and Haven (2012) were the first to tackle the problem of linear regression in the context of fully homomorphic encryption. The authors propose the use of Cramer's rule for matrix inversion but unfortunately this approach calls for the computation of the determinant of $\boldsymbol{X}^T\boldsymbol{X}$, which quickly becomes intractable for even low dimensional problems (e.g., $P > 4$). Specifically, the multiplicative depth is unbounded with growing dimension and, consequently, bootstrapping seems unavoidable. Graepel et al. (2013) mention that regression can be done by gradient descent, but do not implement the method or give further details.

Roadmap. We start by giving details on the data representation and encoding necessary for encrypted regression analysis (§4.1) as well as reviewing the standard and penalised linear regression model (§4.2); we then describe several iterative solutions, compare their computational properties and show how to introduce regularisation (§4.3); next, we examine several acceleration strategies of fundamental importance (§4.4); after this, we present a set of results, including simulations and applications to real-world datasets (§4.5); and finally, discuss (§4.6) and summarise (§4.7) our findings.

4.1 Data representation and encoding

Because homomorphic encryption schemes can only natively encrypt representations of integers, we must encode non-integer data with this restriction in mind. We use the transformation $\dot{z} \equiv \lfloor 10^{\phi} z \rceil \in \mathbb{Z}$, for $z \in \mathbb{R}$ and $\phi \in \mathbb{N}$. Here ϕ represents the desired level of accuracy or, more precisely, the number of decimal places to be retained. By construction, $\dot{z} \approx 10^{\phi} z$, so that smooth relative distances between elements encoded in this way are approximately preserved. The encoding is simple, gives the user complete control over input accuracy through ϕ and can accommodate both discrete (integer-valued) data and an approximation to continuous (real-valued) data. See §3.1.2 for more details. Throughout, we assume that covariates are standardised and responses centred, which can be done as pre-processing before encryption.

4.2 The linear regression model

The standard linear regression model assumes

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}_N, \sigma^2 \boldsymbol{I}_{NN}) \tag{4.1}$$

where $\boldsymbol{y}$ is a response vector of length N; $\boldsymbol{X}$ is a design matrix of size $N \times P$; $\boldsymbol{\beta}$ is a parameter vector of length P; and $\boldsymbol{\varepsilon}$ is a vector of length N of independent and normally distributed errors with zero mean and constant variance σ^2. N and P denote, respectively, the numbers

of observations and predictors; $\mathbf{0}_N$ is a vector of zeros of length N; and $\boldsymbol{I}_{NN}$ is the identity matrix of size $N \times N$.

4.2.1 Ordinary least squares

Let $||\boldsymbol{z}||_q = (\sum_k |z_k|^q)^{1/q}$ denote the L_q-norm of a vector $\boldsymbol{z}$, and let $||\boldsymbol{z}||_\infty = \max_k\{z_k\}$ denote the infinity norm. Ordinary least squares (OLS) solves the following optimisation problem:

$$\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 \tag{4.2}$$

that is, it minimises the residual sum of squares $\boldsymbol{e}^T\boldsymbol{e} = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$ to obtain parameter estimates $\hat{\boldsymbol{\beta}}_{\text{ols}}$ of $\boldsymbol{\beta}$. The problem has a closed form solution, provided $\boldsymbol{X}^T\boldsymbol{X}$ is invertible, given by

$$\hat{\boldsymbol{\beta}}_{\text{ols}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4.3}$$

with well known statistical properties. Although this estimator is optimal in the sense that it has the smallest variance among all linear and unbiased estimators, it may be nonetheless undesirable in certain circumstances. For instance, we may be willing to sacrifice the unbiasedness of $\hat{\boldsymbol{\beta}}_{\text{ols}}$ in exchange for smaller parameter variance, as this reduces the variance of predicted values and thus may increase the model's out-of-sample performance in prediction tasks (Hastie et al., 2009). Shrinkage and regularisation techniques can achieve precisely this.

4.2.2 Regularised least squares

A number of regularisation approaches have been proposed in the literature, all of which adopt the common strategy of adding a restriction of the form

$$\sum_{j=1}^{P} |\beta_j|^\gamma \leq c \tag{4.4}$$

to the OLS minimisation problem, for some γ of choice. By bounding the norm of the parameter vector, this class of restrictions imposes a penalty for complexity, resulting in shrinkage of regression coefficients towards zero. The most commonly used regularisation techniques in the context of linear regression include ridge (L_2; Hoerl and Kennard, 1970),

lasso (L_1; Tibshirani, 1996), bridge (a generalisation of the first two; Fu, 1998) and elastic net (a combination of the first two; Zou and Hastie, 2005). Regularisation techniques have been shown to perform better empirically than OLS in a number of tasks, although no one method seems to dominate the others (Fu, 1998; Tibshirani, 1996).

We focus on L_2 regularisation (ridge; henceforth RLS), which solves the problem

$$\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 \quad \text{s.t.} \quad ||\boldsymbol{\beta}||_2^2 \leq c \tag{4.5}$$

or, equivalently,

$$\min_{\boldsymbol{\beta}} \left\{ ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \alpha ||\boldsymbol{\beta}||_2^2 \right\} \tag{4.6}$$

for some $c, \alpha \in [0, \infty)$. Here, the L_2 norm $||\boldsymbol{\beta}||_2$ works as a measure of model complexity. Hence, more complex models will be penalised more, the larger the α. The standard solution to the problem,

$$\hat{\boldsymbol{\beta}}_{\text{rls}}(\alpha) = (\boldsymbol{X}^T\boldsymbol{X} + \alpha \boldsymbol{I}_{PP})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4.7}$$

reveals that α is also key in converting ill-conditioned problems into well-conditioned ones (Hoerl and Kennard, 1970). OLS is especially unstable in the presence of multicollinearity, where the advantages of RLS are two-fold: avoiding spurious results by conditioning the problem, and preventing overfitting by shrinking the parameters towards the origin.

Importantly, OLS can be retrieved from (4.7), and a profile can be computed for $\hat{\boldsymbol{\beta}}_{\text{rls}}(\alpha)$ as a function of α, ranging from $\mathbf{0}_P$ to $\hat{\boldsymbol{\beta}}_{\text{ols}}$ as α goes from $+\infty$ to 0.

Bayesian interpretation. The L_2-regularised least squares estimator in (4.27) has a straightforward correspondence with a Bayesian maximum a posteriori (MAP) estimate for a Normal conjugate prior on the regression coefficients (Gelman et al., 2004, §3.6). Consider the following likelihood and prior:

$$\boldsymbol{y}|\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{I}_{NN}) \tag{4.8a}$$

$$\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \tag{4.8b}$$

The resulting posterior distribution of the regression coefficients is then

$$\boldsymbol{\beta}|\boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{4.9}$$

where $\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Sigma}_0^{-1} + \frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{X}$ and $\boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 + \frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{y}\right)$. Now, for $\boldsymbol{\mu}_0 = \mathbf{0}_P$ and $\boldsymbol{\Sigma}_0 = \tau^{-1}\boldsymbol{I}_{PP}$, the posterior mean simplifies to

$$\boldsymbol{\mu} = (\boldsymbol{X}^T\boldsymbol{X} + \tau\sigma^2\boldsymbol{I}_{PP})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4.10}$$

which gives the Bayesian MAP estimator in the same form as the RLS estimator with $\alpha = \tau\sigma^2$.

4.3 Least squares via descent methods

Conventional solutions to (4.3) and (4.7) can be found in closed form using standard matrix inversion techniques with runtime complexity $\mathcal{O}(P^3)$ for the matrix inversion and $\mathcal{O}(NP^2)$ for the matrix product. Iterative methods, such as incremental forward stagewise regression (IFSR), are alternative methods that converge to the same solution as the number of iterations is increased, but typically with small computational complexity per iteration (Efron et al., 2004).

Consider the IFSR algorithm: let $\boldsymbol{X}_{\bullet j}$ denote the jth column of $\boldsymbol{X}$, that is, the observations on covariate j. Following initialisation of the parameter vector $\boldsymbol{\beta} = \mathbf{0}_P$ and the residuals $\boldsymbol{e} = \boldsymbol{y}$, at each iteration the covariate most correlated with the residual, and the residual, are updated:

$$\rho_j = \text{corr}(\boldsymbol{e}, \boldsymbol{X}_{\bullet j}), \forall j \in \mathbb{N}_{1:P} \tag{4.11a}$$

$$j_\star = \text{argmax}_j\, |\rho_j| \tag{4.11b}$$

$$\beta_{j_\star} = \beta_{j_\star} + \tau \cdot \text{sign}(\rho_{j_\star}) \tag{4.11c}$$

$$\boldsymbol{e} = \boldsymbol{e} - \boldsymbol{X}_{\bullet j_\star} \cdot \tau \cdot \text{sign}(\rho_{j_\star}) \tag{4.11d}$$

The steps in (4.11) are repeated for several iterations, with step size determined by τ, and $\text{sign}(\rho_{j_\star})$ giving the direction of the increment. By selecting the most highly correlated

variable at each step, Efron et al. (2004) showed that the algorithm produces sparse coefficient paths, and can closely approximate the coefficients of a L_1-regularised (lasso) regression (Tibshirani, 1996).

Our primary goal would be to implement a regularised regression algorithm of this type. However, the implementation of IFSR is impossible under FHE because the maximum, sign and absolute value functions cannot be computed when their inputs are ciphertexts. In what follows, we study two other iterative algorithms and their properties which make them suitable or unsuitable for homomorphic computation; and later we show how to incorporate regularisation (L_2, ridge) in these methods.

4.3.1 Iterative methods

Since the operations maximum, sign and absolute value have no equivalent in the ciphertext space, we study a modified version of IFSR for encrypted least squares (`ELS`) estimation that avoids these functions while retaining the iterative character of the algorithm, which are equivalent to known descent methods.

In fact, we analyse two variants of this modified algorithm, one which updates all parameters in $\boldsymbol{\beta}^{[k]}$ *simultaneously* at each iteration k, using the vector $\boldsymbol{\beta}^{[k-1]}$ (`ELS-GD`); and another which updates these parameters *sequentially*, using always the most current estimate for each parameter (`ELS-CD`). For example, if parameters are updated in ascending order at each iteration, the sequential algorithm updates

$$\begin{aligned}
\beta_1^{[k]} &= F\left(\boldsymbol{\beta}_{1:P}^{[k-1]}\right) \\
\beta_2^{[k]} &= F\left(\boldsymbol{\beta}_1^{[k]}, \boldsymbol{\beta}_{2:P}^{[k-1]}\right) \\
&\vdots \\
\beta_P^{[k]} &= F\left(\boldsymbol{\beta}_{1:(P-1)}^{[k]}, \boldsymbol{\beta}_P^{[k-1]}\right)
\end{aligned} \tag{4.12}$$

while the simultaneous algorithm updates

$$\begin{aligned} \beta_1^{[k]} &= F\left(\boldsymbol{\beta}_{1:P}^{[k-1]}\right) \\ \beta_2^{[k]} &= F\left(\boldsymbol{\beta}_{1:P}^{[k-1]}\right) \\ &\vdots \\ \beta_P^{[k]} &= F\left(\boldsymbol{\beta}_{1:P}^{[k-1]}\right) \end{aligned} \tag{4.13}$$

where F denotes the update function, potentially dependent on k and/or j. These two iterative modes of updates have known counterparts in the literature of linear system of equations. Namely, the sequential update mode (4.12) is related to the Gauss–Seidel method, while the simultaneous update mode (4.13) is related to the Jacobi method (Varga, 2000, chapter 3; Björck, 1996, chapter 7). These methods correspond to coordinate and gradient descent methods, respectively.

Note the important difference between coordinate and gradient descent: for K iterations, in CD each regression coefficient is updated K/P times while in GD each coefficient is updated K times. We consider an iteration to be a coefficient update that leads to an increase in the maximum depth of the parameter vector. In CD each update increases the depth, while in GD an increase only occurs at the end of each sweep through the P coefficients.

As we will see below, there are two competing concerns here: statistical efficiency and computational tractability within the cryptographic constraints, with these two types of updates having different properties.

4.3.1.1 Sequential updates via coordinate descent

In light of (4.11), sequentially updating the parameter vectors, which corresponds to iterations of the Gauss–Seidel type, seems the most natural and desirable approach. In this variant, at each iteration k we choose one variable, say $j_k \in \mathbb{N}_{1:P}$, and update the corresponding regression parameter:

$$\beta_{j_k}^{[k]} = \beta_{j_k}^{[k-1]} + \frac{\boldsymbol{X}_{\bullet j_k}^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{\text{old}})}{\boldsymbol{X}_{\bullet j_k}^T \boldsymbol{X}_{\bullet j_k}} \tag{4.14}$$

where $\boldsymbol{\beta}^{\text{old}}$ old contains the components updated on the previous iteration $(k-1)$ or the current one (k) as appropriate, there being many variants for the schedule of coordinates to update. Because only one parameter (coordinate) is updated at each iteration, this type of method is also called *coordinate descent* (Wright, 2015).

However, this cannot be computed under FHE because the required data-dependent division is not feasible. In an alternative variant we can replace this data-dependent factor with a generic step-size δ, and at each iteration update:

$$\beta_{j_k}^{[k]} = \beta_{j_k}^{[k-1]} + \delta \boldsymbol{X}_{\bullet j_k}^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{\text{old}}) \tag{4.15}$$

Two things are noteworthy: first, these equations require only polynomial-type operations; and second, a rescaling is necessary since only integer polynomial functions can be computed homomorphically, and the parameter δ is usually not an integer. We will show how to do this type of rescaling in §4.3.1.2.

As each update uses the most recent estimate of every parameter, the maximum multiplicative depth (MMD) grows by 2 with each parameter update (due to the second term in the update of $\boldsymbol{e}^{[k]}$). This implies that for K iterations over P covariates the MMD is equal to $2KP$. Even though the MMD is independent of N, it is linearly dependent of P. This renders the algorithm very expensive computationally, as it requires bootstrapping of ciphertexts for problems with even moderately large P. As discussed in §2.4.3, bootstrapping in current homomorphic schemes is to be avoided wherever possible because it is very computationally expensive.

4.3.1.2 Simultaneous updates via gradient descent

To circumvent the main problem of CD — namely, the high multiplicative depth — we propose the use of simultaneous updates. That is, we update the entire vector $\boldsymbol{\beta}^{[k]}$ based on the vector $\boldsymbol{\beta}^{[k-1]}$, corresponding iterations of the Jacobi type. This reduces the multiplicative depth from $2KP$ to $2K$, thus crucially independent of P, which enables scalability to higher dimensional models without bootstrapping the ciphertexts. In this case, the update equations

for the regression parameters are as follows:

$$\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} + \delta \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}). \tag{4.16}$$

The choice $\delta = (\boldsymbol{X}^T\boldsymbol{X})^{-1}$ gives the OLS solution in one step; that is, $\boldsymbol{\beta}^{[0]} = \mathbf{0}_P$ implies $\boldsymbol{\beta}^{[1]} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. Since this is infeasible in the context of homomorphic computation, we choose δ to be a small value which controls the increment to the previous estimate, also called the step size. As it turns out, this is equivalent to *gradient descent*: the function to be optimised is $S(\boldsymbol{\beta}) = ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2$ and its gradient is

$$\nabla S(\boldsymbol{\beta}) = \frac{\partial S(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \tag{4.17}$$

and gradient descent can be written as $\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} - \delta\nabla S(\boldsymbol{\beta}^{[k-1]})$, which reduces to (4.16). In the numerical analysis literature this is also known as Richardson first-order method (Ryaben'kii and Tsynkov, 2006, chapter 6).

As mentioned in the previous section, the update equations must be rescaled for homomorphic computation. This is strictly necessary to accommodate the transformed data (see §4.1) and to overcome the fact that we cannot divide and, typically, $\delta < 1$.

Let $\delta \equiv 1/\nu$ for $\nu \in \mathbb{N}$; and consider the following transformations, according to the method proposed in §4.1: $\tilde{\boldsymbol{X}} \approx 10^{\phi}\boldsymbol{X}$, $\tilde{\boldsymbol{y}} \approx 10^{\phi}\boldsymbol{y}$, $\tilde{\nu} \approx 10^{\phi}\nu$, $\tilde{\boldsymbol{\beta}}^{[0]} \approx 10^{\phi}\boldsymbol{\beta}^{[0]}$. At iteration $k \in \mathbb{N}_{1:K}$ we want to compute

$$\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} + \delta \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) \tag{4.18}$$

or, using the transformed data,

$$\tilde{\boldsymbol{\beta}}^{[k]} \equiv F_k\left(\tilde{\boldsymbol{\beta}}^{[k-1]}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \tilde{\nu}\right) \propto \boldsymbol{\beta}^{[k]} \tag{4.19}$$

so that the true parameters, $\boldsymbol{\beta}^{[k]}$, can be easily obtained from the homomorphically computed parameters, $\tilde{\boldsymbol{\beta}}^{[k]}$. What remains to derive is the exact form of the transformed update

Algorithm 4.1 `ELS-GD`: encrypted least squares via gradient descent

Input: data $\{\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}\}$, parameters $\{K, \tilde{\nu}\}$
initialise: $\tilde{\boldsymbol{\beta}}^{[0]} \leftarrow \mathbf{0}_P$
for $k = 1$ **to** K **do**
  $\tilde{\boldsymbol{\beta}}^{[k]} \leftarrow 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{X}}^T(10^{k\phi}\tilde{\nu}^{k-1}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]})$
end for

equations $\{F_k\}_{k\in\mathbb{N}_{1:K}}$. For $k = 1$ we have:

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[1]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[0]} + \tilde{\boldsymbol{X}}^T(10^{\phi}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[0]}) \\
&= 10^{\phi}\cdot 10^{\phi}\nu\cdot 10^{\phi}\boldsymbol{\beta}^{[0]} + 10^{\phi}\boldsymbol{X}^T(10^{\phi}\cdot 10^{\phi}\boldsymbol{y} - 10^{\phi}\boldsymbol{X}\cdot 10^{\phi}\boldsymbol{\beta}^{[0]}) \\
&= 10^{3\phi}\nu\boldsymbol{\beta}^{[0]} + 10^{3\phi}\nu\delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[0]}) \\
&= 10^{3\phi}\nu\{\boldsymbol{\beta}^{[0]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[0]})\} \\
&\equiv 10^{3\phi}\nu\boldsymbol{\beta}^{[1]}
\end{aligned} \tag{4.20}$$

where the term $10^{3\phi}\nu$ is data-independent and can be reveled without compromising privacy. The secret key holder can then obtain the true regression parameters from the by computing $\boldsymbol{\beta}^{[1]} = \mathtt{Dec}_{sk}(\tilde{\boldsymbol{\beta}}^{[1]})/(10^{3\phi}\nu)$. By using (4.18) recursively we can determine F_k for any $k \in \mathbb{N}_{1:K}$ (see §B.2 for details):

$$\tilde{\boldsymbol{\beta}}^{[k]} \equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{X}}^T(10^{k\phi}\tilde{\nu}^{k-1}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]}) \tag{4.21a}$$

$$= 10^{(2k+1)\phi}\nu^k\boldsymbol{\beta}^{[k]} \tag{4.21b}$$

The rescaling factors are independent of the data and known a priori, and so can be grouped during computation, e.g., $10^{k\phi}\tilde{\nu}^{k-1}$ in (4.21a) can be encrypted as a single value. The procedure is summarised in Algorithm 4.1. Retrieval of the coefficient can be done by the secret key holder by computing $\boldsymbol{\beta}^{[K]} \leftarrow \mathtt{Dec}_{\mathtt{sk}}(\tilde{\boldsymbol{\beta}}^{[K]})/(10^{(2K+1)\phi}\nu^K)$.

Theorem 4.1 (Convergence of `ELS-GD`). *Define* $\boldsymbol{\beta}^{[k+1]} \equiv \boldsymbol{\beta}^{[k]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k]})$ *and let* $\boldsymbol{\beta}^{[0]} \equiv \mathbf{0}_P$. *Then,* $\lim_{k\to\infty}\boldsymbol{\beta}^{[k]} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \hat{\boldsymbol{\beta}}_{ols}$ *for* $\delta \in (0, 2/\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X}))$ *where* $\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$ *is the spectral radius of* $\boldsymbol{X}^T\boldsymbol{X}$.

The proof of Theorem 4.1 rests on the fact that the gradient descent iterative process (4.16) can be written as $\boldsymbol{\beta}^{[k]} = \boldsymbol{M}\boldsymbol{\beta}^{[k-1]} + \delta\boldsymbol{X}^T\boldsymbol{y}$, where $\boldsymbol{M} = \boldsymbol{I}_{PP} - \delta\boldsymbol{X}^T\boldsymbol{X}$ is called the *iteration matrix* and its spectrum defines the convergence properties of the algorithm. The conditions

given in the theorem follow from the spectral analysis of $\boldsymbol{M}$ (see §B.1 for a proof). Figure 4.1 shows the behaviour of the gradient descent algorithm around the threshold $2/\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$.

The method can also be cast in the framework of Lipschitz continuity. The Lipschitz constant L corresponds to $\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$ in the case of linear regression and gradient descent is guaranteed to converge when $\delta < 1/L$, a more restrictive bound than that provided in Theorem 4.1.

The optimal choice of step size, in the sense that it minimises the spectral radius, is $\delta_\star = 2/(\lambda_{\max}+\lambda_{\min})$, implying an optimal spectral radius $\mathcal{S}_\star = (\lambda_{\max}-\lambda_{\min})/(\lambda_{\max}+\lambda_{\min})$, where $\lambda_{\max}$ and $\lambda_{\min}$ denote the largest and smallest eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$, respectively (Ryaben'kii and Tsynkov, 2006, theorem 6.3).

The iterative process (4.16) can be written recursively as

$$\boldsymbol{\beta}^{[k]} = \sum_{n=1}^{k}(-1)^{n+1}\binom{k}{k-n}\delta^n(\boldsymbol{X}^T\boldsymbol{X})^{n-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{4.22}$$

Although GD is optimal in the sense mentioned above, we show in §4.4 that it is possible to improve on this by using acceleration methods that exploit the oscillatory nature of the gradient descent algorithm. This result is crucial as it allows us to make use of a particularly useful method for accelerating the convergence of the series $\{\boldsymbol{\beta}^{[k]}\}_{k\geq 0}$ (presented in §4.4.2). Additionally, it illustrates the role of δ^n as a damping term in the algorithm.

4.3.2 Prediction

Note that the form of the gradient descent equations (4.21) implies a common scaling factor, $10^{(2k+1)\phi}\nu^K$, for all parameters. Performing encrypted prediction is then straightforward as it requires only the computation of a dot product:

$$\tilde{y}_i = \tilde{\boldsymbol{X}}_{i\bullet}^T\tilde{\boldsymbol{\beta}}^{[K]} = 10^{(2K+1)\phi}\nu^K\hat{y}_i \tag{4.23}$$

where $\hat{y}_i = \boldsymbol{X}_{i\bullet}^T\boldsymbol{\beta}^{[K]}$ denotes the predicted values in the space of the original data and $\tilde{y}_i$ denotes the corresponding transformed version. Upon receiving the transformed predicted values, the secret key holder can obtain the desired predictions by simply rescaling the result: $\mathtt{Dec}_{\mathtt{sk}}(\tilde{y}_i)/(10^{(2K+1)\phi}\nu^K)$. The procedure increases the MMD of the algorithm by only 1.

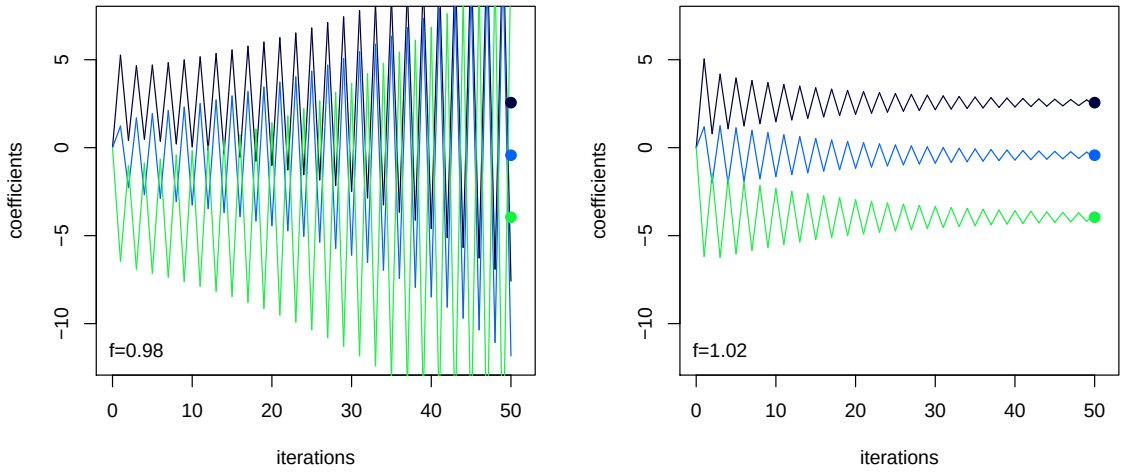


Figure 4.1: Convergence of `ELS-GD` for different values of $\delta = 2/(\mathrm{f} \cdot \lambda_{\max})$, where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{X}^T\boldsymbol{X}$ (see Theorem 4.1 for an explanation). We show two regimes: **[left]** oscillatory divergence, $\mathrm{f} = 0.98 \Rightarrow \delta > 2/\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$; **[right]** oscillatory convergence, $\mathrm{f} = 1.02 \Rightarrow \delta < 2/\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$. Full circles represent OLS estimates.

4.3.3 Inference

Inference in the linear regression model (e.g., confidence intervals, hypothesis testing, variable selection) requires knowledge of the standard errors of the regression coefficients,

$$\mathbb{V}[\hat{\boldsymbol{\beta}}_{\text{ols}}] = \hat{\sigma}^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}, \qquad \hat{\sigma}^2 = \boldsymbol{e}^T\boldsymbol{e}/(N-P). \tag{4.24}$$

The matrix inversion is intractable under homomorphic computation except for very small P, and so standard errors cannot be computed as usual. An alternative is to estimate the standard errors by bootstrapping: (1) sample N values from $\mathbb{N}_{1:N}$ with replacement to obtain a vector of resampled indices $\boldsymbol{R}_b$; (2) construct the bth bootstrap dataset $\boldsymbol{D}_b = \{\boldsymbol{X}_{\boldsymbol{R}_b\bullet}, \boldsymbol{y}_{\boldsymbol{R}_b}\}$; (3) compute $\hat{\boldsymbol{\beta}}_b$ via `ELS-GD` using $\boldsymbol{D}_b$; (4) repeat this process B times to obtain $\{\hat{\boldsymbol{\beta}}_b\}_{b\in\mathbb{N}_{1:B}}$. Then, for each $j \in \mathbb{N}_{1:P}$, we have:

$$\mathbb{V}[\hat{\boldsymbol{\beta}}_j] \approx \frac{1}{B}\sum_{b=1}^{B}(\hat{\boldsymbol{\beta}}_{(b)j} - \bar{\boldsymbol{\beta}}_j)^2, \qquad \bar{\boldsymbol{\beta}}_j = \frac{1}{B}\sum_{b=1}^{B}\hat{\boldsymbol{\beta}}_{(b)j}. \tag{4.25}$$

All divisions involve known, constant denominators, which can be factored out as usual. Thus, the server can compute a quantity proportional to $\mathbb{V}[\hat{\boldsymbol{\beta}}_j]$ and return it to the client,

who uses this to do inference.

4.3.4 Regularisation

L_2-regularised (ridge) least squares is easy to implement using the well known data augmentation procedure (Allen, 1974):

$$\mathring{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{X} \\ \sqrt{\alpha}\boldsymbol{I}_{PP} \end{bmatrix} \qquad \text{and} \qquad \mathring{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0}_P \end{bmatrix}. \tag{4.26}$$

OLS estimates when using the augmented data, $(\mathring{\boldsymbol{X}}, \mathring{\boldsymbol{y}})$, are equivalent to RLS estimates when using the original data, $(\boldsymbol{X}, \boldsymbol{y})$; that is,

$$\hat{\boldsymbol{\beta}}_{\text{rls}}(\alpha) = (\mathring{\boldsymbol{X}}^T\mathring{\boldsymbol{X}})^{-1}\mathring{\boldsymbol{X}}^T\mathring{\boldsymbol{y}} = (\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{I}_{PP})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4.27}$$

Because the augmentation terms ($\sqrt{\alpha}\boldsymbol{I}_{PP}$ and $\boldsymbol{0}_P$) are independent of the data ($\boldsymbol{X}$ and $\boldsymbol{y}$), the iterative methods already developed for least squares (§4.3.1) can be used with the augmented data to obtain regularised parameter estimates. The proof of convergence follows from Theorem 4.1 with the only difference that convergence is now to $\hat{\boldsymbol{\beta}}_{\text{rls}}(\alpha)$ in (4.7) rather than $\hat{\boldsymbol{\beta}}_{\text{ols}}$ in (4.3).

We can also determine the effect of the regularisation parameter α on the spectrum of $\mathring{\boldsymbol{X}}^T\mathring{\boldsymbol{X}}$, compared to the spectrum of $\boldsymbol{X}^T\boldsymbol{X}$. Letting $\{\mathring{\lambda}_j\}_{j\in\mathbb{N}_{1:P}}$ denote the eigenvalues of $\mathring{\boldsymbol{X}}^T\mathring{\boldsymbol{X}}$, we can show that

$$\mathring{\lambda}_{\max} = \lambda_{\max} + \alpha \tag{4.28}$$

and so a new step size $\mathring{\delta}$ can be chosen for the regularised problem straightforwardly (according to Theorem 4.1), for any penalty α, without requiring any additional computation.

4.3.5 Theoretical parameter requirements for the `FV` cryptosystem

Here we give some guidance on how to choose the cryptographic parameters in the `FV` scheme to run `ELS-GD`. We face the growth of the transformed regression coefficients, and we need

to ensure that the MCS (`max`) is also large enough to accommodate this value. The MCS to be stored is $||\tilde{\boldsymbol{\beta}}^{[K]}||_\infty = \max_{i\in\mathbb{N}_{1:P}}\{\tilde{\boldsymbol{\beta}}_i^{[K]}\}$, the value of the largest transformed regression coefficient at the last iteration of the GD algorithm as given in (4.21).

Let $s \in \{x \in \mathbb{R} : -10 \leq x \leq 10\}$ denote the message to be encrypted (standardised); let $\tilde{s} \in \{x \in \mathbb{Z} : -10^{\phi+1} \leq x \leq 10^{\phi+1}\}$ denote the corresponding transformed message, according to the encoding method presented in §3.1.2; and let $\mathring{s}(x)$ denote the corresponding binary polynomial representation, as in §2.4.1. ϕ denotes the input accuracy (number of decimal places to be retained). The maximum degree of $\mathring{s}(x)$ is then $\deg(\mathring{s}) = \log_2(10^{\phi+1}) = (\phi+1)\log_2(10) \equiv n$.

Now, let a denote a fresh ciphertext and b denote a ciphertext that is the result of homomorphic computation, both in polynomial form. By definition, $\deg(a) \leq n \leq \deg(b) \leq m$. Additionally, $||a||_\infty = 1$ and assume $||b||_\infty = B$. Assume also that $\nu \leq 10^{2\phi}$. Then, we have the following rules, which we will use to derive an upper bound on $||\tilde{\boldsymbol{\beta}}^{[K]}||_\infty$ (by which we mean the largest coefficient in polynomial representation of $\tilde{\boldsymbol{\beta}}^{[K]}$):

$$(i) \qquad ||a+b||_\infty \leq B+1 \tag{4.29a}$$

$$(ii) \qquad ||ab||_\infty \leq (n+1)B \tag{4.29b}$$

For iteration $k=1$, (4.21) simplifies to $\tilde{\boldsymbol{\beta}}^{[1]} = 10^\phi \tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}$, which we can decompose as $\tilde{\boldsymbol{\beta}}^{[1]} = 10^\phi \tilde{\boldsymbol{\beta}}'^{[1]}$, where $\tilde{\boldsymbol{\beta}}'^{[1]} = \tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}$. The matrix-vector product yields $||\boldsymbol{\beta}'^{[1]}||_\infty = ||\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}||_\infty \leq N(n+1)$. Putting this together with the scalar-vector product gives:

$$||\tilde{\boldsymbol{\beta}}^{[1]}||_\infty \leq \phi\log_2(10)N(n+1) \leq nN(n+1). \tag{4.30}$$

The procedure can be applied recursively (see §B.3 for details) to give:

$$||\tilde{\boldsymbol{\beta}}^{[k]}||_\infty \leq (4n+(n+1)^2)NP||\boldsymbol{\beta}^{[k-1]}||_\infty + (4k-3)n(n+1)N \tag{4.31}$$

and from this we can determine the MCS required. The MMD (`L`) grows linearly with the number of iterations and is equal to $2K$.

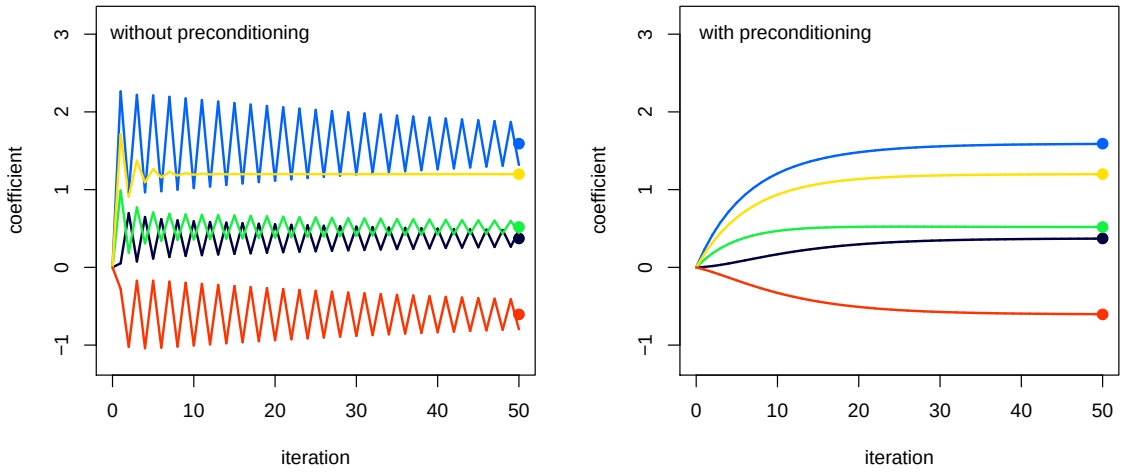


Figure 4.2: The effect of diagonal scaling preconditioning on the convergence paths of `ELS-GD`. Full circles represent OLS estimates. $[N = 100, P = 5, \rho = 0.1]$

4.4 Acceleration

Although `ELS-GD` is guaranteed to convergence to the OLS solution, the rate of convergence can be slow, for instance when predictors are highly correlated. Here we propose some classic methods to accelerate convergence and, therefore, to mitigate this problem.

4.4.1 Preconditioning

Preconditioning is often used to accelerate convergence of iterative methods (Björck, 1996, chapter 7). The idea is to introduce a preconditioning matrix $\boldsymbol{D}$ and solve the preconditioned problem:

$$\boldsymbol{D}^{-1}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{X}^T\boldsymbol{y}) = 0 \tag{4.32}$$

with the same solution as the original problem:

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{X}^T\boldsymbol{y} = 0 \tag{4.33}$$

but with the preconditioned iteration matrix $\boldsymbol{M_D} = \boldsymbol{I}_{PP} - \delta\boldsymbol{D}^{-1}\boldsymbol{X}^T\boldsymbol{X}$ having a more favourable spectrum than the original $\boldsymbol{M} = \boldsymbol{I}_{PP} - \delta\boldsymbol{X}^T\boldsymbol{X}$.

A simple way to precondition the system is by diagonal scaling of the columns of $\boldsymbol{X}$. Let $\boldsymbol{D} = \text{diag}(d_1, \ldots, d_P)$ be a $P \times P$ diagonal matrix with diagonal entries $\{d_1, \ldots, d_P\}$ where $d_j = ||\boldsymbol{X}_{\cdot j}||_2^2$. Since $d_j \approx N$ for all $j \in \mathbb{N}_{1:P}$ as a result of standardisation (see §4.1) the preconditioning matrix becomes

$$\boldsymbol{D}^{-1} \approx \text{diag}(1/N, \ldots, 1/N) = N^{-1}\boldsymbol{I}_{PP} \tag{4.34}$$

The preconditioned update equation for the regression parameters is then

$$\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} + (\delta/N)\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) \tag{4.35}$$

which differs from (4.16) only in the step size, which is rescaled from δ to δ/N. Preconditioning the system effectively accelerates convergence by smoothing the convergence path (Figure 4.2) but the number of iterations required is still large in the context of homomorphic encryption.

4.4.2 van Wijngaarden transformation

The van Wijngaarden (1965) transformation (VWT) is a method for accelerating the convergence of an alternating series and it is a variant of the Euler transform. Given the partial sums of an alternating series,

$$S_{1,k} = \sum_{n=1}^{k}(-1)^n a_n \tag{4.36}$$

we can compute averages of these partial sums,

$$S_{z,k} = \frac{S_{z-1,k} + S_{z-1,k+1}}{2}, \quad z \in \mathbb{N}_{2:K} \tag{4.37}$$

(and averages of the averages) to form a matrix of averaged partial sums. The interesting fact about this process is that, for finite numbers of terms (K), averages of partial sums are often closer to the limiting value, $S_{1,\infty}$, than any single partial sum of the original series, $S_{1,k}$ ($k \in \mathbb{N}_{1:K}$). In short, the alternating nature of the sequence is averaged out, damping its oscillatory behaviour and speeding-up convergence (Table 4.1).

Table 4.1: Using the van Wijngaarden transformation to accelerate the convergence of the Leibniz formula to approximate $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = 0.7854$. The VWT estimates are given by the last element of each column, the one with smallest error being $S_{4,2}$ (highlighted), resulting in an improvement of 0.048 (6%) versus $S_{1,5}$, the standard estimate.

k	1	2	3	4	5
$S_{1,k}$	1.0000	0.6667	0.8667	0.7238	0.8349
$S_{2,k}$	0.8333	0.7667	0.7952	0.7794	
$S_{3,k}$	0.8000	0.7810	0.7873		
$S_{4,k}$	0.7905	**0.7841**			
$S_{5,k}$	0.7873				
ERROR	-0.0019	**0.0013**	-0.0019	0.0060	-0.0495

Algorithm 4.2 `ELS-GD-VWT`: encrypted least squares via VWT-accelerated gradient descent

Input: data $\{\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}\}$, parameters $\{K, \tilde{\nu}\}$
initialise: $\tilde{\boldsymbol{\beta}}^{[0]} \leftarrow \mathbf{0}_P$
for $k = 1$ **to** K **do**
  $\tilde{\boldsymbol{\beta}}^{[k]} \leftarrow 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{X}}^T(10^{k\phi}\nu^{k-1}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]})$
end for
$k^\star \leftarrow \lfloor K/3 \rfloor + 1$
$\tilde{\boldsymbol{\beta}}_{\text{vwt}} \leftarrow \sum_{k=k^\star}^{K} \binom{K-k^\star}{k-k^\star} \tilde{\boldsymbol{\beta}}^{[k]}$

As shown in (4.22), the values $\{\boldsymbol{\beta}^{[k]}\}_{k \in \mathbb{N}_{1:K}}$ computed with `ELS-GD` form an alternating series. As a result, the VWT provides a valid way to accelerate the convergence of the regression coefficients. van Wijngaarden proposed to include only the last two thirds of the series in this calculation. By discarding some of the earlier estimates, this compromise balances the fact that, on the one hand, averaging improves convergence while, on the other, the early estimates tend to be further away from the limiting value, thus worsening the overall estimate.

The implementation of (4.37) has a simple, closed form solution. For a stopping column $k^\star = \lfloor K/3 \rfloor + 1$ we compute the average partial sum

$$S_\star = \frac{1}{2^{K-k^\star}} \sum_{n=k^\star}^{K} \binom{K-k^\star}{n-k^\star} S_{1,n} \tag{4.38}$$

and take this as our best approximation to the value of the series at convergence. The procedure is summarised in Algorithm 4.2.

Notably, this can be implemented homomorphically with the exception of the division, but since this is a constant factor (independent of the data) we can compute instead $\tilde{S}_\star = 2^{K-k^\star} S_\star$ and do the appropriate correction upon decryption. The computational cost of the VWT is

Algorithm 4.3 `ELS-NAG`: encrypted least squares via Nesterov's accelerated gradient

Input: data $\{\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}\}$, parameters $\{K, \tilde{\nu}, \tilde{\boldsymbol{\eta}}\}$
initialise: $\tilde{\boldsymbol{s}}^{[0]} \leftarrow \mathbf{0}_P$ and $\tilde{\boldsymbol{\beta}}^{[0]} \leftarrow \mathbf{0}_P$
for $k = 1$ **to** K **do**
  $\tilde{\boldsymbol{s}}^{[k]} \leftarrow 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{X}}^T(10^{(2k-1)\phi}\tilde{\nu}^{k-1}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]})$
  $\tilde{\boldsymbol{\beta}}^{[k]} \leftarrow (10^{\phi} + \tilde{\eta}_k)\tilde{\boldsymbol{s}}^{[k]} - 10^{2\phi}\tilde{\nu}\tilde{\eta}_k\tilde{\boldsymbol{s}}^{[k-1]}$
end for

minimal, involving approximately $2K/3$ additions and multiplications, where K will typically be small; and increases the MMD of the algorithm by only 1.

4.4.3 Nesterov's accelerated gradient

Nesterov's accelerated gradient (NAG) achieves a convergence rate of $\mathcal{O}(1/K^2)$ as opposed to the $\mathcal{O}(1/K)$ achieved by regular GD (Nesterov, 1983). The NAG algorithm can be written as follows:

$$\boldsymbol{s}^{[k]} = \boldsymbol{\beta}^{[k-1]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) \tag{4.39a}$$

$$\boldsymbol{\beta}^{[k]} = \boldsymbol{s}^{[k]} + \eta_k(\boldsymbol{s}^{[k]} - \boldsymbol{s}^{[k-1]}) = (1 + \eta_k)\boldsymbol{s}^{[k]} - \eta_k\boldsymbol{s}^{[k-1]}, \qquad \eta_k < 0. \tag{4.39b}$$

The first step, (4.39a), is the same as the standard gradient descent step in (4.16). Additionally, an extra step is taken, (4.39b), which is proportional to the momentum term $\boldsymbol{s}^{[k]} - \boldsymbol{s}^{[k-1]}$, and which is responsible for the acceleration. Intuitively, the algorithm takes a bigger extra step if a parameter was updated by a large amount in the previous iteration, which happens when that parameter is far from its value at convergence.

As before, these equations must be rescaled for homomorphic computation. The transformed equations are as follows:

$$\tilde{\boldsymbol{s}}^{[k]} \equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{X}}^T(10^{(2k-1)\phi}\tilde{\nu}^{k-1}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]}) = 10^{3k\phi}\nu^k\boldsymbol{s}^{[k]} \tag{4.40a}$$

$$\tilde{\boldsymbol{\beta}}^{[k]} \equiv (10^{\phi} + \tilde{\eta}_k)\tilde{\boldsymbol{s}}^{[k]} - 10^{2\phi}\tilde{\nu}\tilde{\eta}_k\tilde{\boldsymbol{s}}^{[k-1]} = 10^{(3k+1)\phi}\nu^k\boldsymbol{\beta}^{[k]} \tag{4.40b}$$

where $\tilde{\boldsymbol{y}} = 10^{\phi}\boldsymbol{y}$ is the transformed vector of responses, according to §4.1, and similarly for the remaining variables, except $\{\tilde{\boldsymbol{s}}^{[k]}, \tilde{\boldsymbol{\beta}}^{[k]}\}_{k\in\mathbb{N}_{0:K}}$ (see §B.4 for details). The method is synthesised in Algorithm 4.3.

All scaling constants are independent of the data and known a priori, and so can be used by the secret key holder to obtain the parameter estimates as follows: $\texttt{Dec}_{\texttt{sk}}(\tilde{\boldsymbol{\beta}}^{[K]})/(10^{(3K+1)\phi}\nu^K)$. Because of the extra acceleration step, `ELS-NAG` has a MMD equal to $3K$ (see Table 4.2).

Table 4.2: Maximum multiplicative depth for different acceleration methods.

Algorithm	**MMD**
Preconditioned gradient descent	2K
van Wijngaarden transformation	2K+1
Nesterov's accelerated gradient	3K

4.5 Results

In this section we illustrate the methods proposed for encrypted linear regression using simulated and real data. The implementation uses the `FV` cryptosystem presented in §2.4 (Aslett, Esperança, and Holmes, 2015a; Fan and Vercauteren, 2012). The runtimes reported for encrypted analysis are relative to a 48-core server (see §A.3 for hardware and software details). Memory usage is proportional to the amount of encrypted data stored and increases with the multiplicative depth parameter (see Figure 2.7).

We use the root mean squared deviation (RMSD) of two vectors, $\boldsymbol{u}$ and $\boldsymbol{v}$ of length n, as a distance measure between the (elements of the) two vectors: $\text{rmsd}(\boldsymbol{u},\boldsymbol{v}) = \sqrt{(\boldsymbol{u}-\boldsymbol{v})^T(\boldsymbol{u}-\boldsymbol{v})/n}$. If $\boldsymbol{v}$ is the target quantity of the process generating $\boldsymbol{u}$, then $\text{rmsd}(\boldsymbol{u},\boldsymbol{v})$ represents the L_2 error norm of $\boldsymbol{u}$ with respect to the target $\boldsymbol{v}$.

This section is divided into two parts. In the first, we present some results with simulated data and give important insights into the behaviour of the algorithms in different scenarios (§4.5.1). In the second, we apply the methods to real data sets and give details on computational costs and performance (§4.5.2). We use $\phi = 2$ throughout.

4.5.1 Simulations

For simulations with *independent data* we generate regression coefficients $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{0}_P, \boldsymbol{I}_{PP})$, covariates $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}_P, \boldsymbol{\Sigma})$ and responses $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{I}_{PP})$. For simulation with *correlated data*

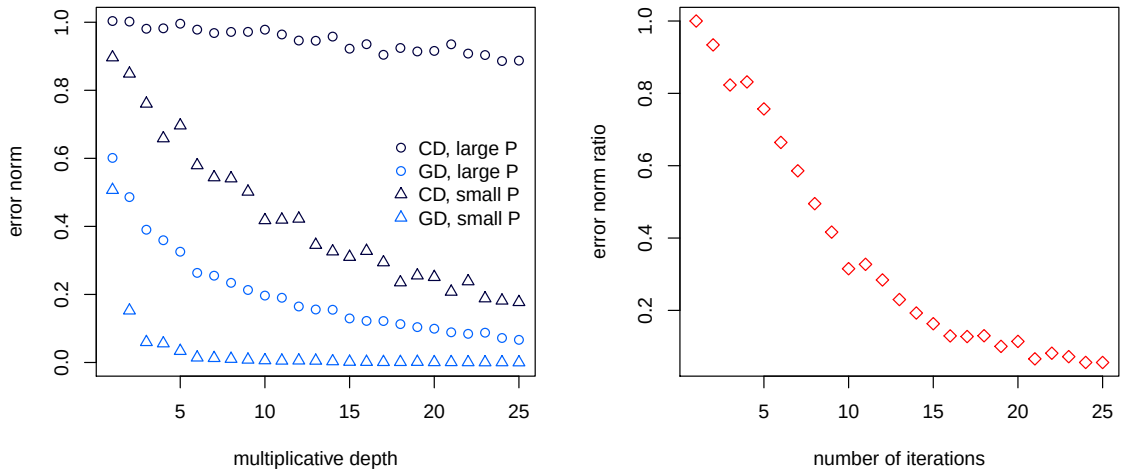


Figure 4.3: Performance of `ELS-CD`, `ELS-GD` and `ELS-GD-VWT` on simulated, independent data. **[left]** `ELS-CD` vs. `ELS-GD`. The values correspond to the error norm with respect to the OLS solution. The comparison between models is done for a fixed multiplicative depth, which is a function of the number of updates and is equal to $2KP$ for CD and $2K$ for GD; **[right]** Acceleration via VWT. The values shown are ratios of the error norm of `ELS-GD-VWT` to that of `ELS-GD`, where the error norm is relative to the OLS solution. [$N = 100$, large $P = 50$, small $P = 5$]

we use Normal copulas: we generate predictors whose pairwise correlations are all equal to ρ, that is, $\boldsymbol{\Sigma} = \rho \mathbf{1}_{PP} - (\rho - 1)\boldsymbol{I}_{PP}$, meaning $\text{corr}(\boldsymbol{X}_{\cdot j}, \boldsymbol{X}_{\cdot j'}) \approx \rho$, for all $j \neq j'$. Responses and regression coefficients are generated as in the case of independent data.

Independent data

We start by illustrating the computational properties of the coordinate and gradient descent methods for encrypted regression (Figure 4.3, left). For fairness, we compare the performance of CD and GD algorithms for a fixed maximum multiplicative depth (MMD). Since the MMD is a prime determinant of the computational cost of homomorphic operations, this gives a proxy for the error as a function of computational complexity. `ELS-GD-VWT` outperforms `ELS-CD` overall, for a fixed computational cost, and is thus preferred. Notice that because convergence is monotonic in the case of CD, the VWT cannot be applied to accelerate convergence.

Focusing now on the GD algorithms, we examine the improvement provided by the VWT (Figure 4.3, right). The ratio of the error norms of `ELS-GD-VWT` to `ELS-GD` approaches zero as the number of iterations increases, as a result of the acceleration provided by the VWT.

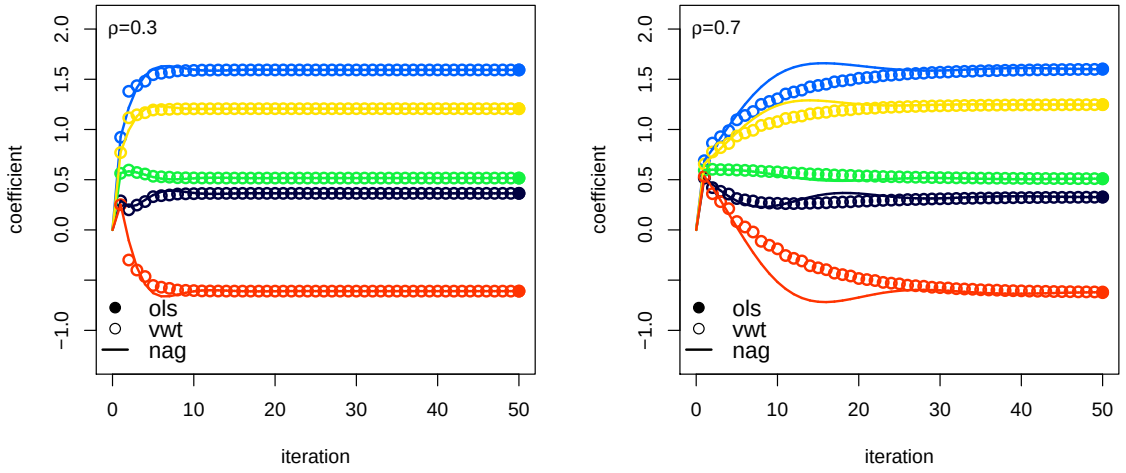


Figure 4.4: Convergence of `ELS-GD-VWT` and `ELS-NAG` for different levels of correlation: **[left]** $\rho = 0.3$; **[right]** $\rho = 0.7$. $[N = 100, P = 5]$

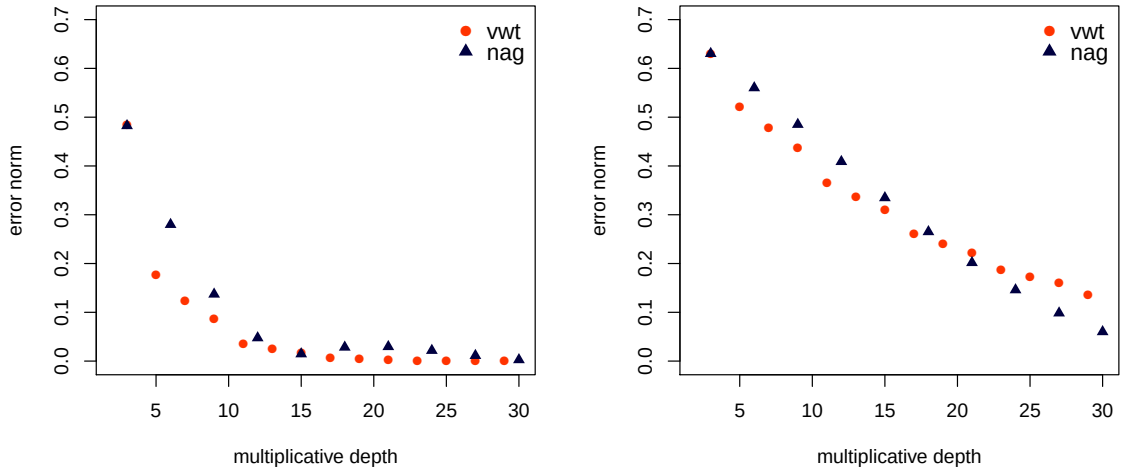


Figure 4.5: Error norm of `ELS-GD-VWT` and `ELS-NAG` as a function of multiplicative depth, for different levels of correlation: **[left]** $\rho = 0.3$; **[right]** $\rho = 0.7$. $[N = 100, P = 5]$

Correlated data

Naturally, one should question how these methods behave when predictors are correlated. In general, higher correlation among predictors implies less favorable spectral properties for $\boldsymbol{X}^T\boldsymbol{X}$, which in turn makes convergence slower. For both `ELS-GD-VWT` and `ELS-NAG`, convergence is slowed down as correlation increases (Figure 4.4).

Because computational costs are very high, a fair comparison must control for the fact that the two algorithms have different computational complexities per iteration. Using the MMD

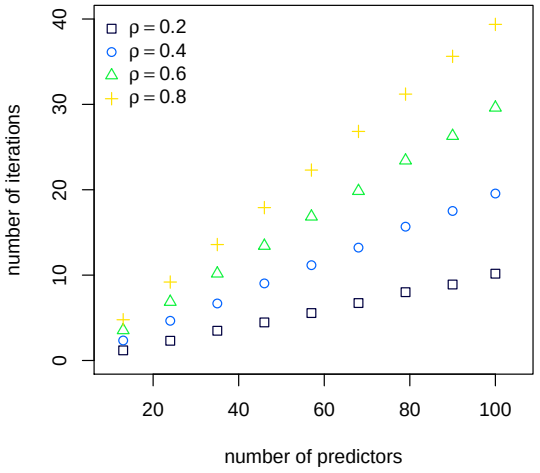


Figure 4.6: A measure of the maximum number of iterations required to reduce the norm of the initial error vector by a factor e in `ELS-GD` using simulated data. For a given level of correlation among predictors, the number of required iterations is linear in P, a good property for scalability. This measure is the reciprocal of the average convergence rate. Results are averaged over 100 simulations. $[N = 100, K = 4]$

as a proxy, we can compare the error for a fixed complexity level. The results show that `ELS-GD-VWT` typically outperforms `ELS-NAG` (Figure 4.5). For very high correlation settings, however, this relationship can be reversed, but only for large numbers of iterations, which will typically not be the case in practice, as this pushes up computational costs very considerably.

We stress that this choice is conditional on the computational framework that we are working under. Though in an unrestricted framework NAG is the state-of-the-art, in the restricted framework of FHE the VWT is a better choice.

The convergence properties are also affected by the number of predictors in the model. In particular, the maximum number of iterations required to reduce the norm of the initial error vector by a factor e (equivalent to the reciprocal of the average convergence rate; Varga, 2000, p.69), when analysed for different problem sizes, gives us an idea of the relationship between number of predictors and speed of convergence. For any level of correlation, this measure of complexity increases linearly with the number of predictors, which is a good property when considering scalability to large-P datasets (Figure 4.6).

Finally, the computational costs of `ELS-GD` are given in Figure 4.7. Runtime grows quickly with the algorithm's multiplicative depth, which increases with the number of iterations. However, for a fixed multiplicative depth the runtime is roughly linear in both N and P (see

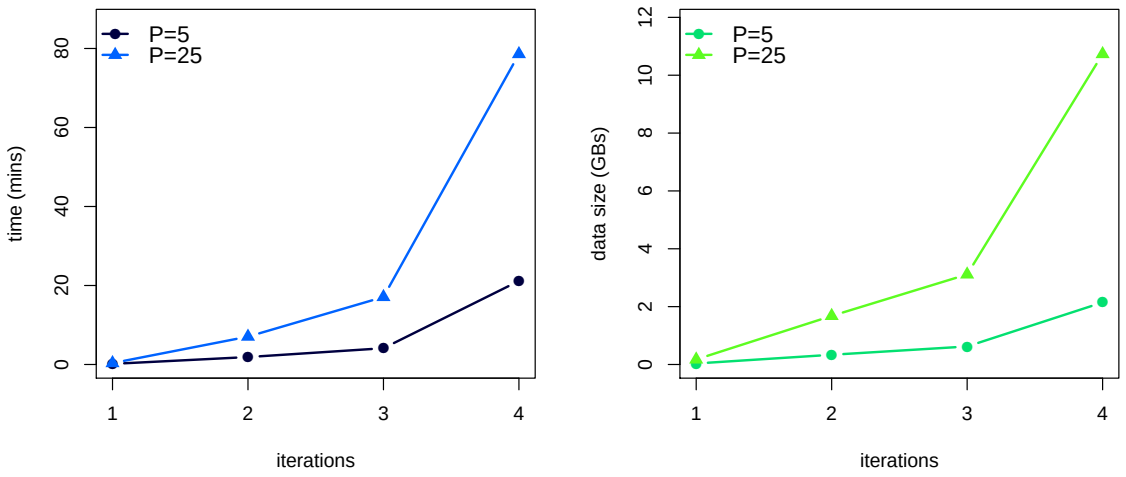


Figure 4.7: `ELS-GD` runtimes and memory requirements: **[left]** runtimes (in minutes); **[right]** theoretical data size in memory, $\{\boldsymbol{X}, \boldsymbol{y}\}$, excluding overheads (in gigabytes). [$N = 100$]

the discussion on scalability in §4.6). Memory requirements grow in a similar fashion. The theoretical ciphertext size equals the amount of memory required by d integers, each q bits long, as required to store the polynomial that is the ciphertext.

4.5.2 Applications

4.5.2.1 Mood stability data

Our first application is to a dataset containing information on mood stability in bipolar patients (Bonsall et al., 2012; Holmes et al., 2016).

Data description. The QIDS-SR_{16} (quick inventory of depressive symptomatology—self-rated) is a psychometric, self-report assessment test designed to monitor symptoms of depression as well as its severity. Patients respond to a 16-item questionnaire, choosing the answers that best describe themselves over the past week (Rush et al., 2003). The questionnaire covers the major depressive disorder symptoms from the well established DSM-IV-TR diagnostic criteria for depressive episodes such as depressed mood, loss of energy and feelings of worthlessness. Although there are 16 original questions with range 0–3, these are grouped into 9 items so that the final score has range 0–27. A higher global score implies that a patient is at a higher risk of having a depression episode. Bipolar patients answered the QIDS-SR_{16}

test and their answers were scored. Data on $N = 14$ patients were collected during a period of $T_{\text{pre}} = 28$ days before and $T_{\text{post}} = 28$ days after treatment was carried out, which results in 2 time series of length 28 for each of the 14 patients in the study (Figure 4.8).

Of interest in this application is the characterisation of the stochastic process governing the resulting time series in a privacy-preserving manner. The mood stability can be modeled as an autoregressive process, for instance an AR(2):

$$\mathbb{E}(\mathtt{y}_{i,t}) = \beta_{i1} \cdot \mathtt{y}_{i,t-1} + \beta_{i2} \cdot \mathtt{y}_{i,t-2} \tag{4.41}$$

where $\mathtt{y}_{i,t}$ denotes the test score of patient i at time t (Hamilton, 1994). The sign and magnitude of the autoregressive coefficients $\{\beta_{i1}, \beta_{i2}\}$ are important to characterise and understand the mood fluctuations in each patient i.

Figure 4.9 shows the convergence for the case of patient 8, pre-treatment and post-treatment. In both cases, convergence is achieved by both `ELS-GD-VWT` and `ELS-NAG` within 2 iterations ($||\hat{\boldsymbol{\beta}}^{[2]}_{\texttt{els-vwt}} - \hat{\boldsymbol{\beta}}_{\texttt{ols}}||_\infty = 0.039$), which takes around 12 seconds and requires under 15 MB of memory, excluding overheads (see §A.3 for hardware and software details). Runtimes and memory requirements are given in Figure 4.13.

4.5.2.2 Prostate cancer data

Our second application is on a prostate cancer dataset (Hastie et al., 2009; Stamey et al., 1989).

Data description. The dataset contains information on the level of prostate-specific antigen (`lpsa`) in men who were about to receive a radical prostatectomy, and a number of clinical measures: seminal vesicle invasion (`svi`), capsular penetration (`lcp`), percent of Gleason scores 4 or 5 (`pgg45`), Gleason score (`gleason`), cancer volume (`lcavol`), age (`age`), prostate weight (`lweight`) and benign prostatic hyperplasia (`lbph`); prefix '`l`' implies log levels. There are $N = 97$ observations and $P = 8$ predictors. Some of the covariates exhibit a high degree of correlation, making this a hard testing dataset (maximum correlation is 0.75; see Figure 4.10).

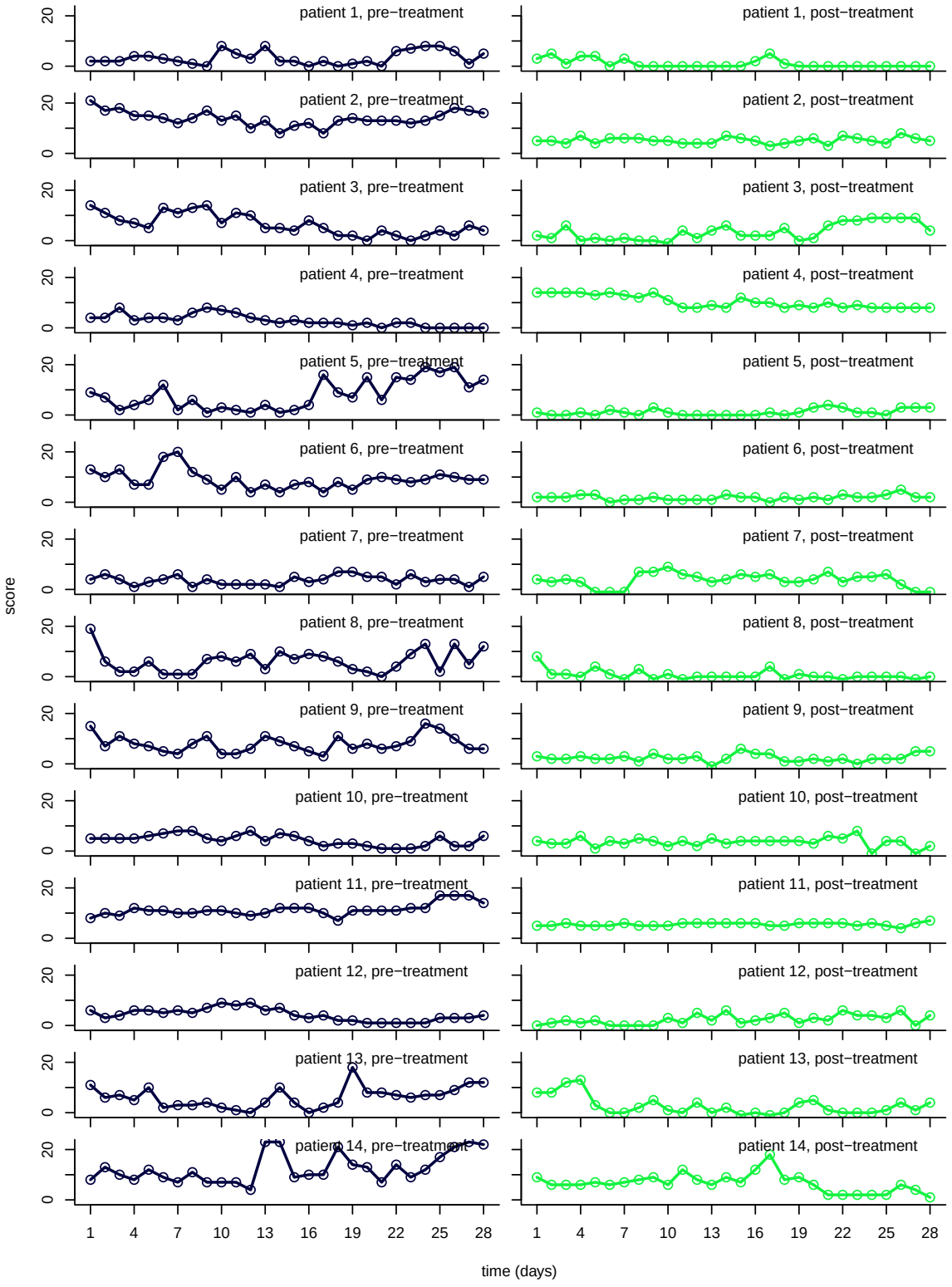


Figure 4.8: QIDS-SR$_{16}$ scores for the 14 patients in the mood stability dataset: **[left]** pre-treatment; **[right]** post-treatment. Some missing values were imputed by interpolation.

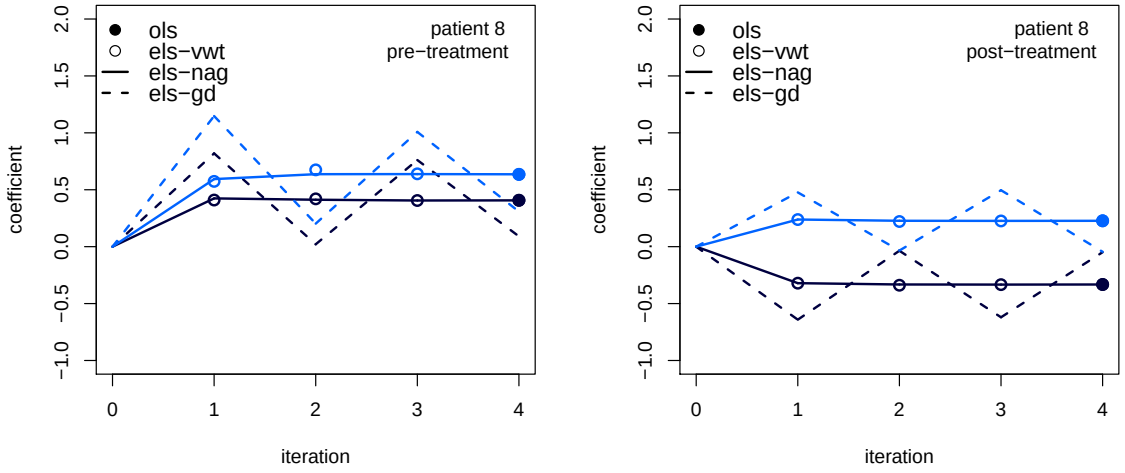


Figure 4.9: Convergence of different algorithms in the mood stability application when modeling the test scores time series of patient 8 with an AR(2) process: **[left]** pre-treatment; **[right]** post-treatment.

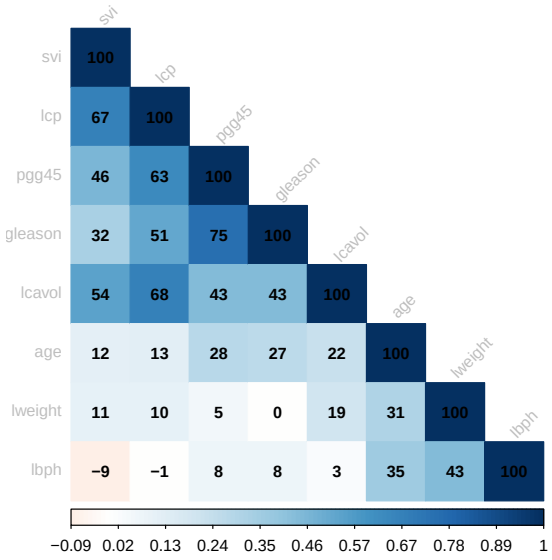


Figure 4.10: Correlation plot for the prostate cancer data. The values inside the squares are rescaled correlations with range (-100,100).

The model here is a standard linear regression,

$$\begin{aligned}\mathbb{E}(\texttt{lpsa}_i) = \beta_1 \cdot \texttt{svi}_i + \beta_2 \cdot \texttt{lcp}_i + \beta_3 \cdot \texttt{pgg45}_i + \beta_4 \cdot \texttt{gleason}_i + \beta_5 \cdot \texttt{lcavol}_i \\ + \beta_6 \cdot \texttt{age}_i + \beta_7 \cdot \texttt{lweight}_i + \beta_8 \cdot \texttt{lbph}_i.\end{aligned} \tag{4.42}$$

The convergence of different algorithms applied to the prostate cancer dataset is shown in Figure 4.11. Although not all parameters have completely converged by iteration 4 with unregularised `ELS-GD-VWT` ($||\hat{\boldsymbol{\beta}}^{[4]}_{\texttt{els-vwt}} - \hat{\boldsymbol{\beta}}_{\texttt{ols}}||_\infty = 0.267$), the fitted values are reasonably

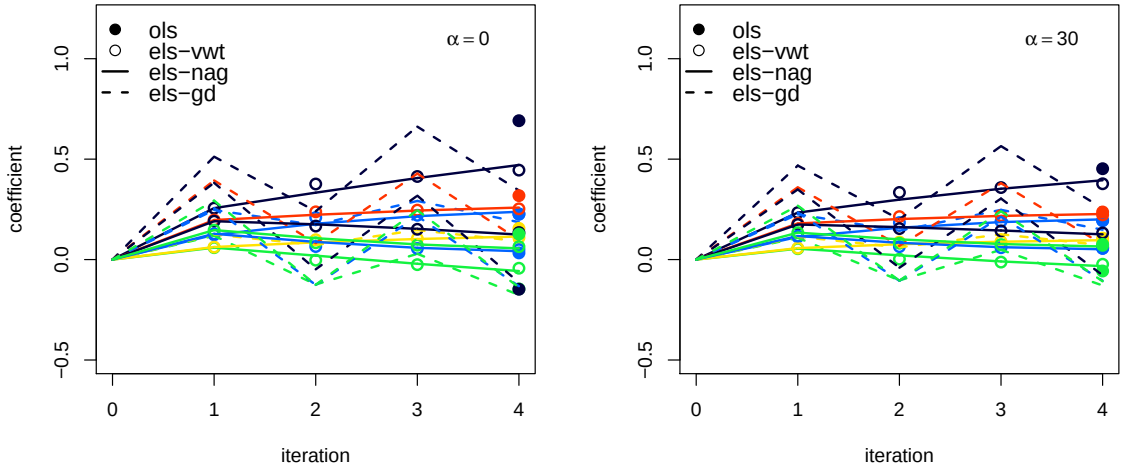


Figure 4.11: Convergence of different algorithms in the prostate data application: **[left]** without regularisation ($\alpha = 0$); **[right]** with regularisation ($\alpha = 30$).

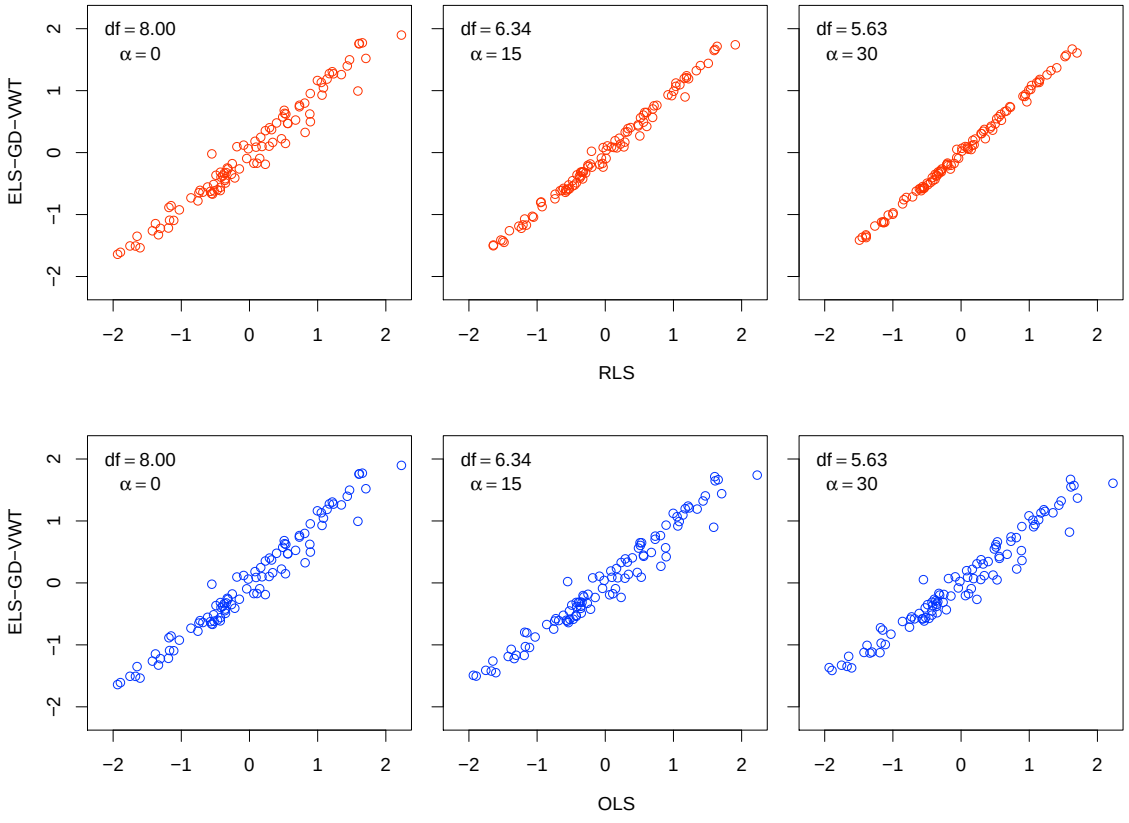


Figure 4.12: Predictions from standard `ELS-GD-VWT` ($K = 6$) with different regularisation setting, $\alpha \in \{0, 15, 30\}$, versus standard OLS and standard RLS: **[top row]** RLS versus `ELS-GD-VWT`, i.e., $\boldsymbol{X}\hat{\boldsymbol{\beta}}_{\text{rls}}$ versus $\boldsymbol{X}\hat{\boldsymbol{\beta}}_{\text{els-gd-vwt}}(\alpha)$; **[bottom row]** OLS versus `ELS-GD-VWT`, i.e., $\boldsymbol{X}\hat{\boldsymbol{\beta}}_{\text{ols}}$ versus $\boldsymbol{X}\hat{\boldsymbol{\beta}}_{\text{els-gd-vwt}}(\alpha)$.

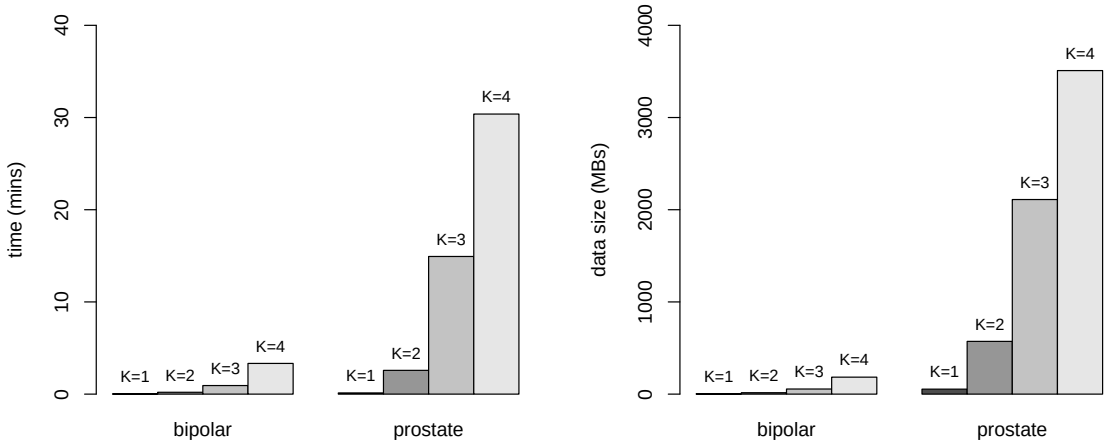


Figure 4.13: `ELS-GD` runtimes and memory requirements in the bipolar and prostate applications: **[left]** runtime (in minutes); **[right]** theoretical encrypted data size in memory, $\{\boldsymbol{X}, \boldsymbol{y}\}$ (in megabytes, excluding overheads).

close to those produced by RLS, and the two converge as the penalty term α is increased and the effective degrees of freedom

$$\text{df} = \text{trace}(\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T + \alpha\boldsymbol{I}_{PP}) \tag{4.43}$$

of the model are reduced. The effect of regularisation via parameter shrinkage translates directly into a shrinkage of the predicted values; that is, as regularisation is tightened, the effective degrees of freedom of the problem decrease and the predictions converge. More importantly, predictions from regularised `ELS-GD-VWT` do not deviate excessively from those produced by OLS (Figure 4.12). Runtimes and memory requirements are given in Figure 4.13.

4.6 Discussion

Scalability

Datasets containing highly correlated predictors constitute a real challenge (see Figure 4.4). Potential solutions include faster acceleration methods or stabilisation transformations that improve the spectral properties of $\boldsymbol{X}^T\boldsymbol{X}$. In scaling the methods to datasets with large numbers of predictors (large-P) we face similar issues (see Figure 4.6) and so solutions would,

in principle, help both cases. Datasets containing a large number of observations (large-N) present more manageable challenges, as the problems in this case are essentially computational, not methodological, that is, they require only more memory and computational power. Because all arithmetic is based on matrix operations (i.e., inner products), these can easily be split into smaller blocks of computation to fit to the amount of resources available, and parallelised. The size of the problem that we can effectively tackle depends crucially on the correlation structure of the predictors: large problems for lowly correlated predictors; small problems with highly correlated predictors. From this perspective, K can be viewed as a function of the number of predictors and, particularly, their correlation structure.

Lasso-type regularisation and variable selection

At present, L_1-norm regularisation (lasso) remains an open problem. Indeed, by their nature, variable selection techniques present exceptionally challenging problems for homomorphic computation without multi-party communication. This is because they typically require comparison operations involving encrypted elements at each step of an iterative process — and recall from §2.4.3 that comparison is out-of-reach in homomorphic computation. An entirely new approach will be needed to circumvent this issue, which is an important one since performing encrypted lasso-regularised regression would mean the ability to perform encrypted variable selection.

Step size and ridge penalty

Choosing the gradient descent step size and ridge penalisation also present real challenges. Typically, the step size is chosen as a function of the spectral radius of $\boldsymbol{X}^T\boldsymbol{X}$, while the ridge penalisation is chosen by cross-validation — both out-of-bounds under homomorphic computation. Regarding the first issue (the step size), it seems reasonable for the data holder to provide the server with an approximation to the spectral radius of the problem, $\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$, alongside the dataset, $\{\boldsymbol{X}, \boldsymbol{y}\}$. In fact, an upper bound exists for this quantity: $\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X}) \leq ||(\boldsymbol{X}^T\boldsymbol{X})^k||^{1/k} \equiv \mathcal{B}(k)$ for $k \in \mathbb{N}$. This requires a small amount of computation and, additionally, the sequence $\{\mathcal{B}(k)\}_{k\geq 1}$ is decreasing and converges to $\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$, meaning that we can obtain arbitrarily precise estimates of the spectral radius. In principle, this would

leak minimal information about the individual data records. Regarding the second issue (the penalty term), the standard approach — selecting the ridge penalty via cross-validation — cannot be implemented homomorphically in a straightforward way. A possible solution is to allow rounds of communication during which the key holder validates the model. In the next chapter we propose a framework of this type to solve a different problem, although it can arguably be adapted to provide a solution to the two problems just discussed. For higher security at all stages of the process one may wish to perturb the intermediate values returned in a differentially private manner to ensure that no sensitive information is leaked to this additional party (Wang et al., 2015b; Zhang et al., 2012; see §2.5.2).

Weighted least squares

One natural extension with important applications is weighted least squares (WLS) regression. This would have important applications in clinical decision support and would allow personalised predictions to be made for each patient at the cost of fitting a regression each time. The challenge here is to find a weight a function which can be implemented homomorphically and has good predictive performance relative to OLS.

4.7 Summary

In this chapter we presented algorithms for privacy-preserving regularised linear regression in the context of fully homomorphic encryption that allow both estimation and prediction to be carried out in the encrypted domain. Unlike other related methods in the literature, we give extensive details on the statistical and computational properties of these algorithms as relevant to their use in homomorphic computation.

Standard methods for solving the least squares problem require potentially large matrix inversions or decompositions, which fall outside of the current possibilities of FHE. We compare iterative methods, namely coordinate descent (CD) and gradient descent (GD) as alternatives to traditional methods and describe the computational costs of implementing each of these homomorphically. Analysis of the maximum multiplicative depth (MMD) — a key determinant of the efficiency of an algorithm in this framework — reveals that GD scales linearly

with the number of iterations K and is independent of the sample size N and the number of predictors P (MMD $= 2K$), while CD scales linearly in the number of iterations and the number of predictors P (MMD $= 2KP$). GD is, therefore, favoured. It should be noted that GD also depends on the number of predictors, although only indirectly. This is because a larger number of iterations will be typically required for convergence to be achieved in larger problems. We can, therefore, think of K as being positively dependent on P or, more precisely, on the correlation structure among predictors. We show that the net effect is still that GD is more efficient than CD for a fixed MMD, i.e., achieves lower estimation error.

Because large-depth algorithms are extremely expensive to implement homomorphically, we study several acceleration techniques and propose the use of the van Wijngaarden transformation (VWT), a simple algorithm to accelerate convergence that takes advantage of the oscillatory nature of the coefficient profiles produced by GD. VWT is extremely cheap computationally, increasing MMD by only 1. Although NAG is typically considered the state-of-the-art acceleration method, in the computationally restricted framework of FHE the VWT is a more attractive option due to its lower error per computational cost unit, as measured by the MMD.

Regression under homomorphic computation remains a difficult problem. Both traditional and SMPC approaches have serious limitation, implying restricted problem sizes that they can tackle (Hall et al., 2011; Wu and Haven, 2012). The acceleration methods proposed in this chapter constitute significant improvement over existing methods, although we are still unable to tackle large-P problems or problems with highly correlated predictors.

Chapter 5

Cooperative Bayesian regression with distributed data

In this chapter we propose a privacy-preserving computational framework for cooperative Bayesian linear regression with horizontally-partitioned, distributed data. In this framework, a number of parties cooperate to compute posterior estimates of regression coefficients incorporating information from all the data available (i.e., from all parties) but without sharing their specific blocks of data with the remaining parties. Suppose parties $s \in \mathbb{N}_{0:S}$ each own a horizontally partitioned subset $\boldsymbol{D}_{(s)}$ of the total data available, $\boldsymbol{D} = \{\boldsymbol{D}_{(s)}\}_{s \in \mathbb{N}_{0:S}}$, physically located on a different and potentially untrusted site (e.g., cloud server). The goal is to cooperatively compute a statistic $f(\boldsymbol{D})$ without leaking information that could be used to identify individual records and without parties $\mathbb{N}_{0:S} \backslash s$ learning about $\boldsymbol{D}_{(s)}$.

Related work. The Bayesian regression model has received comparatively little attention in the privacy-preserving methods literature. To our knowledge, the only satisfactory treatment of the problem is provided by Ghosh and Reiter (2013). The authors use secure multi-party computation (SMPC) for Bayesian model averaging with horizontally partitioned data where each party s computes a partial model, $\mathcal{M}_{(s)}$, using only their portion of the data, $\boldsymbol{D}_{(s)}$. Using Zellner type priors, they obtain a closed form expression for the marginal likelihood $\mathbb{P}(\boldsymbol{y}|\boldsymbol{X}, \mathcal{M}_{(s)})$ and use SMPC protocols with rounds of communication between parties to compute the global quantities of interest, i.e., those that depend on the entire dataset $\boldsymbol{D}$.

A few limitations of this approach are noteworthy: first, computation cannot be securely outsourced to the cloud since there is no layer of security at the level of the data, and so it must remain stored in a safe location; second, all parties are required to interact in rounds of communication whenever a new block $\boldsymbol{D}_{(s_\star)}$ is added to the pool of data and a new model $\mathcal{M}_{(s_\star)}$ is estimated. The approach proposed in this chapter is able to overcome both of these limitations. In a similar framework, Wu et al. (2016) used additively HE and SMPC protocols for two parties to evaluate decision trees, where the server owns the model and learns nothing about the data owned by the client, who learns only the result and some parameters of the model. In this case, the model is fitted offline (i.e., not securely in the cloud) and only prediction is provided in a secure fashion.

Roadmap. We begin by giving details on the data representation and encoding necessary for encrypted regression analysis (§5.1) and describing the Bayesian linear regression framework in the special case of fully conjugate priors (§5.2); next, we describe the proposed computational framework in detail, including the multi-party and modelling aspects (§5.3); after this, we present approximate methods for Bayesian linear regression estimation designed to circumvent the restrictions of FHE in the context of the computational framework proposed (§5.4); we then show some results illustrating the potential and limitations of the methods developed (§5.5); and finally, discuss (§5.6) and summarise (§5.7) our findings.

5.1 Data representation and encoding

Following the previous chapters, we encode real-valued data by using the approximate expansion methods, so as to comply with the restrictions of FHE, where only integers can be encrypted. Briefly here, we represent a value $z \in \mathbb{R}$ as an integer by applying the transformation $z \mapsto \lfloor 10^\phi z \rceil$, where ϕ denotes the accuracy desired (number of decimal places) and $\lfloor \cdot \rceil$ denotes rounding. For more details see §3.1.2.

5.2 The Bayesian linear regression model

Let $\boldsymbol{X}$ denote a $N \times P$ design matrix; $\boldsymbol{y}$ denote a $N \times 1$ response vector; and $\boldsymbol{\beta}$ denote a $P \times 1$ parameter vector. N and P represent the numbers of observations and predictors, respectively. We can write the Bayesian linear regression model as follows

$$\boldsymbol{y}|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{I}_{NN}) \tag{5.1}$$

$$(\boldsymbol{\beta}, \sigma^2) \sim p(\boldsymbol{\beta}, \sigma^2) \tag{5.2}$$

that is, it is assumed that the responses are independent and Normally distributed around the linear predictor $\boldsymbol{X}\boldsymbol{\beta}$, with constant variance σ^2; and a prior is also assumed for the unknown parameters $(\boldsymbol{\beta}, \sigma^2)$ given by a parametric distribution p. We choose a fully conjugate prior distribution for the likelihood. In the case of the Normal likelihood, this corresponds to the Normal-Inverse-Gamma distribution (O'Hagan and Forster, 2004, §11.10):

$$p(\boldsymbol{\beta}, \sigma^2) \propto (\sigma^2)^{-(b+P+2)/2} \exp\{-[(\boldsymbol{\beta} - \boldsymbol{m})^T \boldsymbol{V}^{-1} (\boldsymbol{\beta} - \boldsymbol{m}) + a]/(2\sigma^2)\} \tag{5.3}$$

implying

$$(\boldsymbol{\beta}, \sigma^2) \sim \mathcal{NG}^{-1}(a, b, \boldsymbol{m}, \boldsymbol{V}). \tag{5.4}$$

This can be decomposed as $p(\boldsymbol{\beta}, \sigma^2) = p(\boldsymbol{\beta}|\sigma^2)p(\sigma^2)$, giving a Gamma distribution for the error variance and a Normal distribution for the regression coefficients (conditional on the error variance):

$$p(\sigma^2) \propto (\sigma^2)^{-(b+2)/2} \exp\{-a/(2\sigma^2)\} \tag{5.5}$$

$$p(\boldsymbol{\beta}|\sigma^2) \propto (\sigma^2)^{-P/2} \exp\{-(\boldsymbol{\beta} - \boldsymbol{m})^T \boldsymbol{V}^{-1} (\boldsymbol{\beta} - \boldsymbol{m})/(2\sigma^2)\} \tag{5.6}$$

implying

$$\sigma^2 \sim \mathcal{G}^{-1}(a, b) \qquad \text{and} \qquad \boldsymbol{\beta}|\sigma^2 \sim \mathcal{N}(\boldsymbol{m}, \sigma^2 \boldsymbol{V}) \tag{5.7}$$

where $\boldsymbol{m}$ and $\boldsymbol{V}$ are location and dispersion parameters controlling the prior mean and variance of $\boldsymbol{\beta}$; and a and b are shape and rate parameters of the inverse gamma distribution. From a Bayesian modeling perspective these are hyperparameters.

Combining the likelihood with the prior via Bayes' rule gives the posterior distribution of the unknown parameters:

$$p(\boldsymbol{\beta}, \sigma^2; \boldsymbol{y}, \boldsymbol{X}) \propto (\sigma^2)^{-(b+N+P+2)/2} \exp\{-Q/2\sigma^2\} \tag{5.8}$$

where

$$\begin{aligned} Q &= \boldsymbol{\beta}^T(\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{V}^{-1})\boldsymbol{\beta} - \boldsymbol{\beta}^T(\boldsymbol{V}^{-1}\boldsymbol{m} + \boldsymbol{X}^T\boldsymbol{y}) \\ &\quad - (\boldsymbol{m}^T\boldsymbol{V}^{-1} + \boldsymbol{y}^T\boldsymbol{X})\boldsymbol{\beta} + \boldsymbol{m}^T\boldsymbol{V}^{-1}\boldsymbol{m} + \boldsymbol{y}^T\boldsymbol{y} + a \\ &= (\boldsymbol{\beta} - \boldsymbol{m}_\star)^T\boldsymbol{V}_\star^{-1}(\boldsymbol{\beta} - \boldsymbol{m}_\star) + a_\star \end{aligned} \tag{5.9}$$

implying

$$(\boldsymbol{\beta}, \sigma^2)|\boldsymbol{y}, \boldsymbol{X} \sim \mathcal{NG}^{-1}(a_\star, b_\star, \boldsymbol{m}_\star, \boldsymbol{V}_\star). \tag{5.10}$$

The posterior hyperparameters can then be read directly from (5.8) and (5.9):

$$\boldsymbol{V}_\star = (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{V}^{-1})^{-1} \tag{5.11a}$$

$$\boldsymbol{m}_\star = \boldsymbol{V}_\star(\boldsymbol{V}^{-1}\boldsymbol{m} + \boldsymbol{X}^T\boldsymbol{y}) \tag{5.11b}$$

$$a_\star = a + \boldsymbol{m}^T\boldsymbol{V}^{-1}\boldsymbol{m} + \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{m}_\star^T\boldsymbol{V}_\star^{-1}\boldsymbol{m}_\star \tag{5.11c}$$

$$b_\star = b + N \tag{5.11d}$$

which gives a straightforward rule to update the hyperparameters defining the distribution of the posterior mean and variance of the regression coefficients: $\mathbb{E}[\boldsymbol{\beta}] = \boldsymbol{m}_\star$ and $\mathbb{V}[\boldsymbol{\beta}] = a_\star\boldsymbol{V}_\star/(b_\star - 2)$.

5.3 Computational framework

Our goal is to provide a computational framework for secure computation with distributed data — i.e., where each of several parties holds a portion of the data that it is not willing to share with other parties. The framework embodies a number of characteristics, namely distributed and horizontally partitioned data (§5.3.1); a multi-party computational model (§5.3.2); and recursive Bayesian updating (§5.3.3). A diagram of the framework proposed will be shown further ahead after we define it in detail (Figure 5.3).

5.3.1 Distributed and horizontally partitioned data

Let $s \in \mathbb{N}_{0:S}$ denote the party index and $\boldsymbol{D}_{(s)} = \{\boldsymbol{X}_{(s)}, \boldsymbol{y}_{(s)}\}$ denote the portion (or block) of the total data $\boldsymbol{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ held by party s. We consider the case of horizontally partitioned data where each party has a subset N_s of the N observations on the same set of P covariates:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_{(0)} \\ \boldsymbol{y}_{(1)} \\ \boldsymbol{y}_{(2)} \\ \vdots \end{pmatrix}, \qquad \boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}_{(0)} \\ \boldsymbol{X}_{(1)} \\ \boldsymbol{X}_{(2)} \\ \vdots \end{pmatrix} \tag{5.12}$$

that is, $\boldsymbol{y}_{(s)}$ is of size $N_s \times 1$ and $\boldsymbol{X}_{(s)}$ is of size $N_s \times P$, with $\sum_{s=0}^{S} N_s = N$.

5.3.2 Multiple parties

The central figure in our computational framework is the *master*, or party 0, who plays a double role. In addition to (optionally) contributing his data $\boldsymbol{D}_{(0)} = \{\boldsymbol{X}_{(0)}, \boldsymbol{y}_{(0)}\}$, the master remains an active party in computation throughout the entire cooperative effort and it is assumed that it can perform both homomorphic and non-homomorphic computation on its own data, $\boldsymbol{D}_{(0)}$, and on any additional statistics it receives from the remaining parties during the cooperative process. The master also initialises the encryption scheme and holds the secret key, and so it can be viewed as a managing party. All remaining parties, or *clients*, may have access to the public key but not to the secret key, which means that they cannot decrypt the statistics they receive from other parties, and that they can only perform homomorphic computation. For example, in a biomedical setting, the role of the master could be played by a central analyst (e.g., research institute) while the clients could be data providers (e.g., hospitals).

5.3.3 Recursive Bayesian updating

Suppose there are a total of $S + 1$ cooperating parties. Each party $s \in \mathbb{N}_{0:S}$ holds a block of the data, $\boldsymbol{D}_{(s)}$, which it is not willing to share with the remaining parties. To achieve

our goal we will cooperatively compute a sequence of posterior hyperparameter updates, $\boldsymbol{h}_{(s)} = \{\boldsymbol{m}_{(s)}, \boldsymbol{V}_{(s)}\}$, where $\boldsymbol{h}_{(s)}$ represents the set of posterior hyperparameters of interest incorporating all data from parties 0 to s, that is, $\{\boldsymbol{D}_{(r)}\}_{r\in\mathbb{N}_{0:s}}$.[1] Bayesian modelling is, therefore, especially suitable to this cooperative scenario because of the consistency property of recursive Bayesian updating, which allows updates to the posterior distribution to be done sequentially in a straightforward manner for independent and identically distributed data (O'Hagan and Forster, 2004, §3.5). Figure 5.1 shows a diagram of this process.

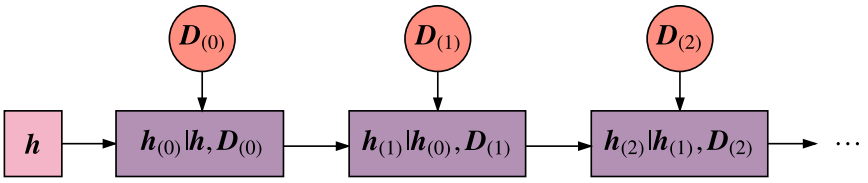


Figure 5.1: Cooperation via recursive Bayesian updating. At each cooperation step s, the hyperparameters $\boldsymbol{h}_{(s)} = \{\boldsymbol{m}_{(s)}, \boldsymbol{V}_{(s)}\}$ are updated by party s to reflect the information in its portion of the data, $\boldsymbol{D}_{(s)} = \{\boldsymbol{X}_{(s)}, \boldsymbol{y}_{(s)}\}$, and then passed along to the next party. The prior, held by the master party, is represented by $\boldsymbol{h}$.

Focusing on the main hyperparameters of interest, mean and variance, at each cooperation step $s \in \mathbb{N}_{0:S}$ they are updated according to (5.11) as follows:

$$\boldsymbol{V}_{(s)} \leftarrow \left(\boldsymbol{X}_{(s)}^T \boldsymbol{X}_{(s)} + \boldsymbol{V}_{(s-1)}^{-1}\right)^{-1} \tag{5.13a}$$

$$\boldsymbol{m}_{(s)} \leftarrow \boldsymbol{V}_{(s)} \left(\boldsymbol{V}_{(s-1)}^{-1} \boldsymbol{m}_{(s-1)} + \boldsymbol{X}_{(s)}^T \boldsymbol{y}_{(s)}\right) \tag{5.13b}$$

where $\boldsymbol{V}_{(-1)} \equiv \boldsymbol{V}$ and $\boldsymbol{m}_{(-1)} \equiv \boldsymbol{m}$ denote the priors. The objective is, therefore, for party s to compute an updated set of hyperparameters, $\boldsymbol{h}_{(s)}$, using its own data $\boldsymbol{D}_{(s)}$, after receiving the hyperparameters from the previous party, $\boldsymbol{h}_{(s-1)}$.

Notably, the recursive Bayesian updating framework is also ideal in that long-term memory requirements of encrypted data are minimal. Because $\boldsymbol{m}_{(s)}$ and $\boldsymbol{V}_{(s)}$ include all the necessary information for future updates, once the update s is completed the data $\boldsymbol{D}_{(s)}$ can be discarded. For long-term storage of data we can use a non-homomorphic encryption scheme like RSA, which is much more efficient for this purpose.

[1] We will focus on the mean and variance hyperparameters and define $\boldsymbol{h}_{(s)} = \{\boldsymbol{m}_{(s)}, \boldsymbol{V}_{(s)}\}$, although this can naturally be expanded to include $a_{(s)}$, $b_{(s)}$ as well.

5.4 Approximate recursive Bayesian updating

Although straightforward in principle, the computational framework for secure, cooperative Bayesian linear regression presented in the previous section hides a number of difficulties arising from the restrictions that are the trademark of homomorphic computation (see §2.4.3).

Consider the posterior updates in (5.13). Immediately, the inversion of a possibly large matrix required to update the posterior variance in (5.13a) should be acknowledged as challenging, if not altogether impossible. Moreover, updating the posterior mean in (5.13b) poses similar problems. In this section we propose approximate methods to overcome both of these obstacles.

5.4.1 Estimating the posterior variance

Our strategy takes advantage of the special role played by the master in the proposed computational framework. At each cooperation step s, the posterior variance must be updated in agreement with (5.13a), where $\boldsymbol{X}_{(s)}$ is the block of data owned by party s and $\boldsymbol{V}_{(s)}$ is the posterior variance after cooperation step s. Under the computational framework proposed (§5.3), the master party computes this quantity non-homomorphically, since it requires matrix inversion — which it can do by assumption. However, for security reasons, clients cannot share $\boldsymbol{X}_{(s)}$ with the master in unencrypted format and, therefore, $\{\boldsymbol{V}_{(s)}\}_{s>0}$ must be estimated. We propose that the master party uses $\boldsymbol{X}_{(0)}$ as a proxy for $\boldsymbol{X}_{(s)}$ when estimating the posterior:

$$\boldsymbol{V'}_{(s)} \equiv \left(\boldsymbol{X}_{(0)}^T \boldsymbol{X}_{(0)} + \boldsymbol{V}_{(s-1)}^{-1}\right)^{-1} \approx \boldsymbol{V}_{(s)}. \tag{5.14}$$

This is a valid approach if we assume that the two blocks of data, $\boldsymbol{X}_{(0)}$ and $\boldsymbol{X}_{(s)}$, are identically distributed, which will be reasonable assumption to make in most scenarios.

Instead of simply using $\boldsymbol{X}_{(0)}$, we can average the result of $\boldsymbol{V'}_{(s)}$ over multiple bootstrap samples of $\boldsymbol{X}_{(0)}$. This also gives us a natural measure of the variability in these estimates. Additionally, when parties hold blocks of different sizes, N_s, bootstrapping provides a simple and effective way to estimate the correct variability in $\boldsymbol{X}_{(s)}^T \boldsymbol{X}_{(s)}$ by using an appropriate subsample size, N_s.

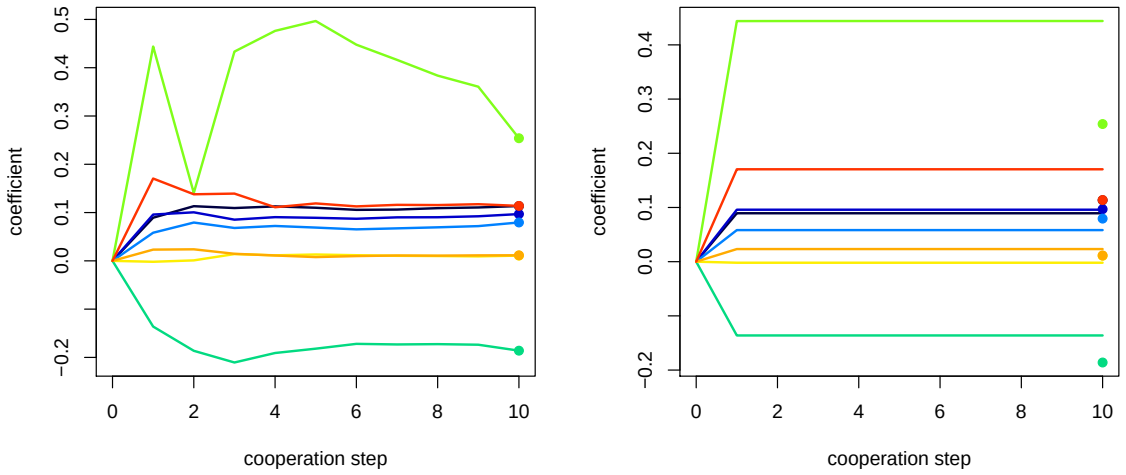


Figure 5.2: Recursive Bayesian updating via (5.13): **[left]** exact variance update using $\boldsymbol{D}_{(s)}$ at iteration s; **[right]** approximate variance update using $\boldsymbol{D}_{(0)}$ at all iterations. The lines represent coefficient profiles and the dots represent the targets [`concrete` dataset: $N = 1000, N_s = 100, P = 8$]

5.4.2 Estimating the posterior mean

Given the approximation to the posterior variance in (5.14), computed by the master, one could implement the exact posterior mean update as in (5.13b) by having the master compute, encrypt and send both $\boldsymbol{V'}_{(s)}$ and $\boldsymbol{V'}^{-1}_{(s-1)}$ to party s. However, because we are estimating the posterior variance, this approach would target the wrong quantity from the outset. In fact, errors can be extremely large, up to 100% of the coefficients' magnitude (Figure 5.2). Here, we propose an iterative approach that outperforms the exact update approach when the posterior variances are approximate.

Newton method. Newton's method for maximising $\log(p(\boldsymbol{\beta}, \sigma^2; y))$ w.r.t. $\boldsymbol{\beta}$ in (5.8) can be written as

$$\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} + J^{-1}(\boldsymbol{\beta}^{[k-1]}) \cdot S(\boldsymbol{\beta}^{[k-1]}) \tag{5.15}$$

where $S(\cdot)$ and $J(\cdot)$ represent the gradient and the observed information, which for the Bayesian problem specified earlier are given respectively by

$$\begin{aligned} S(\boldsymbol{\beta}) &= \frac{\partial \log(p(\boldsymbol{\beta}, \sigma^2; y))}{\partial \boldsymbol{\beta}} = -\frac{\partial Q}{\partial \boldsymbol{\beta}} \\ &= -2(\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{V}^{-1})\boldsymbol{\beta} + 2(\boldsymbol{V}^{-1}\boldsymbol{m} + \boldsymbol{X}^T\boldsymbol{y}) \\ &= 2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + 2\boldsymbol{V}^{-1}(\boldsymbol{m} - \boldsymbol{\beta}) \end{aligned} \tag{5.16}$$

and

$$\begin{aligned} J(\boldsymbol{\beta}) &= -\frac{\partial^2 \log(p(\boldsymbol{\beta}, \sigma^2; y))}{\partial \boldsymbol{\beta}^2} = \frac{\partial^2 Q}{\partial \boldsymbol{\beta}^2} \\ &= 2(\boldsymbol{X}^T \boldsymbol{X} + \boldsymbol{V}^{-1}) \end{aligned} \tag{5.17}$$

yielding the following iterative equation for cooperation step s:

$$\begin{aligned} \boldsymbol{\beta}^{[k]} &= \boldsymbol{\beta}^{[k-1]} + \underbrace{(\boldsymbol{X}^T \boldsymbol{X} + \boldsymbol{V}_{(s)}^{-1})^{-1}}_{\boldsymbol{V}_{(s+1)}} [\boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) + \boldsymbol{V}_{(s)}^{-1} (\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[k-1]})] \\ &= \boldsymbol{\beta}^{[k-1]} + \boldsymbol{V}_{(s+1)} [\boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) + \boldsymbol{V}_{(s)}^{-1} (\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[k-1]})] \end{aligned} \tag{5.18}$$

In the present Bayesian setting, optimising $\boldsymbol{\beta}$ via the Newton method gives the maximum a posteriori (MAP) estimate which, by the symmetry of the Normal distribution, coincides with the posterior mean update of (5.13b). The posterior mean can then be approximated (as $\boldsymbol{V}$ is approximate) by the value of $\boldsymbol{\beta}$ at iteration K:

$$\boldsymbol{m'}_{(s)} \equiv \boldsymbol{\beta}^{[K]} \approx \boldsymbol{m}_{(s)}. \tag{5.19}$$

Quasi-Newton method. Because $\log(p(\boldsymbol{\beta}, \sigma^2; y))$ is quadratic, Newton's method converges in one step. However, in the cooperative framework proposed above, parties are not allowed to share their data, $\boldsymbol{D}_{(s)}$, which means that we cannot compute $J(\boldsymbol{\beta})$ exactly. Using the ideas behind the estimation of the posterior variance (§5.4.1), we propose a quasi-Newton approach that replaces the information matrix $J(\boldsymbol{\beta})$ with an approximation computed by the master using only its data, $\boldsymbol{D}_{(0)}$:

$$J'(\boldsymbol{\beta}) \equiv 2(\boldsymbol{X}_{(0)}^T \boldsymbol{X}_{(0)} + \boldsymbol{V'}^{-1}) \approx J(\boldsymbol{\beta}). \tag{5.20}$$

Under regularity conditions, the quasi-Newton method converges to the same value as the Newton method since $J(\boldsymbol{\beta})$ plays the role of a step size (analogous to δ in GD; §4.3.1.2), and a valid approximation impacts only the coefficient profiles, not the steady-state of the iterative process.

Overall, the method proposed in this section — namely the approximations to the variance (§5.4.1) and mean (§5.4.2) — incurs two approximations: one in the gradient vector $S(\boldsymbol{\beta})$,

resulting from the approximation of $\boldsymbol{V}$ in (5.14); and one in the information matrix $J(\boldsymbol{\beta})$. While the first has an effect on the steady-state of the process, the second does not, as it only affects the step size of the quasi-Newton method. In contrast, when the posterior variance is approximated, the exact update equation for the posterior mean (5.13b) contains two approximate values, and both contribute to a degradation in estimation accuracy (Figure 5.2).

Transformed equations. Because homomorphic computation requires the data to be represented as integers (see §5.1), some equations must be appropriately rescaled to avoid divisions. In the case of cooperative Bayesian linear regression, the quasi-Newton equations must be modified as follows:

$$\begin{aligned}\tilde{\boldsymbol{\beta}}^{[k]} &\equiv 10^{3\phi}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{V}}'_{(s+1)}[\tilde{\boldsymbol{X}}^T_{(s)}(10^{(3k-2)\phi}\tilde{\boldsymbol{y}}_{(s)} - \tilde{\boldsymbol{X}}_{(s)}\tilde{\boldsymbol{\beta}}^{[k-1]}) \\ &\qquad + 10^{\phi}\tilde{\boldsymbol{V}}'^{-1}_{(s)}(10^{3(k-1)\phi}\tilde{\boldsymbol{m}}'_{(s-1)} - \tilde{\boldsymbol{\beta}}^{[k-1]})] \\ &\equiv 10^{(3k+1)\phi}\boldsymbol{\beta}^{[k+1]}\end{aligned} \tag{5.21}$$

for $k \in \mathbb{N}_{1:K}$ and where the tildes represent rescaled data such that $\tilde{\boldsymbol{\beta}}^{[0]} \equiv 10^{\phi}\boldsymbol{\beta}^{[0]}$ and similarly for the remaining variables, except $\{\tilde{\boldsymbol{\beta}}^{[k]}\}$. The method for obtaining these equations is similar to that used in the previous chapter for gradient descent, (4.20); see §B.5 for details. Each party s repeats this process for K iterations using the transformed inputs, $\{\tilde{\boldsymbol{X}}_{(s)}, \tilde{\boldsymbol{y}}_{(s)}, \tilde{\boldsymbol{m}}'_{(s-1)}, \tilde{\boldsymbol{V}}'_{(s+1)}, \tilde{\boldsymbol{V}}'^{-1}_{(s)}\}$. The regression coefficients can then be easily retrieved by the secret key holder, requiring only a simple rescaling of the transformed coefficients: $\boldsymbol{\beta}^{[K]} = \mathtt{Dec}_{\mathtt{sk}}(\tilde{\boldsymbol{\beta}}^{[K]})/10^{(3K+4)\phi}$.

5.4.3 Cooperative Bayesian linear regression

Having described the approximations necessary to overcome the difficulties imposed by FHE, we now describe the computational framework in full detail.

In the first phase — the initialisation phase — the master (party 0) initialises the encryption scheme by setting the cryptographic parameters and generating the keys. The master also sets the prior mean $\boldsymbol{m}_{(-1)} = \boldsymbol{m}$ and the prior variance $\boldsymbol{V}_{(-1)} = \boldsymbol{V}$. Next, it proceeds to update the hyperparameters to obtain $\{\boldsymbol{m}_{(0)}, \boldsymbol{V}_{(0)}\}$ via (5.13), using only its own data,

$\boldsymbol{D}_{(0)} = \{\boldsymbol{X}_{(0)}, \boldsymbol{y}_{(0)}\}$. This stage is performed locally and non-homomorphically, following the assumption that the master is an honest-but-curious party.

The second phase of the algorithm — the cooperative phase — consists of alternating rounds between the master and each of the remaining parties $s \in \mathbb{N}_{1:S}$ as follows:

(i) the master updates the posterior variances approximately as in (5.14) to obtain $\boldsymbol{V'}^{-1}_{(s)}$ and $\boldsymbol{V'}_{(s+1)}$, using only its data, $\boldsymbol{D}_{(0)}$, and the hyperparameters from the previous cooperation step;

(ii) the master then transforms, encrypts and sends the current hyperparameters to party s, namely $\{\boldsymbol{m'}_{(s-1)}, \boldsymbol{V'}^{-1}_{(s)}, \boldsymbol{V'}_{(s+1)}\}$;

(iii) party s homomorphically updates the posterior mean as in (5.21) to obtain $\boldsymbol{m'}_{(s)}$, using the hyperparameters received from the master together with its own data, $\boldsymbol{D}_{(s)}$. Party s then returns $\boldsymbol{m'}_{(s)}$ to the master;

(iv) the master re-encrypts $\boldsymbol{m'}_{(s)}$ to reduce ciphertext noise and proceeds to *(i)* to begin the cooperative step $s + 1$.

A diagram of this cooperative framework is shown in Figure 5.3. In short, the master approximates the posterior variance while clients approximate the posterior mean (Algorithm 5.1). Crucially, parties do not see other parties' individual data in the process, thus guaranteeing privacy.

5.4.4 Data standardisation

We typically assume that the data is standardised. In the present framework, however, because the data is distributed between different parties who would not like to share their data with the remaining parties, standardising the data is a problem because we do not have access to the overall standardisation factors (SFs): response mean, predictor mean, and predictor standard deviation (for all predictors).

A first option is to use a cooperative process (e.g., SMPC), prior to running the estimation step, where we would obtain the SFs based on the entire dataset, $\boldsymbol{D} = \{\boldsymbol{D}_{(s)}\}_{s \in \mathbb{N}_{0:S}}$. To

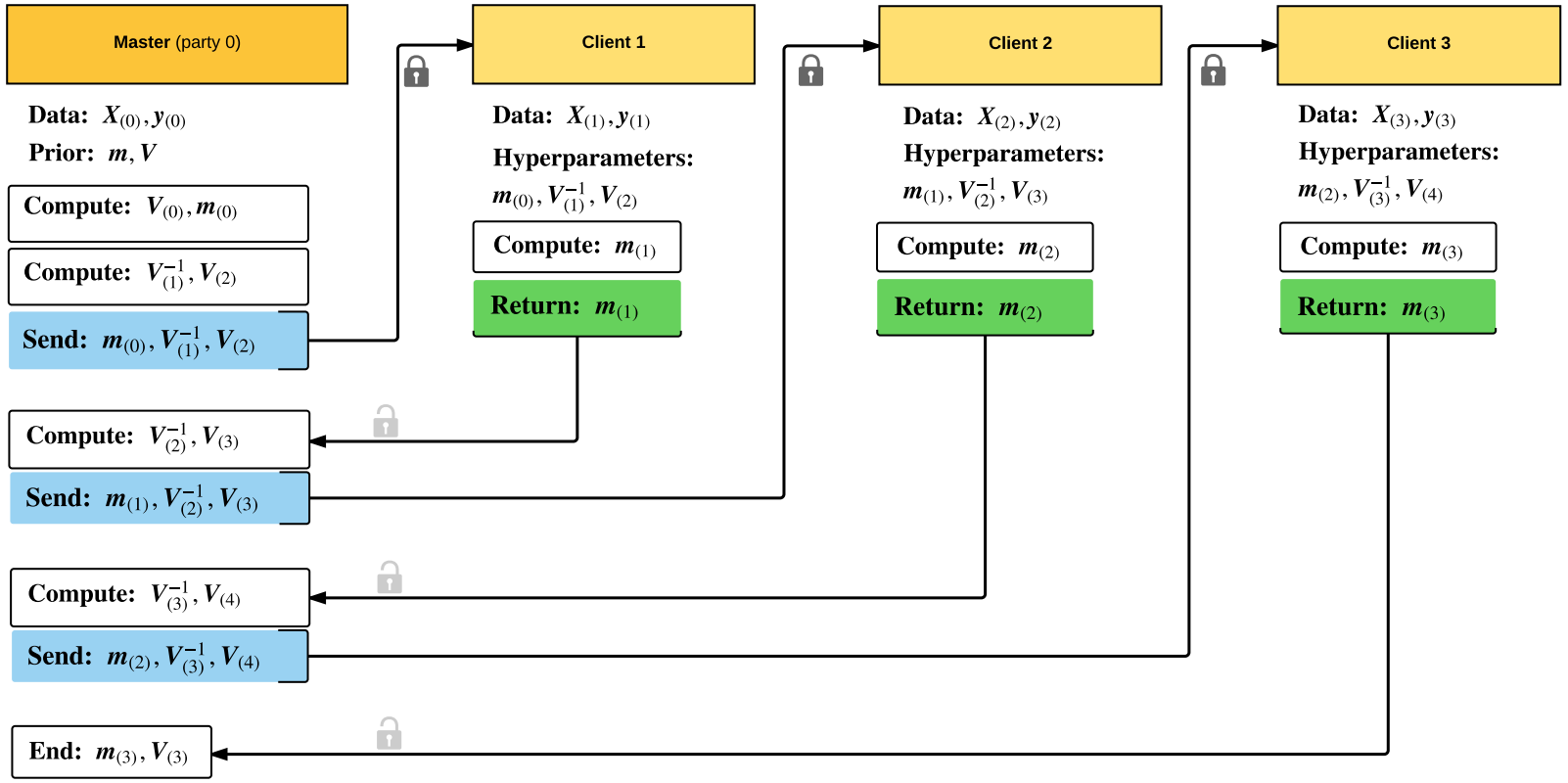


Figure 5.3: Diagram of the cooperative framework for Bayesian linear regression. **Legend:** dark grey, closed locks represent encryption; light grey, open locks represent decryption; blue boxes represent master→client communication; green boxes represent client→master communication. **Description:** at each cooperation step s, the master computes $\boldsymbol{V}_{(s)}^{-1}$ and $\boldsymbol{V}_{(s+1)}$, encrypts and sends these, together with $\boldsymbol{m}_{(s-1)}$, to party s (at step $s = 0$ the master also updates $\boldsymbol{m}_{(0)}$). Party s then computes $\boldsymbol{m}_{(s)}$ homomorphically and returns the result to the master.

Algorithm 5.1 `ECBLR-QN`: encrypted cooperative Bayesian linear regression via quasi-Newton iterations

Input: data $\{\boldsymbol{X}_{(0)}, \boldsymbol{y}_{(0)}, \tilde{\boldsymbol{X}}_{(s)}, \tilde{\boldsymbol{y}}_{(s)}\}_{s\in\mathbb{N}_{1:S}}$, priors $\{\boldsymbol{m}, \boldsymbol{V}\}$, parameters $\{S, K\}$
for $s = 0$ **to** S **do**
  if s = 0 **then**
    MASTER:
    • compute $\boldsymbol{V}_{(0)}$ and $\boldsymbol{m}_{(0)}$ according to (5.13), using $\boldsymbol{V}_{(-1)} = \boldsymbol{V}$ and $\boldsymbol{m}_{(-1)} = \boldsymbol{m}$.
  else
    MASTER:
    • compute ${\boldsymbol{V}'}^{-1}_{(s)}$ and ${\boldsymbol{V}'}_{(s+1)}$ according to (5.14).
    • encrypt $\tilde{\boldsymbol{V}}'^{-1}_{(s)} \leftarrow \mathtt{Enc}_{\mathtt{pk}}({\boldsymbol{V}'}^{-1}_{(s)})$ and $\tilde{\boldsymbol{V}}'_{(s+1)} \leftarrow \mathtt{Enc}_{\mathtt{pk}}({\boldsymbol{V}'}_{(s+1)})$ and
      $\tilde{\boldsymbol{m}}'_{(s-1)} \leftarrow \mathtt{Enc}_{\mathtt{pk}}(\mathtt{Dec}_{\mathtt{sk}}({\boldsymbol{m}'}_{(s-1)}))$.
    • send $\{\tilde{\boldsymbol{V}}'^{-1}_{(s)}, \tilde{\boldsymbol{V}}'_{(s+1)}, \tilde{\boldsymbol{m}}'_{(s-1)}\}$ to client s.
    CLIENT:
    • compute ${\boldsymbol{m}'}_{(s)}$ according to (5.21) with K iterations.
    • return ${\boldsymbol{m}'}_{(s)}$ to the master.
  end if
end for

obtain these *global* SFs cooperatively, each party computes, for each predictor $j \in \mathbb{N}_{1:P}$, $G_{(s)j} = \sum_{i=1}^{N_s} X_{(s)ij}$ and $H_{(s)j} = \sum_{i=1}^{N_s} X^2_{(s)ij}$, and passes it to the next party. At the end we obtain $G_j = \sum_{s=0}^{S}\sum_{i=1}^{N_s} X_{(s)ij}$ and $H_j = \sum_{s=0}^{S}\sum_{i=1}^{N_s} X^2_{(s)ij}$. Then the master computes the mean and standard deviation of each predictor as follows (after decrypting G_j and H_j): $\mathbb{E}(X_j) = G_j/N$ and $\mathbb{V}(X_j) = H_j/N - \mathbb{E}(X_j)^2$. The mean of the responses is also required and can be calculated analogously to the mean of the predictors. Naturally, this would add a considerable amount of computation and communication costs to the algorithm, and would require interaction between all parties whenever the dataset is augmented.

A second option is for each party s to standardise its own data using only information in its block of the total data, $\boldsymbol{D}_{(s)}$. We will generally prefer this *party-specific* standardisation option. The effects of these two different standardisation approaches on the estimation procedure will be discussed in §5.5.

A third option is for the master to supply the clients with SFs based only on $\boldsymbol{D}_{(0)}$. This seems less warranted theoretically, and also performs worse empirically.

5.4.5 Theoretical parameter requirements for the FV cryptosystem

Here we give some guidance on how to choose the cryptographic parameters in the FV scheme to run ECBLR-QN. A bound for the MCS (max) can be found by solving the recursion $||\tilde{\boldsymbol{\beta}}^{[k]}||_\infty \leq (3k-2)n(n+1)^2P(P+N)+[3n+(n+1)^3NP^2+(3k-2)n(n+1)^2P^2]||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$ with starting value $||\tilde{\boldsymbol{\beta}}^{[1]}||_\infty \leq n(n+1)^2P(P+N)$. The method for obtaining this bound is similar to that used in the previous chapter for gradient descent, §4.3.5; see §B.6 for details. The MMD (L) grows linearly with the number of QN iterations and is equal to $3K$.

5.5 Results

Data description. We apply the methods developed in the previous chapters to a real-world regression dataset from the UCI machine learning data repository (Lichman, 2013). We chose the concrete dataset ($N = 1000$ observation and $P = 8$ predictors), which contains information on concrete compressive strength (response) and a set of variables such as age and ingredients (predictors). The reason that this dataset was selected was that it has approximately the size we were looking for to conduct our experiments. In particular, is has a large the number of observations which allows us to partition the dataset horizontally into blocks with a reasonable number of observations. We use $\phi = 4$ throughout.

We use the root mean squared deviation (RMSD) of two vectors, $\boldsymbol{u}$ and $\boldsymbol{v}$ of length n, as a distance measure between the (elements of the) two vectors: $\text{rmsd}(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{(\boldsymbol{u}-\boldsymbol{v})^T(\boldsymbol{u}-\boldsymbol{v})/n}$. If $\boldsymbol{v}$ is the target quantity of the process generating $\boldsymbol{u}$, then $\text{rmsd}(\boldsymbol{u}, \boldsymbol{v})$ represents the L_2 error norm of $\boldsymbol{u}$ with respect to the target $\boldsymbol{v}$.

5.5.1 Performance

ECBLR-QN produces coefficient profiles that approximately match the true recursive Bayesian posterior updates in (5.13). The accuracy of these estimates depends on the size of the data blocks, N_s, given a fixed number of observations, N. Figure 5.4 shows the convergence of the posterior mean of the regression coefficients. In the scenario with 10 parties ($N_s = 100$), the quasi-Newton method typically converges within $K = 2$ iterations, with the error

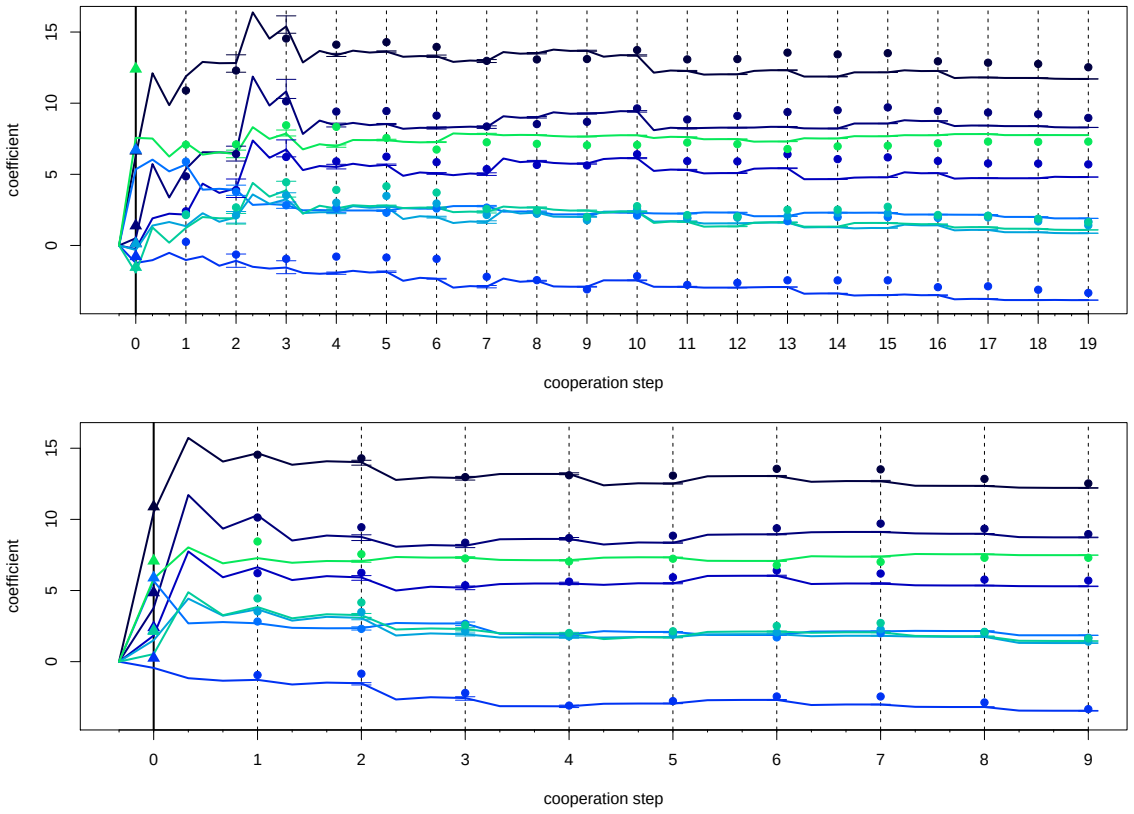


Figure 5.4: Convergence `ECBLR-QN` with global SFs for different data block sizes: **[top]** 50 observations per party (o.p.p.); **[bottom]** 100 o.p.p.. With only 2 iterations per party ($K = 3$ shown), the estimated values (colored lines intersected with vertical dashed lines) are very close to the corresponding exact values (triangles/circles). Error bars (± 2 stdv) reflect the variation in the bootstrap samples used to approximate $\boldsymbol{V}$ (party 1 receives the exact variance from party 0 and, therefore, has no error bars).

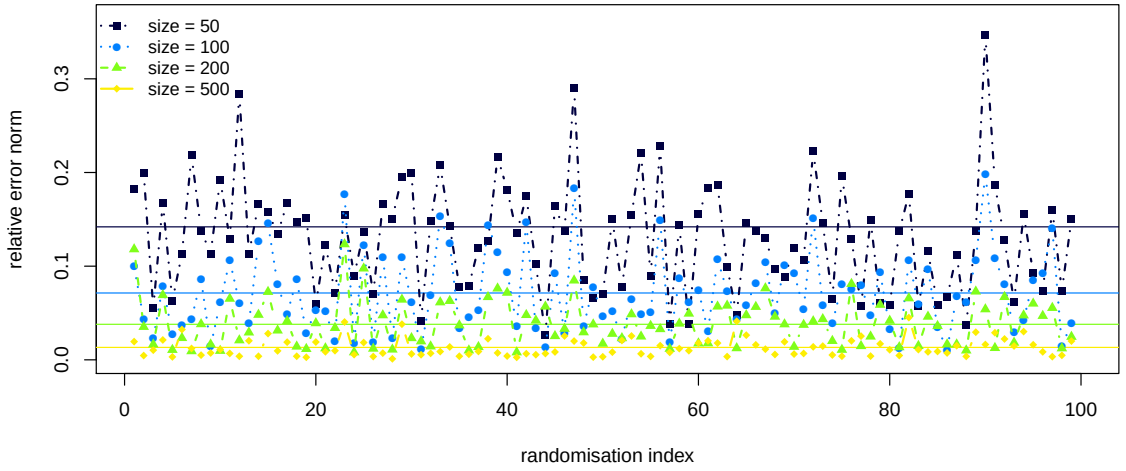


Figure 5.5: The relative error norm of the posterior mean decreases with increasing partition size. Along the x-axis we show different randomised splits of the data; horizontal lines are the corresponding averages for each of four partition sizes.

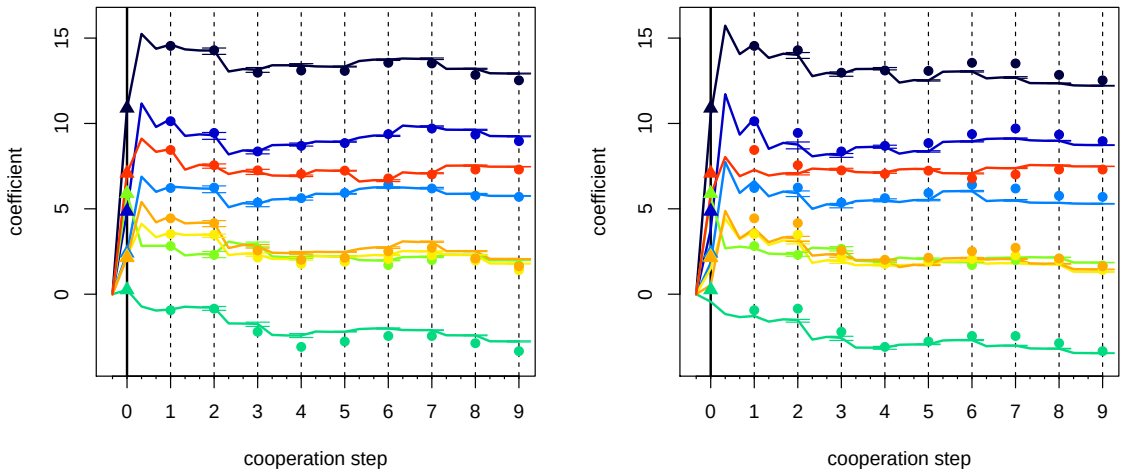


Figure 5.6: The effect of the type of data standardisations on `ECBLR-QN`: **[left]** global standardisation, i.e., based on the entire dataset; **[right]** party-specific standardisation, i.e., based solely on each party's data block. The lines represent the `ECBLR-QN` estimates and the points represent the true values. $[N_s = 100, K = 3]$

relative to the true posterior mean stemming from the approximation of the posterior variance. Figure 5.5 shows the relative error norm of the posterior mean for different partition sizes. Where error bars are shown, they were computed based on $B = 10$ bootstrap samples of the master's data used to approximate the posterior variance. The effect of this type of bootstrapping on the estimated posterior mean and variance is negligible in practice, as can be seen from the small error bars in all Figures.

The strategy used to compute the SFs also has an impact on the accuracy of `ECBLR-QN`. In Figure 5.6 we compare *global* versus *party-specific* standardisation. Even though global SFs produce the most accurate estimates, using party-specific SFs we are able to achieve remarkably good results as well.

One concern in this iterative scenario is whether the algorithm can handle non-homogeneously distributed samples. For example, if the dataset includes a binary predictor Z taking values $z_i \in \{0, 1\}$, then some parties may hold a minority of $z_i = 0$ samples while other parties hold a minority of $z_i = 1$ samples. To test this, we created an artificial binary predictor correlated with the response variable ($\text{corr}(Y, Z) = 0.55$) and only moderately correlated with the remaining predictors ($\max_j \text{corr}(X_j, Z) = 0.24$). Each party is then allocated N_s of the total N observations where the samples z_i are unevenly distributed, that is, some parties have a minority of $z_i = 1$ samples while others have a majority. The `ECBLR` algorithm is able

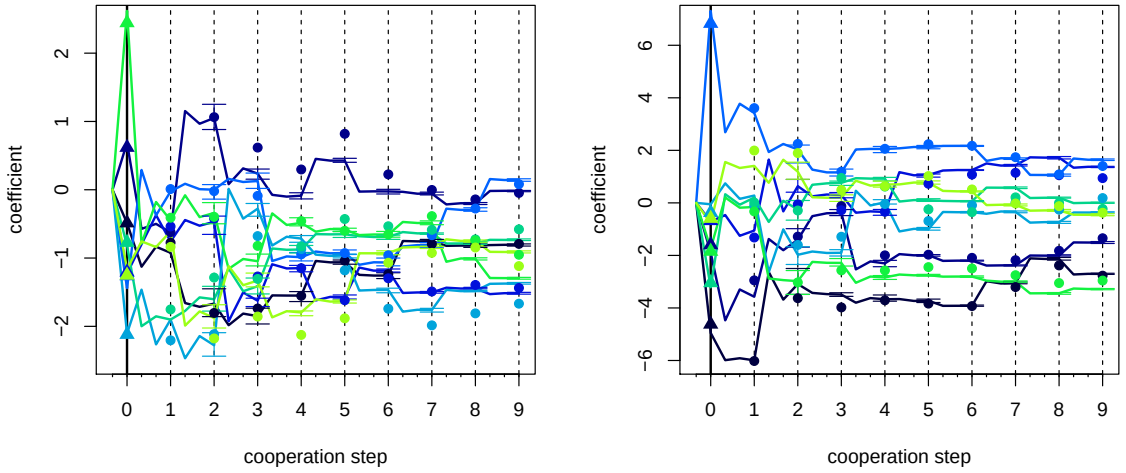


Figure 5.7: Robustness of `ECBLR-QN` with respect to the correlation structure among predictors: **[left]** low correlation, $\rho = 0.46$; **[right]** high correlation, $\rho = 0.82$ $[N_s = 100, K = 3]$

to learn the posterior means of all other variables very accurately, although it shows some difficulty in learning the posterior mean of the binary predictor when the minority class is very under-represented (Figure 5.8). Note that this refers only to the way in which samples are distributed among the cooperative parties, not the unbalance in the binary predictor overall, which is assumed to be balanced, i.e., $\sum_{i=1}^{N} \mathbb{I}(z_i = 1) = \sum_{i=1}^{N} \mathbb{I}(z_i = 0) = N/2$.

One may also wish to use the posterior variance matrix to quantify the uncertainty around the posterior mean. $\boldsymbol{V}$ tends to be estimated quite precisely, especially when the data is distributed homogeneously among parties (Figure 5.9). Although in both cases the majority of the values concentrate around 0, when data is non-homogeneously distributed (based on a binary predictor, as before) we observe more extreme values, particularly associated with the binary predictor. Note that the values shown, $(\boldsymbol{V}_{\text{est}} - \boldsymbol{V}_{\text{true}})/\boldsymbol{V}_{\text{true}}$, are relative to the true variance, and the large values are a product of the very small magnitude of the denominator.

Unlike the accelerated least squares algorithm (`ELS-GD`, chapter 4), the present method is robust in the presence of highly correlated predictors (Figure 5.7). The reason is that the quasi-Newton method uses second-order derivative information and, therefore, is able to take into account the curvature of the log-likelihood to obtain a more accurate update, regardless of the structure of the predictors' correlation matrix.

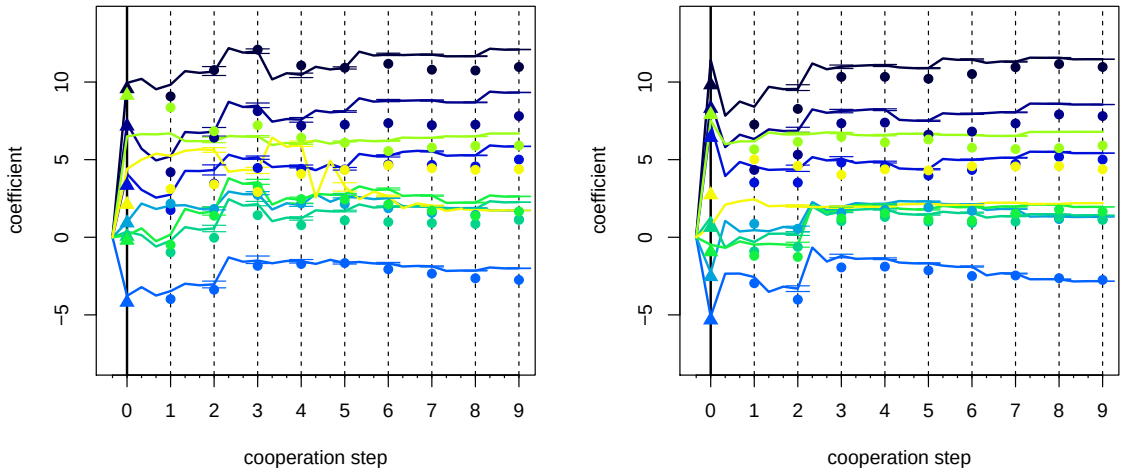


Figure 5.8: Robustness of `ECBLR-QN` with respect to the data allocation method when the distribution of an artificial binary predictor Z taking values $z_i \in \{0, 1\}$ (correlated with the response) is uneven: **[left]** clustered allocation (parties 1–5 have a minority of $z_i = 1$ samples; parties 6–10 have a minority of $z_i = 0$ samples); **[right]** random allocation (party s is allocated a random sample, which could contain a minority or a majority of $z_i = 1$ samples). The minority class constitutes 20% and the majority class 80%. The lines represent the `ECBLR-QN` estimates and the points represent the true values, with the binary predictor profile shown in yellow. $[N_s = 100, K = 3]$

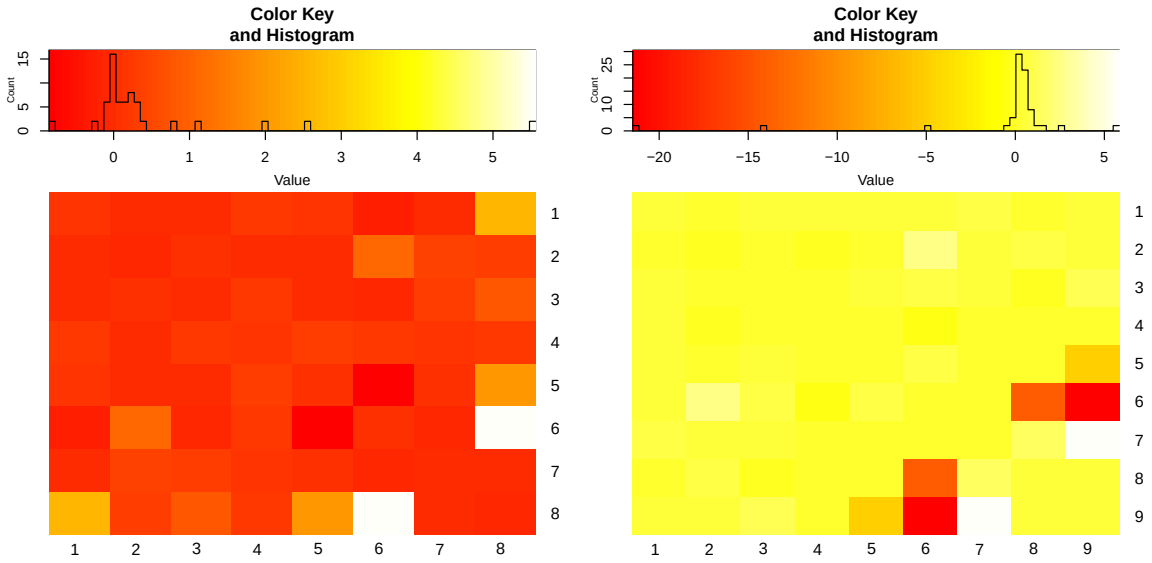


Figure 5.9: Heatmap depicting the relative difference between the final estimated posterior variance matrix and the true one, i.e., $(\boldsymbol{V}_{\text{estimated}} - \boldsymbol{V}_{\text{true}})/\boldsymbol{V}_{\text{true}}$: **[left]** without a non-homogeneously distributed binary predictor; **[right]** with a non-homogeneously distributed binary predictor (random allocation; minority class 20%). $[N_s = 100, K = 3]$

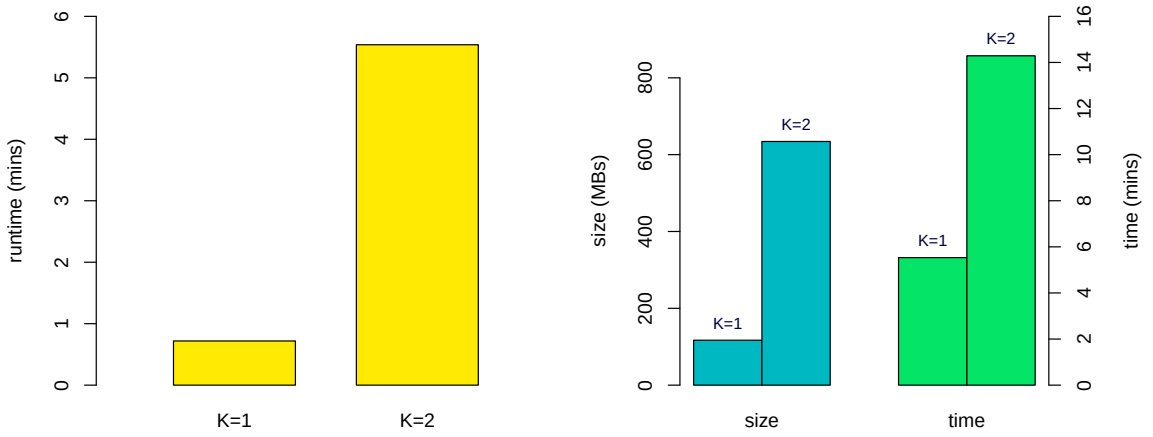


Figure 5.10: Computational costs of `ECBLR-QN` per cooperation step: **[left]** runtime (in minutes); includes the homomorphic update of the posterior mean via QN method by the client but excludes the non-homomorphic update of the posterior variance by the master, which is negligible; **[right]** total transfer size (in megabytes; blue, left axis) and total transfer time (in minutes; green, right axis); includes the transfer of ciphertexts $\{\tilde{\boldsymbol{V}}'^{-1}_{(s)}, \tilde{\boldsymbol{V}}'_{(s+1)}, \tilde{\boldsymbol{m}}'_{(s-1)}\}$ from master to client and $\{\tilde{\boldsymbol{m}}'_{(s)}\}$ from client to master, totaling $P(P+2)$ ciphertext elements. [`concrete` dataset, $N_s = 100, P = 8$]

5.5.2 Computational costs

In Figure 5.10 we show the computational costs associated with the `ECBLR-QN` algorithm for the `concrete` dataset with 10 parties ($N = 1000, N_s = 100, P = 8$). The costs are given per cooperation step. Runtimes include the client's update of the posterior mean via the quasi-Newton method and were measured on a 48-core server. Communication costs include the transfer of all required ciphertexts between master and client, a total of $P(P+2)$ elements, namely: P elements from $\tilde{\boldsymbol{m}}'_{(s-1)}$ and $2((P^2-P)/2+P)$ elements from $\tilde{\boldsymbol{V}}'^{-1}_{(s)}$ and $\tilde{\boldsymbol{V}}'_{(s+1)}$. See §A.3 for hardware and software details.

The computational costs increase rapidly with the number of quasi-Newton iterations used in each cooperation step. Typically, $K = 2$ gives very accurate results (Figure 5.4), which implies a computational time of around 5.5 minutes and a total communication time of around 14 minutes per cooperation step.

In terms of memory requirements, the theoretical ciphertext size (excluding overheads) is around 1 MB, implying a total of 1.03 GBs for $\boldsymbol{D}_{(s)} = \{\boldsymbol{X}_{(s)}, \boldsymbol{y}_{(s)}\}$, i.e., per cooperative party.

5.6 Discussion

Scalability

The computational framework proposed is particularly well-suited for analysis of large-N problems. It is possible to artificially increase the number of cooperative parties by splitting data into smaller blocks, which significantly reduces the storage and memory costs per cooperative step. Similarly, the method can be used for multi-party, dynamic learning in an online data stream scenario. In the case of large-P problems we are faced with increased communication costs — $P(P+2)$ ciphertext elements per cooperation step — and computational costs, but otherwise an increase in the number of predictors has little effect on the convergence of the quasi-Newton method.

Non-homogeneity

An important limitation of `ECBLR` is when the blocks of data are split non-homogeneously among parties, which is to say: when their empirical distributions deviate significantly from each other. The simplest instance of this problem (and also the most extreme) occurs when observations on a binary predictor, $z_i \in \{0, 1\}$, are not distributed uniformly among cooperative parties, say when some parties hold a minority of $z_i = 1$ samples while others hold a majority. Although the algorithm works well for reasonable levels of non-homogeneity, e.g., minority class $> 20\%$, for more extreme levels of non-homogeneity it becomes increasingly difficult to estimate the posterior mean and variance accurately. A cooperative solution to this problem is to do a round of communication before the computation stage where we randomise the observations held by each party. In this scenario, at least some parties must be willing to share homomorphically encrypted data with other parties. If this solution is acceptable by all parties, one may go a step further and pool all data together, thus reducing the problem to a 2-party computation exercise, with the added benefit that estimation of the posterior mean requires only one approximation, and so is expected to be more accurate.

Theoretical analysis

An important analysis that remains to be done is the effect of the approximated variance on the convergence of the algorithm proposed from a theoretical perspective. Assuming that this is the only source of approximation (i.e., excluding errors stemming from data standardisation and the quasi-Newton method), it should be possible to derive theoretical bounds on the asymptotic error incurred as a function of the precision of the variance approximation. This will provide us with an idea of how inaccurate out method can be under different scenarios, e.g., homogeneous vs. non-homogeneous data splits.

Increased security

In the computational framework developed, the master is assumed to be *honest-but-curious* or *semi-honest*; that is, it does not deviate from the protocol established but merely gathers information from it. This is a standard assumption in multi-party models of computation. Because the master gets to see the sequence $\{\boldsymbol{m}_{(s)}\}_{s\in\mathbb{N}_{1:S}}$ in unencrypted form, there may be concerns regarding the security of the data held by parties 1 through S. Specifically, knowing these values may leak some information about about the individual records in $\{\boldsymbol{X}_{(s)}, \boldsymbol{y}_{(s)}\}$, and thus compromise privacy. Therefore, depending on the level of trust that parties put on the master, it may be desirable to strengthen security in these intermediate steps of computation. One possible solution is to make use of differential privacy (DP) to perturb the posterior mean before it is returned to the master (see §2.5.2). Although work has been done on differentially-private regression models (Wang et al., 2015b; Zhang et al., 2012) and stochastic gradient descent methods (Rajkumar and Agarwal, 2012; Song et al., 2013), differentially-private quasi-Newton iterative approaches have not been researched in the context of linear regression, even though they have been researched in different contexts (Ji et al., 2014; Yuan et al., 2016). This constitutes a very interesting avenue of research in the context of FHE and MPC, as a complementary security guarantee.

Cooperative Bayesian linear regression by gradient descent

The quasi-Newton method can be seen as equivalent to a gradient descent algorithm with a particular choice of step size, which is a matrix instead of a scalar. The simple gradient descent method with scalar step size is, nonetheless, a valid approach. We will mention here some practical issues, and note that the work presented in chapter 4 can be straightforwardly applied in the present framework to estimate the posterior mean, and so constitutes an alternative to the quasi-Newton method.

Choosing the step size. The step size δ should be chosen within the spectral radius of $\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{V}^{-1}$, but because $\boldsymbol{V}^{-1}$ comes encrypted, it cannot be computed. However, if the party sending $\boldsymbol{V}^{-1}$ encrypted is willing to share the largest eigenvalue of $\boldsymbol{V}^{-1}$ in unencrypted form, we can use some theoretical results to estimate the step size. For matrices $A, B, C = A+B$, bounds for the largest eigenvalue of C as a function of the largest eigenvalues of A and B exist (Fulton, 2000, chapter 1): if $\alpha_{\max}, \beta_{\max}, \gamma_{\max}$ are the largest eigenvalues of A, B, C, respectively, then $\gamma_{\max} \leq \alpha_{\max} + \beta_{\max}$ which implies $1/\gamma_{\max} \geq 1/(\alpha_{\max} + \beta_{\max})$.

Speeding-up convergence. In addition to the methods proposed in chapter 4 to accelerate convergence of GD, we can leverage the fact that our present framework allows intermediate communication. Because we can only perform a small number of iterations, it may be useful to split a party's data into smaller chunks and update the posterior with each chunk of data sequentially, which is equivalent to artificially increasing the number of parties and, therefore, the total number of iterations in the algorithm. However, smaller chunks also means lower accuracy, and so this poses a trade-off that needs to balanced. Alternatively, we may simply allow the master to re-encrypt ciphertexts after a certain number of iterations, so that parties can continue the iterative procedure without risk of ciphertext noise overflow.

5.7 Summary

In this chapter we provided a complete computational framework for cooperative Bayesian linear regression with distributed data. In this framework, several parties interact to compute a global statistic (a function of all the data that is distributed among all parties) without any

party gaining access to the data of the remaining parties. As a consequence, we are able to learn the posterior mean and variance in a privacy-preserving manner.

We highlight two key methodological contributions of this chapter. First, the computational framework itself, which is the cornerstone of our approach. An honest-but-curious party (the master), capable of performing non-homomorphic computation securely, cooperates with a group of 'smaller' parties (the clients), to sequentially update the posterior mean and variance of the regression coefficients without leaking data-level information. The approach makes use of recursive Bayesian updating, which allows: (i) future data to be easily incorporated in the model (requiring only one round of cooperation between the master and the new party); (ii) very efficient storage of information (only the posterior mean vector must be stored in homomorphically encrypted format). Second, we propose two approximation methods to overcome computational difficulties imposed by FHE. In particular, we show how to efficiently estimate the two key hyperparameters, the regression coefficients' posterior variance and mean. The former is approximated by using the master's data as a proxy for that of the client; while the later is estimated by using an iterative procedure based on the quasi-Newton method.

The topic of Bayesian linear regression has not received its due attention in the secure computation literature, particularly in the context of FHE. The computational framework proposed here improves considerably on current methods (Ghosh and Reiter, 2013), namely by allowing outsourcing of computation to the cloud and reducing the amount of interactions between the parties involved.

Chapter 6

Conclusion

6.1 Summary

Contributions

In this thesis we developed statistical and machine learning methods for privacy-preserving analysis of sensitive information in the cloud. Our contribution is threefold.

(1) Overview. The first contribution is informative and comprises an introduction to the issues of data privacy and security (chapter 1), and an in-depth overview of fully homomorphic encryption, including its potential and limitations for privacy-preserving computation (chapter 2). These are delivered in a manner that is accessible to statisticians and applied scientists. Privacy breaches can carry catastrophic consequences, not only for those whose privacy is infringed, but also for their extended families in certain situations (e.g., genetic data) and in general for research (e.g., reluctance to donate data to medical studies). Thus, privacy-preserving methods backed by strong security guarantees constitute a keystone in the effort of minimising privacy risks and facilitating collaborative research involving private data. FHE is a recent development in cryptography which allows data to be simultaneously protected from attacks and usable in computation. More precisely, FHE allows computation to be performed directly with encrypted data (i.e., without the need of decrypting it), with the guarantee that the final result, when decrypted, matches the result that we would have obtained had we performed the same computation with unencrypted data. In short, FHE

allows the computation of statistical functions without seeing the data and, for this reason, holds enormous promise for the future of privacy-preserving computation. The cost of privacy comes in the form of increased computational costs and a restricted set of algorithms that can be implemented. Working under these constraints, our goal was to design and implement statistical algorithms so as to reach a good balance between security, accuracy and computational costs.

(2) Methodology. The second contribution is methodological and comprises the design and study of the trade-offs of different statistical algorithms under the computationally-restricted framework of FHE (chapters 3–5). We first tackled classification. To overcome the problems encountered with traditional classifiers in the context of FHE (e.g., matrix inversions, non-polynomial function evaluations) we proposed a simple, semiparametric Bayes classifier which directly models the decision boundary between the two response classes as combination of univariate logistic regressions, for which a suitable approximation was also presented. Notably, both estimation and prediction can be implemented homomorphically. The algorithm performs competitively against other methods in terms of classification error and has very attractive computational properties, namely a low and constant MMD that is independent of the problem size. We then tackled regression via two approaches. First, we proposed an accelerated version of gradient descent (VWT) that differs from the state-of-the-art approach (NAG) but which achieves convergence quicker under the computationally restricted framework of FHE. Unlike other methods in the literature, this approach allows both estimation and prediction to take place in the encrypted domain. Second, we proposed a computational framework for cooperative Bayesian linear regression with distributed data, in which several parties cooperate to learn a joint model without leaking information about their individual data to the remaining parties. The approach is grounded on recursive Bayesian updating, which allows the posterior mean and variance of the regression coefficients to be updated sequentially and efficiently with respect to memory requirements. Suitable approximations to the posterior variance and mean are proposed: the former relies on the homogeneity of the distributed data blocks, and later on the quasi-Newton method. In both of these approaches the MMD scales linearly with the number of iterations, K, although in the first approach K should be viewed as a function of the correlation structure among predictors and, therefore, of the number of predictors itself. For all the methods proposed, we give details on any

encoding and scaling of the data and algorithms that are necessary to represent the data in the encryption scheme's message space (integers). Additionally, we derive theoretical bounds for the cryptographic parameters of the `FV` scheme necessary to guarantee correct decryption. Without these, we have no guarantees to be able to correctly decrypt the results of even simple statistical algorithms.

(3) Implementation. The third contribution is practical and comprises an open-source implementation of the privacy-preserving methods developed in this thesis (appendix A). Building on the `HomomorphicEncryption` package — which implements the cryptographic primitives of the `FV` cryptosystem (Aslett, Esperança, and Holmes, 2015a) — the `EncryptedStats` package provides a user-friendly implementation of the classification and regression methods, including estimation and prediction stages. The package is in continued development and we plan to expand it to include other FHE-based statistical methods put forward in the literature, such as linear means and linear discriminant classifiers (Graepel et al., 2013).

Limitations

FHE comes with a set of computational constraints. The most important are the absence of encrypted division and logical comparison operators (see §6.2 for some recent developments on this issue), the restrictions on the evaluation of non-polynomial functions, and the reduced multiplicative depth of the algorithms that can be efficiently evaluated homomorphically.

As a result, traditional strategies for solving regression and classification problems — for instance those based on matrix inversions and decompositions — seem destined to fail in the context of homomorphic computation without multi-party communication methods. Similarly with iterative methods that require data-dependent rescaling at each iteration, for example, conjugate gradient descent and principal component analysis; or iterative methods that simply converge slowly and, consequently, have high multiplicative depth. The natural solution in these scenarios would be to employ FHE-based multi-party computation, which we will discuss in §6.3.

Another important class of algorithms that are currently intractable under FHE includes those requiring logical operators such as inequality checking, for instance, simulation methods

(e.g., rejection sampling), decision trees (e.g., random forests) and lasso-type regularisation (e.g., incremental forward stagewise regression). Circumventing these issues is non-trivial. Using a special data representation method, the MNB classifier presented in §3.3 is able to sidestep inequality checking. Similarly, Aslett, Esperança, and Holmes (2015b) propose a tailored method for fitting random forests with the same property. The computational costs are relatively high, but perhaps the approach can be applied successfully to other classes of algorithms.

6.2 Developments

Research in FHE continues to produce new and exciting methods with important implications for the field of privacy-preserving computation. Here we report on some recent developments which bring FHE closer to being practical.

Speeding-up homomorphic computation

Graphics processing units. GPUs are capable of providing phenomenal performance improvements for problems that can be represented as *single instruction, multiple-data* (SIMD). Early implementations report speed-ups in homomorphic operations of the order of around 7×, relatively to CPU implementations, and even faster methods have recently appeared in the literature (Dai and Sunar, 2015; Khedr and Gulak, 2016; Lee et al., 2015; Wang et al., 2012, 2015a).

Field programmable gate arrays and microprocessors. FPGAs are integrated circuits that can be reprogrammed after manufacturing to fit specific computational requirements. With FPGAs, one effectively programmes the machine hardware, which can lead to significant efficiency gains when compared to traditional software programmes that run on standard processors. There as been recent work on using FPGAs to speed-up FHE computation: Chen et al. (2015); Cilardo and Argenziano (2016); Du et al. (2016); Öztürk et al. (2015); Roy et al. (2015), In particular, these works focus on improvement the efficiency of the large integer arithmetic calculations required by homomorphic operations, such as large integer

multiplication and polynomial multiplication. Breuer and Bowen (2016) developed a microprocessor that 'works encrypted': data in memory is kept encrypted and so can be used to securely process data remotely.

Ciphertext packing. Typically, a dataset is encrypted by encrypting each of its entries separately, so that to each data point corresponds a ciphertext. However, it is possible to *pack* several plaintexts into one ciphertext while retaining the ability to compute with these ciphertexts in a *single instruction, multiple data* (SIMD) manner (Brakerski et al., 2013; Carpov and Sirdey, 2016; Smart and Vercauteren, 2014; Yasuda et al., 2015). This feature is currently not implemented in the `HomomorphicEncryption` package, but is planned to be incorporated.

Expanding the set of homomorphic operations

Encrypted comparison. One important bottleneck in homomorphic computation is the fact that there is currently no scheme that support comparison of encrypted values. However, just recently Arita and Nakasato (2016) proposed an extension of the `FV` cryptosystem that supports encrypted comparison: given only $\texttt{Enc}_{\texttt{pk}}(x_1)$ and $\texttt{Enc}_{\texttt{pk}}(x_2)$, the authors show how to obtain $\texttt{Enc}_{\texttt{pk}}(b)$, where $b = 1$ if $x_1 > x_2$ and $b = 0$ otherwise. Importantly, this does not require knowledge of the secret key or communication with the secret key holder. However, no implementations of the method exist yet. Note that this still does not allow algorithmic control flow (e.g., `if` statements and `while` loops), as the method gives only an *encrypted* version of the logical condition.

Fixed-point and floating-point arithmetic. Several authors have recently proposed extensions of FHE scheme to accommodate encryption and arithmetic with fixed-point/floating-point numbers (Arita and Nakasato, 2016; Cheon et al., 2016; Costache et al., 2016; Laine and Player, 2016). Representating real numbers natively is certainly more elegant than the alternative representations commonly used (see §3.1), although the computational overheads of these two alternative approaches have not yet been explicitly compared, and so it remains unclear whether native encoding of real values is computationally preferable. Particularly interesting is the strategy of Laine and Player (2016) to encode fixed-point numbers, which builds on the `FV` scheme and uses the same polynomial representation as the implementation

in the `HomomorphicEncryption` package (Aslett, Esperança, and Holmes, 2015a; see §2.4.1). This is very recent work (published on 6 September 2016) and the scope of applicability of this encoding method remains unclear. In particular, some coefficients in the polynomial hold information on the integer part of a real value while the others hold information on the fractional part. The two sets of coefficients cannot interact during computation and it is unclear what restrictions this imposes on computation. The authors' `C++` implementation does not use any external dependencies such as the `gmp` library for efficient, large-integer arithmetic, and so homomorphic operations are significantly slower than those obtained using the `HomomorphicEncryption` package (Aslett, Esperança, and Holmes, 2015a).

6.3 Future work

Much work remains to be done in FHE-based, privacy-preserving statistical analysis. In fact, this subfield of secure computation has only just emerged (Graepel et al., 2013; Wu and Haven, 2012). The work in this thesis reveals a few research directions that deserve special attention for their importance as cornerstone methods in statistical applications and for the challenges they present.

Some were already mentioned in §6.1 and in the main chapters of this thesis, and include: (i) data transformation techniques that lead to feasible implementation of otherwise impossible arithmetic operations, e.g., comparisons (§3.1); (ii) efficient non-homomorphic computation using methods tailored of homomorphic computation (§3.6); (iii) acceleration methods for correlated data (§4.6); (iv) weighted regression for clinical decision support (§4.6); (v) increased security in MPC via DP (§5.6). Therefore, we present here a brief outline of what in our opinion is one of the most promising directions for future research.

FHE-based MPC with added DP

The future of privacy-preserving computation will likely involve elements from all the three major privacy paradigms: fully homomorphic encryption (FHE), multi-party computation (MPC) and differential privacy (DP). FHE provides strong privacy guarantees and allows

computation to be securely outsourced to the cloud; MPC provides the possibility of computing functions intractable under FHE; and DP provides additional security to preserve privacy during the communication process.[1] We shall refer to this framework as *multi-party and differentially-private fully homomorphic encryption* (MADFHE).

Given the state-of-the-art cryptographic methods available, algorithms such as principal component analysis and conjugate gradient descent can be tackled only in a MPC scenario. However, because solving these problems via FHE-based MPC would require intermediate summary statistics to be shared in unencrypted format during communication stages — so that divisions and comparisons can be performed — this may raise privacy concerns regarding the potential leakage of information contained in these statistics. In that case, DP can be useful to prevent honest-but-curious parties from obtaining information that could be used to identify individual's records in the database.

To date, and to the best of our knowledge, only one article in the literature has considered this possibility (Aono et al., 2015). The authors examine the case of linear regression, but for other methods — e.g., the quasi-Newton method used in chapter 5 — it remains unclear which perturbation mechanism provides the best accuracy–privacy trade-off (e.g., Laplace or functional mechanisms; Dwork and Roth, 2014; Zhang et al., 2012), and work in this area would positively strengthen the privacy guarantees of such an approach.

Moreover, by allowing intermediate statistics to be re-encrypted during the round of communication between the server and the master client, MADFHE would effectively eliminate the need for ciphertext bootstrapping, thus allowing for algorithms with high multiplicative depth to be implemented.

[1] We use the term MPC here in the sense of FHE-based cooperative computation, as described in chapter 5, and not in the traditional sense (SMPC) which involves, e.g., garbled circuits or oblivious transfer protocols.

Appendix A

Software and implementation

All encrypted analysis presented in thesis was done with code that is now part of the package `EncryptedStats` for the `R` programming language. It implements the methodology for encrypted statistical analysis developed in Chapters 3–5 and is the subject of §A.2. This package makes use of the `R` package `HomomorphicEncryption` (Aslett, Esperança, and Holmes, 2015a), which implements the basic primitives for the `FV` cryptosystem (Fan and Vercauteren, 2012) detailed in Section 2.4 and is briefly described in §A.1.

A.1 The `HomomorphicEncryption` package

The `HomomorphicEncryption` package, developed by Louis Aslett, provides a `C++` implementation of the `FV` cryptosystem accessible from `R`. It includes the three essential steps in any encryption scheme, namely: parameter selection, key generation and encryption/decryption. Additionally, it implements homomorphic operations (addition and multiplication) using operator overloading.

A.1.1 Initialisation and basic cryptographic functions

A.1.1.1 Parameter selection

The `FV` scheme has four parameters which must be selected:

- `d`, the power of the cyclotomic polynomial ring (default 4096);
- `sigma`, the standard deviation of the discrete Gaussian used to induce a distribution on the cyclotomic polynomial ring (default 16.0);
- `qpow`, the power of 2 to use for the coefficient modulus (default 128);
- `t`, the value to use for the message space modulus (default 32768).

which are arguments of the `pars` function, the first argument of which selects the encryption scheme to use. Presently only Fan and Vercauteren (2012) is implemented (`"FandV"`). The default cryptographic parameters are assumed unless specifically set:

```
> p <- pars("FandV")
> p
Fan and Vercauteren parameters
phi = x^4096+1
q = 340282366920938463463374607431768211456 (128-bit integer)
t = 32768
delta = 1038459371706965525706099265844019 2
sigma = 16
Security level approx 128-bits
Supports multiplicative depth of 3 with overwhelming probability
(i.e. lower bound, likely more possible)
```

Say the defaults are acceptable except for the message space modulus which should be doubled, it could be set by running:

```
> p <- pars("FandV", t=65536)
> p
Fan and Vercauteren parameters
phi = x^4096+1
q = 340282366920938463463374607431768211456 (128-bit integer)
t = 65536
```

```
delta = 5192296858534827628530496329220096
sigma = 16
Security level approx 128-bits
Supports multiplicative depth of 3 with overwhelming probability
(i.e. lower bound, likely more possible)
```

The selection of these parameters can be bewildering, so a feature of the package is a parameter helper function `parsHelp` which has a simpler set of options and will use theoretical bounds from the cryptography literature to automatically select the parameters (Lepoint and Naehrig, 2014; Lindner and Peikert, 2011). These simpler choices are:

- `lambda`, the security level required (bits), default is 80;
- `max`, the largest absolute value you will need to store encrypted, default is 1000;
- `L`, the deepest multiplication depth you need to be able to evaluate encrypted, default is 4.

So, for example, we could have selected the ability to store a larger message, but requiring at least 140 bits of security:

```
> p <- parsHelp("FandV", lambda=140, max=65536)
> p
Fan and Vercauteren parameters
phi = x^8192+1
q = 365375409332725729550921208179070754913983135744 (158-bit integer)
t = 131072
delta = 27875931498163278926919647840810451882475552
sigma = 16
Security level approx 273-bits
Supports multiplicative depth of 4 with overwhelming probability
(i.e. lower bound, likely more possible)
```

Note that the parameter helper will often be conservative in setting the security level: the theoretical bounds may not allow all constraints to be simultaneously satisfied exactly and so the security is treated as a minimum required level.

A.1.1.2 Key generation

With the parameter values stored in the output from either the `pars` or `parsHelp` functions, generating the keys is then straight-forward:

```
> keys <- keygen(p)
```

The public key is stored in `keys$pk`, the secret key in `keys$sk` and the relinearisation key (used during multiplication operations) in `keys$rlk`.

In order to save the keys for future use (or to give the public key to another party), see the `saveFHE` and `loadFHE` functions in the package.

A.1.1.3 Encryption

Encryption and decryption take place according to the same format as in §2.2.1, with encryption requiring only the public key:

```
> ct <- enc(keys$pk, 2)
> ct
Fan and Vercauteren ciphertext
( c_0 = 13x^4096+13735238375605008815549746554299664177 0x^4095+7041424...,
c_1 = -46602244866771058520503760576262076531x^4095-4071136373586351... )
```

Notice the size of the ciphertext $c = (c_0, c_1)$, a pair of polynomials of degree 4096 (abbreviated in the output above) with very large integer coefficients.

A.1.1.4 Decryption

Decryption uses the secret key to reverse the encryption operation and retrieve the a plaintext fom the ciphertext:

```
> ct <- enc(keys$pk, 2)
> m <- dec(keys$sk, ct)
> m
[1] 2
```

A.1.2 Homomorphic operations

Using homomorphic operations is the simplest aspect of all, because the package operator overloads addition (+) and multiplication (*) operations and also implements native support for vectors and matrices being encrypted to ciphertexts.

The following example mixes scalars, vectors and matrices and showcases both scalar and matrix operations:

```
> x <- 2
> y <- c(1,2,3)
> z <- matrix(1:9, nrow=3)
> z
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> xct <- enc(keys$pk, x)
> yct <- enc(keys$pk, y)
> zct <- enc(keys$pk, z)
> res <- x + y %*% t(z) %*% z
> resct <- xct + yct %*% t(zct) %*% zct
> res
```

```
     [,1] [,2] [,3]
[1,]  230  554  878
> dec(keys$sk, resct)
     [,1] [,2] [,3]
[1,]  230  554  878
```

Homomorphic operations automatically use CPU parallelisation to implement the vector or matrix operations, without any input being required from the user.

A.2 The `EncryptedStats` package

The `R` package `EncryptedStats`, developed by Pedro Esperança and Louis Aslett, provides an implementation of statistical and machine learning methods for homomorphically encrypted data. We describe here the implementation of the semiparametric naive Bayes (SNB) classifier; other implemented methods are described in Aslett, Esperança, and Holmes (2015a).

A.2.1 Semiparametric naive Bayes classifier

There are two steps in working with the SNB classifier in the package: (i) fit a model to the data, i.e., estimate the factors that make up the parameters $\{\hat{\theta}, \hat{\alpha}, \hat{\beta}\}$ (see §3.4); (ii) predict a new observation.

The user has the choice of working with encrypted or unencrypted data (this is automatically detected), with the provision that both steps must use the same type of data, i.e., either both use encrypted data or both use unecrypted data.

Assume we have an design matrix `X` of size $N \times P$ and a binary response vector `y` of length N; and let `cX` and `cy` denote the corresponding ciphertexts, encrypted as described in §A.1.

A.2.1.1 Estimation

Besides the data there are only two parameters in the estimation step: `paired` controls whether paired or unpaired optimisation is to be used (presently, only the paired version is

implemented; the unpaired version is planned to be implemented in the next release); and `pk` is the public key used to encrypt `cX` and `cy` — it is used to encrypt some constants required in the estimation process and is only needed if encrypted data is used.

Estimating the class counts $\{\text{counts}(y = 0), \text{counts}(y = 1)\}$ and the coefficients $\{a_j, b_j, d_j\}$ and is then straightforward using the fitting function `SNB.fit`. Say $N = 10$ and $P = 2$; then, with unencrypted data we obtain

```
> snb1.fit <- SNB.fit(X, y)
> snb1.fit
$counts.y
[1] 4 6
$coeffs
  [,1] [,2]
a  110  462
b   -8 -120
d  221  105
```

while with encrypted data we obtain

```
> snb2.fit <- SNB.fit(cX, cy, pk=keys$pk)
> snb2.fit
$counts.y
Vector of 2 Fan and Vercauteren ciphertexts
$coeffs
Matrix of 3 x 2 Fan and Vercauteren ciphertexts
```

We can then confirm that encrypted and decrypted results match:

```
> dec(keys$sk, snb2.fit$counts.y) == snb1.fit$counts.y
[1] TRUE TRUE
> dec(keys$sk, snb2.fit$coeffs)  == snb1.fit$coeffs
  [,1] [,2]
```

```
a TRUE TRUE
b TRUE TRUE
d TRUE TRUE
```

A.2.1.2 Prediction

The function `SNB.fit` generates an object of class `SNB`, which can be directly fed into the base R `predict` function. Apart from the fitted `model` and the testing data `newX`, there are only two parameters: `type` controls the type of prediction to the returned (raw coefficients or conditional class probabilities); `sk` is the secret key — required if predictions are to be decrypted.

Continuing the example above, we can obtain the probabilities of class $y = 1$,

```
> snb2.predProb <- predict(model=snb.fit2, newX=cX, type="prob", sk=keys$sk)
> snb2.predProb
[1] 0.2212818 0.1973391 0.7221404 0.8975560 0.7293452 0.4353264 0.4621838
[8] 0.4442447 0.7364313 0.8907048
```

or obtain the class counts and coefficients $\{e_{ij}, d_j\}$ (see §3.4) for all testing observations,

```
> snb2.predRaw <- predict(model=snb.fit2, newX=cX, type="raw", sk=keys$sk)
> snb2.predRaw
$counts.y
[1] 4 6
$coeffs.e
      [,1] [,2]
 [1,]  102 -138
 [2,]   70 -138
 [3,]   86  102
 [4,]  102  222
 [5,]   94  102
```

```
 [6,]   70  -18
 [7,]   94  -18
 [8,]   78  -18
 [9,]  102  102
[10,]   86  222
$coeffs.d
[1] 221 105
```

A.3 Implementation details

A.3.1 Hardware

Chapter 2. All encrypted analysis in this chapter was done on a Dell PowerEdge R820 server with 4 × Intel Xeon CPU E5-4657L v2 @ 2.40 GHz and 768 GB of RAM. Each CPU supports 12 cores , giving a total of 48 cores.

Chapter 3. All encrypted analysis in this chapter was done on a `c4.8xlarge` compute cluster Amazon EC2 instance. This corresponds to 36 cores of an Intel Xeon CPU E5-2666 v3 (Haswell) @ 2.9GHz and 60 GB of RAM.

Chapter 4. Same as in chapter 2.

Chapter 5. Same as in chapter 2.

A.3.2 Software

The `R` packages `HomomorphicEncryption` and (by dependency) `EncryptedStats` make use of three high-performance libraries: `GMP` (version 6.1.1; Granlund et al., 2016), a library for arbitrary precision arithmetic; `MPFR` (version 3.1.4; Fousse et al., 2016), a library for multiple-precision floating-point computations; `FLINT` (version 2.4.5; Hart et al., 2016), a library for number theory.

Appendix B

Results

B.1 Proof of Theorem 4.1

The method can be expressed generally as a *matrix iterative method* (Björck, 1996, chapter 7; Ryaben'kii and Tsynkov, 2006, chapter 6). Rewriting (4.16) gives

$$\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} + \delta \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) \tag{B.1a}$$

$$= (\boldsymbol{I}_{PP} - \delta \boldsymbol{X}^T\boldsymbol{X})\boldsymbol{\beta}^{[k-1]} + \delta \boldsymbol{X}^T\boldsymbol{y} \tag{B.1b}$$

$$= \boldsymbol{M}\boldsymbol{\beta}^{[k-1]} + \delta \boldsymbol{X}^T\boldsymbol{y} \tag{B.1c}$$

where $\boldsymbol{M} = \boldsymbol{I}_{PP} - \delta \boldsymbol{X}^T\boldsymbol{X}$ is called the *iteration matrix*. This representation will be useful to prove convergence results.

Theorem 4.1. Define $\boldsymbol{\beta}^{[k+1]} \triangleq \boldsymbol{\beta}^{[k]} + \delta \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k]})$ and let $\boldsymbol{\beta}^{[0]} \triangleq \boldsymbol{0}_P$. Then, $\lim_{k\to\infty} \boldsymbol{\beta}^{[k]} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \hat{\boldsymbol{\beta}}_{\text{ols}}$ for $\delta \in (0, 2/\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X}))$ where $\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$ is the spectral radius of $\boldsymbol{X}^T\boldsymbol{X}$.

Proof. We prove, first, that the iterative algorithm `ELS-GD` gives rise to a convergent sequence $\{\boldsymbol{\beta}^{[k]}\}$; and second, that this sequence converges to $\hat{\boldsymbol{\beta}}_{\text{ols}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$.

The iterative algorithm implied by (B.1) is said to be *convergent* if $\boldsymbol{\beta}^{[k]}$ converges for all initialisations $\boldsymbol{\beta}^{[0]}$. A necessary and sufficient condition for convergence is that the powers of

the iteration matrix converge to the null matrix, that is,

$$\lim_{k\to\infty} \boldsymbol{M}^k = \mathbf{0}_{PP}. \tag{B.2}$$

The eigendecomposition of the real, symmetric matrix $\boldsymbol{X}^T\boldsymbol{X}$ gives

$$\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T \tag{B.3}$$

where $\boldsymbol{\Lambda}$ is a $P\times P$ diagonal matrix with diagonal entries equal to the eigenvalues $[\lambda_1,\ldots,\lambda_P]$ of $\boldsymbol{X}^T\boldsymbol{X}$; and $\boldsymbol{V}$ is a $P\times P$ orthogonal matrix with columns equal to the eigenvectors $[\boldsymbol{V}_1,\ldots,\boldsymbol{V}_P]$ of $\boldsymbol{X}^T\boldsymbol{X}$. Using this decomposition we can write the iteration matrix as

$$\begin{aligned}
\boldsymbol{M} &= \boldsymbol{I}_{PP} - \delta\boldsymbol{X}^T\boldsymbol{X} \\
&= \boldsymbol{I}_{PP} - \delta\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T \\
&= \boldsymbol{V}\boldsymbol{I}_{PP}\boldsymbol{V}^T - \delta\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T \qquad (\boldsymbol{V} \text{ is orthogonal}) \\
&= \boldsymbol{V}(\boldsymbol{I}_{PP} - \delta\boldsymbol{\Lambda})\boldsymbol{V}^T.
\end{aligned} \tag{B.4}$$

and so the powers of $\boldsymbol{M}$ are

$$\boldsymbol{M}^k = \boldsymbol{V}(\boldsymbol{I}_{PP} - \delta\boldsymbol{\Lambda})^k\boldsymbol{V}^T \tag{B.5}$$

which converges to $\mathbf{0}_{PP}$ if and only if $|1-\delta\lambda_j| < 1, \forall j \in \mathbb{N}_{1:P}$; in other words,

$$0 < \delta\lambda_j < 2, \forall j \quad \Leftrightarrow \quad 0 < \delta < \frac{2}{\lambda_j}, \forall\, j \quad \Leftrightarrow \quad 0 < \delta < \frac{2}{\max_j\{\lambda_j\}}. \tag{B.6}$$

Convergence is therefore guaranteed for $\delta \in (0, 2/\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X}))$ where $\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X}) = \max_j\{\lambda_j\}$ is the spectral radius of $\boldsymbol{X}^T\boldsymbol{X}$. We omit the absolute value of the eigenvalues since, under the assumption that $\boldsymbol{X}$ is full rank, $\boldsymbol{X}^T\boldsymbol{X}$ is positive definite and so $\lambda_j > 0, \forall j$.

Assuming that we are in the convergent regime, the limiting value $\bar{\boldsymbol{\beta}} = \lim_{k\to\infty}\boldsymbol{\beta}^{[k]}$ can be easily found by imposing a stationary condition $\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} = \bar{\boldsymbol{\beta}}$ on (B.1),

$$\bar{\boldsymbol{\beta}} = \bar{\boldsymbol{\beta}} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\bar{\boldsymbol{\beta}}) \quad \Leftrightarrow \quad \bar{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \hat{\boldsymbol{\beta}}_{\text{ols}} \tag{B.7}$$

and so $\lim_{k\to\infty}\boldsymbol{\beta}^{[k]} = \hat{\boldsymbol{\beta}}_{\text{ols}}$ if $\delta \in (0, 2/\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X}))$ as required. ■

For fixed δ, this proof corresponds closely to point Jacobi matrix iterative methods, with δ controlling the matrix iteration to have at most unit spectral radius (Varga, 2000, Chapter 3; Björck, 1996, Chapter 7) and is also equivalent to the Richardon first-order method (Ryaben'kii and Tsynkov, 2006, Chapter 6).

B.2 Transformed equations: `ELS-GD`

We present here the transformed (FHE-friendly) equations for linear regression via gradient descent as presented in chapter 4. Consider the data transformations: $\tilde{\boldsymbol{X}} \approx 10^{\phi}\boldsymbol{X}$ and $\tilde{\boldsymbol{y}} \approx 10^{\phi}\boldsymbol{y}$; and the transformed parameters: $\tilde{\nu} \approx 10^{\phi}\nu$ and $\tilde{\boldsymbol{\beta}}^{[0]} \approx 10^{\phi}\boldsymbol{\beta}^{[0]}$, where $\nu \equiv 1/\delta$. All transformations are according to the method proposed in §4.1. At iteration $k \in \mathbb{N}_{1:K}$ we want to compute

$$\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) \tag{B.8}$$

or, using the transformed data,

$$\tilde{\boldsymbol{\beta}}^{[k]} \equiv F\left(\tilde{\boldsymbol{\beta}}^{[k-1]}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \tilde{\nu}\right) \propto \boldsymbol{\beta}^{[k]} \tag{B.9}$$

so that the true parameters, $\boldsymbol{\beta}^{[k]}$, can be easily obtained from the homomorphically computed parameters, $\tilde{\boldsymbol{\beta}}^{[k]}$. Next we derive the exact form of the transformed update equations, i.e., the form of F. The MMD is shown in orange.

For $k = 1$:

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[1]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[0]} + \tilde{\boldsymbol{X}}^T(10^{\phi}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[0]}) \quad \text{[2]} \\
&= 10^{\phi} \cdot 10^{\phi}\nu \cdot 10^{\phi}\boldsymbol{\beta}^{[0]} + 10^{\phi}\boldsymbol{X}^T(10^{\phi} \cdot 10^{\phi}\boldsymbol{y} - 10^{\phi}\boldsymbol{X} \cdot 10^{\phi}\boldsymbol{\beta}^{[0]}) \\
&= 10^{3\phi}\nu\boldsymbol{\beta}^{[0]} + 10^{3\phi}\nu\delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[0]}) \\
&= 10^{3\phi}\nu\{\boldsymbol{\beta}^{[0]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[0]})\} \\
&\equiv 10^{3\phi}\nu\boldsymbol{\beta}^{[1]}
\end{aligned} \tag{B.10}$$

For $k = 2$:

$$\begin{aligned}\tilde{\boldsymbol{\beta}}^{[2]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[1]} + \tilde{\boldsymbol{X}}^T(10^{2\phi}\tilde{\nu}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[1]}) \quad [4] \\ &= 10^{\phi}\cdot 10^{\phi}\nu\cdot 10^{3\phi}\nu\boldsymbol{\beta}^{[1]} + 10^{\phi}\boldsymbol{X}^T(10^{2\phi}\cdot 10^{\phi}\nu\cdot 10^{\phi}\boldsymbol{y} - 10^{\phi}\boldsymbol{X}\cdot 10^{3\phi}\nu\boldsymbol{\beta}^{[1]}) \\ &= 10^{5\phi}\nu^2\boldsymbol{\beta}^{[1]} + 10^{5\phi}\nu^2\delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[1]}) \\ &\equiv 10^{5\phi}\nu^2\boldsymbol{\beta}^{[2]}\end{aligned} \tag{B.11}$$

For $k = 3$:

$$\begin{aligned}\tilde{\boldsymbol{\beta}}^{[3]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[2]} + \tilde{\boldsymbol{X}}^T(10^{3\phi}\tilde{\nu}^2\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[2]}) \quad [6] \\ &= 10^{\phi}\cdot 10^{\phi}\nu\cdot 10^{5\phi}\nu^2\boldsymbol{\beta}^{[2]} + 10^{\phi}\boldsymbol{X}^T(10^{3\phi}\cdot 10^{2\phi}\nu^2\cdot 10^{\phi}\boldsymbol{y} - 10^{\phi}\boldsymbol{X}\cdot 10^{5\phi}\nu^2\boldsymbol{\beta}^{[2]}) \\ &= 10^{7\phi}\nu^3\boldsymbol{\beta}^{[2]} + 10^{7\phi}\nu^3\delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[2]}) \\ &\equiv 10^{7\phi}\nu^3\boldsymbol{\beta}^{[3]}\end{aligned} \tag{B.12}$$

By induction we obtain the following general expression:

$$\begin{aligned}\tilde{\boldsymbol{\beta}}^{[k]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{X}}^T(10^{k\phi}\tilde{\nu}^{k-1}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]}) \quad [2k] \\ &= 10^{\phi}\cdot 10^{\phi}\nu\cdot 10^{(2k-1)\phi}\nu^{k-1}\boldsymbol{\beta}^{[k-1]} + 10^{\phi}\boldsymbol{X}^T(10^{k\phi}\cdot 10^{(k-1)\phi}\nu^{k-1}\cdot 10^{\phi}\boldsymbol{y} \\ &\qquad - 10^{\phi}\boldsymbol{X}\cdot 10^{(2k-1)\phi}\nu^{k-1}\boldsymbol{\beta}^{[k-1]}) \\ &= 10^{(2k+1)\phi}\nu^k\boldsymbol{\beta}^{[k-1]} + 10^{(2k+1)\phi}\nu^k\delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) \\ &\equiv 10^{(2k+1)\phi}\nu^k\boldsymbol{\beta}^{[k]}\end{aligned} \tag{B.13}$$

B.3 Cryptographic parameters: `ELS-GD`

Here, we derive upper bounds for the maximum coefficient size (MCS, `max`) and the minimum degree of the cyclotomic polynomial as a power of two ($d = 2^{\mathtt{d}}$), in the `FV` cryptosystem.

The MCS to be stored is $||\tilde{\boldsymbol{\beta}}^{[K]}||_\infty = \max_{i\in\mathbb{N}_{1:P}}\{\tilde{\boldsymbol{\beta}}_i^{[K]}\}$, the value of the largest transformed regression coefficient at the last iteration of the GD algorithm as given in (4.21).

Let $s \in \{x \in \mathbb{R} : -10 \leq x \leq 10\}$ denote the message to be encrypted (standardised); let $\tilde{s} \in \{x \in \mathbb{Z} : -10^{\phi+1} \leq x \leq 10^{\phi+1}\}$ denote the corresponding transformed message, according to the encoding method presented in §3.1.2; and let $\mathring{s}(x)$ denote the corresponding

binary polynomial representation, as in §2.4.1. ϕ denotes the input accuracy (number of decimal places to be retained). The maximum degree of $\mathring{s}(x)$ is then $\deg(\mathring{s}) = \log_2(10^{\phi+1}) = (\phi+1)\log_2(10) \equiv n$.

Now, let a denote a fresh ciphertext and b denote a ciphertext that is the result of homomorphic computation, both in polynomial form. By definition, $\deg(a) \leq n \leq \deg(b) \leq m$. Additionally, $||a||_\infty = 1$ and assume $||b||_\infty = B$. Assume also that $\nu \leq 10^{2\phi}$. Then, we have the following rules, which we will use to derive an upper bound on $||\tilde{\boldsymbol{\beta}}^{[K]}||_\infty$ (by which we mean the largest coefficient in polynomial representation of $\tilde{\boldsymbol{\beta}}^{[K]}$):

$$(i) \qquad ||a+b||_\infty \leq B+1 \tag{B.14a}$$

$$(ii) \qquad \deg(a+b) \leq m \tag{B.14b}$$

$$(iii) \qquad ||ab||_\infty \leq (n+1)B \tag{B.14c}$$

$$(iv) \qquad \deg(ab) \leq n+m \tag{B.14d}$$

For iteration $k=1$, (4.21) simplifies to $\tilde{\boldsymbol{\beta}}^{[1]} = 10^\phi \tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{y}}$, which we can decompose as $\tilde{\boldsymbol{\beta}}^{[1]} = 10^\phi \tilde{\boldsymbol{\beta}}'^{[1]}$, where $\tilde{\boldsymbol{\beta}}'^{[1]} = \tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{y}}$. The matrix-vector product yields $||\boldsymbol{\beta}'^{[1]}||_\infty = ||\tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{y}}||_\infty \leq N(n+1)$ and $\deg(\tilde{\boldsymbol{\beta}}'^{[1]}) \leq 2n$. Putting this together with the scalar-vector product yields, $||\tilde{\boldsymbol{\beta}}^{[1]}||_\infty = ||10^\phi \tilde{\boldsymbol{\beta}}'^{[1]}||_\infty \leq \phi\log_2(10)N(n+1) \leq nN(n+1)$ and $\deg(\boldsymbol{\beta}^{[1]}) \leq \phi\log_2(10)(2n) \leq 3n$.

For the subsequent iterations k, we can decompose (4.21) into 3 terms as

$$\tilde{\boldsymbol{\beta}}^{[k]} = \tilde{\boldsymbol{\beta}}'^{[k]} + \tilde{\boldsymbol{\beta}}''^{[k]} + \tilde{\boldsymbol{\beta}}'''^{[k]} \tag{B.15}$$

where

$$\tilde{\boldsymbol{\beta}}'^{[k]} = 10^\phi \tilde{\nu} \tilde{\boldsymbol{\beta}}^{[k-1]} = 10^{2\phi} \nu \tilde{\boldsymbol{\beta}}^{[k-1]} \tag{B.16a}$$

$$\tilde{\boldsymbol{\beta}}''^{[k]} = 10^{k\phi} \tilde{\nu}^{k-1} \tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{y}} = 10^{(2k-1)\phi} \nu^{k-1} \tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{y}} \tag{B.16b}$$

$$\tilde{\boldsymbol{\beta}}'''^{[k]} = \tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{X}} \tilde{\boldsymbol{\beta}}^{[k-1]} \tag{B.16c}$$

The trick here is to compute products of terms sequentially such that one of them always corresponds to a freshly encrypted ciphertext, in which case we known that the largest term in its polynomial representation is 1, and so we can use the rules in (B.14). So, for instance, in (B.16c) we compute first the largest coefficient in $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]}$ and then the largest coefficient in $\tilde{\boldsymbol{X}}^T(\tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]})$.

For the first term we get $||\tilde{\boldsymbol{\beta}}'^{[k]}||_\infty = ||10^{2\phi}\nu\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq (\log_2(\nu)+2\phi\log_2(10))\cdot||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq 4n\cdot||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$ and $\deg(\tilde{\boldsymbol{\beta}}'^{[k]}) \leq 4n+\deg(\tilde{\boldsymbol{\beta}}'^{[k-1]})$.

For the second term we get $||\tilde{\boldsymbol{\beta}}''^{[k]}||_\infty = ||\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}10^{(2k-1)\phi}\nu^{k-1}||_\infty \leq [(2k-1)\phi\log_2(10)+(k-1)\log_2(\nu)]N(n+1) \leq [(2k-1)\phi\log_2(10)+(k-1)(2\phi)\log_2(10)]N(n+1) \leq [(4k-3)\phi\log_2(10)]N(n+1) \leq (4k-3)nN(n+1)$ and $\deg(\tilde{\boldsymbol{\beta}}''^{[k]}) \leq 2n+(4k-3)n \leq (4k-1)n$.

For the third term we get $||\tilde{\boldsymbol{\beta}}'''^{[k]}||_\infty = ||\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq N(n+1)[P(n+1)\cdot||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty] \leq NP(n+1)^2\cdot||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$ and $\deg(\tilde{\boldsymbol{\beta}}'''^{[k]}) \leq 2n+\deg(\tilde{\boldsymbol{\beta}}'''^{[k-1]})$.

Finally, putting the 3 terms together we get $||\tilde{\boldsymbol{\beta}}^{[k]}||_\infty \leq 4n\cdot||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty + (4k-3)nN(n+1)+NP(n+1)^2\cdot||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty = (4n+NP(n+1)^2)\cdot||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty + (4k-3)nN(n+1)$ and $\deg(\tilde{\boldsymbol{\beta}}^{[k]}) = \max\{\deg(\tilde{\boldsymbol{\beta}}'^{[k]}),\deg(\tilde{\boldsymbol{\beta}}''^{[k]}),\deg(\tilde{\boldsymbol{\beta}}'''^{[k]})\} = \max\{4n+\deg(\tilde{\boldsymbol{\beta}}^{[k-1]}),(4k-1)n\}$.

These recursive formulas can be implemented straightforwardly to provide upper bounds on the MCS and degree of the cyclotomic polynomial, for different problems.

There is an alternative way to derive a bound for the cyclotomic polynomial degree. Recall from (4.21) that $\tilde{\boldsymbol{\beta}}^{[K]} \equiv 10^{(2K+1)\phi}\nu^k\boldsymbol{\beta}^{[K]}$, where $\nu \equiv 1/\delta$ and $\delta \leq 2/\mathcal{S}(\boldsymbol{X}^T\boldsymbol{X})$. Let the magnitude of the original coefficients be upper-bounded by 10^m, that is, $||\boldsymbol{\beta}||_\infty \leq 10^m$. Then, after K GD iterations we have $||10^{(2K+1)\phi}\nu^K\boldsymbol{\beta}^{[K]}||_\infty \leq 10^{(2K+1)\phi+m}/\delta^K \equiv c$, so that we need $\log_2(c)$ smaller than the degree of the cyclotomic polynomials, d, in order for the transformed coefficients to be decrypted correctly (see §2.4.1). This implies the choice $d \geq ((2K+1)\phi+m)\log_2(10)-K\log_2(\delta)$, which grows linearly in $\{K,m,\phi\}$ and exponentially in the δ.

B.4 Transformed equations: `ELS-NAG`

To find the transformed equations for Nesterov's accelerated gradient algorithm we follow the same process as in §B.2. The MMD is shown in orange.

For $k = 1$:

$$\begin{aligned}
\tilde{\boldsymbol{s}}^{[1]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[0]} + \tilde{\boldsymbol{X}}^T(10^{\phi}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[0]}) \quad [2] \\
&= 10^{\phi}10^{\phi}\nu 10^{\phi}\boldsymbol{\beta}^{[0]} + 10^{\phi}\boldsymbol{X}^T(10^{\phi}10^{\phi}\boldsymbol{y} - 10^{\phi}\boldsymbol{X}10^{\phi}\boldsymbol{\beta}^{[0]}) \\
&= 10^{3\phi}\nu\boldsymbol{\beta}^{[0]} + 10^{3\phi}\nu\delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[0]}) \\
&= 10^{3\phi}\nu\{\boldsymbol{\beta}^{[0]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[0]})\} \\
&\equiv 10^{3\phi}\nu\boldsymbol{s}^{[1]}
\end{aligned} \tag{B.17}$$

and

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[1]} &\equiv (10^{\phi} + \tilde{\eta}_1)\tilde{\boldsymbol{s}}^{[1]} - 10^{2\phi}\tilde{\nu}\tilde{\eta}_1\tilde{\boldsymbol{s}}^{[0]} \quad [3] \\
&= 10^{\phi}(1 + \eta_1)10^{3\phi}\nu\boldsymbol{s}^{[1]} - 10^{2\phi}10^{\phi}\nu 10^{\phi}\eta_1\boldsymbol{s}^{[0]} \\
&= 10^{4\phi}\nu[(1 + \eta_1)\boldsymbol{s}^{[1]} - \eta_1\boldsymbol{s}^{[0]}] \\
&\equiv 10^{4\phi}\nu\boldsymbol{\beta}^{[1]}
\end{aligned} \tag{B.18}$$

For $k = 2$:

$$\begin{aligned}
\tilde{\boldsymbol{s}}^{[2]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[1]} + \tilde{\boldsymbol{X}}^T(10^{3\phi}\tilde{\nu}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[1]}) \quad [5] \\
&= 10^{\phi}10^{\phi}\nu 10^{4\phi}\nu\boldsymbol{\beta}^{[1]} + \delta\nu 10^{\phi}\boldsymbol{X}^T(10^{3\phi}10^{\phi}\nu 10^{\phi}\boldsymbol{y} - 10^{\phi}\boldsymbol{X}10^{4\phi}\nu\boldsymbol{\beta}^{[1]}) \\
&= 10^{6\phi}\nu^2\{\boldsymbol{\beta}^{[1]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[1]})\} \\
&\equiv 10^{6\phi}\nu^2\boldsymbol{s}^{[2]}
\end{aligned} \tag{B.19}$$

and

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[2]} &\equiv (10^{\phi} + \tilde{\eta}_2)\tilde{\boldsymbol{s}}^{[2]} - 10^{2\phi}\tilde{\nu}\tilde{\eta}_2\tilde{\boldsymbol{s}}^{[1]} \quad [6] \\
&= 10^{\phi}(1 + \eta_2)10^{6\phi}\nu^2\boldsymbol{s}^{[2]} - 10^{2\phi}10^{\phi}\nu 10^{\phi}\eta_2 10^{3\phi}\nu\boldsymbol{s}^{[1]} \\
&= 10^{7\phi}\nu^2[(1 + \eta_2)\boldsymbol{s}^{[2]} - \eta_2\boldsymbol{s}^{[1]}] \\
&\equiv 10^{7\phi}\nu^2\boldsymbol{\beta}^{[2]}
\end{aligned} \tag{B.20}$$

For $k = 3$:

$$\begin{aligned}
\tilde{\boldsymbol{s}}^{[3]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[2]} + \tilde{\boldsymbol{X}}^T(10^{5\phi}\tilde{\nu}^2\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[2]}) \quad [8] \\
&= 10^{\phi}10^{\phi}\nu 10^{7\phi}\nu^2\boldsymbol{\beta}^{[2]} + \delta\nu 10^{\phi}\boldsymbol{X}^T(10^{5\phi}10^{2\phi}\nu^2 10^{\phi}\boldsymbol{y} - 10^{\phi}\boldsymbol{X}10^{7\phi}\nu^2\boldsymbol{\beta}^{[2]}) \\
&= 10^{9\phi}\nu^3\{\boldsymbol{\beta}^{[2]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[2]})\} \\
&\equiv 10^{9\phi}\nu^3\boldsymbol{s}^{[3]}
\end{aligned} \tag{B.21}$$

and

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[3]} &\equiv (10^{\phi} + \tilde{\eta}_3)\tilde{\boldsymbol{s}}^{[3]} - 10^{2\phi}\tilde{\nu}\tilde{\eta}_3\tilde{\boldsymbol{s}}^{[2]} \quad [9] \\
&= 10^{\phi}(1+\eta_3)10^{9\phi}\nu^3\boldsymbol{s}^{[3]} - 10^{2\phi}10^{\phi}\nu 10^{\phi}\eta_3 10^{6\phi}\nu^2\boldsymbol{s}^{[2]} \\
&= 10^{10\phi}\nu^3[(1+\eta_3)\boldsymbol{s}^{[3]} - \eta_3\boldsymbol{s}^{[2]}] \\
&\equiv 10^{10\phi}\nu^3\boldsymbol{\beta}^{[3]}
\end{aligned} \tag{B.22}$$

By induction we obtain the following general expression:

$$\begin{aligned}
\tilde{\boldsymbol{s}}^{[k]} &\equiv 10^{\phi}\tilde{\nu}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{X}}^T(10^{(2k-1)\phi}\tilde{\nu}^{k-1}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]}) \quad [3k-1] \\
&= 10^{\phi}10^{\phi}\nu 10^{(3k-2)\phi}\nu^{k-1}\boldsymbol{\beta}^{[k-1]} + \delta\nu 10^{\phi}\boldsymbol{X}^T(10^{(2k-1)\phi}10^{(k-1)\phi}\nu^{k-1}10^{\phi}\boldsymbol{y} \\
&\quad - 10^{\phi}\boldsymbol{X}10^{(3k-2)\phi}\nu^{k-1}\boldsymbol{\beta}^{[k-1]}) \\
&= 10^{3k\phi}\nu^k\{\boldsymbol{\beta}^{[k-1]} + \delta\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]})\} \\
&\equiv 10^{3k\phi}\nu^k\boldsymbol{s}^{[k]}
\end{aligned} \tag{B.23}$$

and

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[k]} &\equiv (10^{\phi} + \tilde{\eta}_k)\tilde{\boldsymbol{s}}^{[k]} - 10^{2\phi}\tilde{\nu}\tilde{\eta}_k\tilde{\boldsymbol{s}}^{[k-1]} \quad [3k] \\
&= 10^{\phi}(1+\eta_k)10^{3k\phi}\nu^k\boldsymbol{s}^{[k]} - 10^{2\phi}10^{\phi}\nu 10^{\phi}\eta_k 10^{3(k-1)\phi}\nu^{k-1}\boldsymbol{s}^{[k-1]} \\
&= 10^{(3k+1)\phi}\nu^k[(1+\eta_k)\boldsymbol{s}^{[k]} - \eta_k\boldsymbol{s}^{[k-1]}] \\
&\equiv 10^{(3k+1)\phi}\nu^k\boldsymbol{\beta}^{[k]}
\end{aligned} \tag{B.24}$$

B.5 Transformed equations: `ECBLR-QN`

We present here the transformed (FHE-friendly) equations for cooperative Bayesian regression via Quasi-Newton iterations as presented in chapter 5. Consider the data standard

transformations: $\tilde{\boldsymbol{X}} \approx 10^{\phi}\boldsymbol{X}$ and $\tilde{\boldsymbol{y}} \approx 10^{\phi}\boldsymbol{y}$; the transformed variables computed by the master (party 0): $\tilde{\boldsymbol{m}} \approx 10^{\phi}\boldsymbol{m}$ and $\tilde{\boldsymbol{V}} \approx 10^{\phi}\boldsymbol{V}$; and the transformed initial guess: $\tilde{\boldsymbol{\beta}}^{[0]} \approx 10^{\phi}\boldsymbol{\beta}^{[0]}$. All transformations are according to the method proposed in §5.1. At iteration $k \in \mathbb{N}_{1:K}$ of cooperation step $s \in \mathbb{N}_{1:S}$ we want to compute

$$\boldsymbol{\beta}^{[k]} = \boldsymbol{\beta}^{[k-1]} + \boldsymbol{V}_{(s+1)}[\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) + \boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[k-1]})] \tag{B.25}$$

or, using the transformed data,

$$\tilde{\boldsymbol{\beta}}^{[k]} \equiv F\left(\tilde{\boldsymbol{\beta}}^{[k-1]}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}}, \tilde{\boldsymbol{m}}_{(s)}, \tilde{\boldsymbol{V}}_{(s)}^{-1}, \tilde{\boldsymbol{V}}_{(s+1)}\right) \propto \boldsymbol{\beta}^{[k+1]} \tag{B.26}$$

so that the true parameters, $\boldsymbol{\beta}^{[k]}$, can be easily obtained from the homomorphically computed parameters, $\tilde{\boldsymbol{\beta}}^{[k]}$. Next we derive the exact form of the transformed update equations, i.e., the form of F. The MMD is shown in orange.

For $k = 1$:

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[1]} &\equiv 10^{3\phi}\tilde{\boldsymbol{\beta}}^{[0]} + \tilde{\boldsymbol{V}}_{(s+1)}[\tilde{\boldsymbol{X}}^T(10^{\phi}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[0]}) + 10^{\phi}\tilde{\boldsymbol{V}}_{(s)}^{-1}(\tilde{\boldsymbol{m}}_{(s-1)} - \tilde{\boldsymbol{\beta}}^{[0]})] \quad [3] \\
&= 10^{3\phi}10^{\phi}\boldsymbol{\beta}^{[0]} + 10^{\phi}\boldsymbol{V}_{(s+1)}[10^{3\phi}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[0]}) + 10^{3\phi}\boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[0]})] \\
&= 10^{4\phi}\{\boldsymbol{\beta}^{[0]} + \boldsymbol{V}_{(s+1)}[\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[0]}) + \boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[0]})]\} \\
&\equiv 10^{4\phi}\boldsymbol{\beta}^{[1]}
\end{aligned} \tag{B.27}$$

For $k = 2$:

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[2]} &\equiv 10^{3\phi}\tilde{\boldsymbol{\beta}}^{[1]} + \tilde{\boldsymbol{V}}_{(s+1)}[\tilde{\boldsymbol{X}}^T(10^{4\phi}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[1]}) + 10^{\phi}\tilde{\boldsymbol{V}}_{(s)}^{-1}(10^{3\phi}\tilde{\boldsymbol{m}}_{(s-1)} - \tilde{\boldsymbol{\beta}}^{[1]})] \quad [6] \\
&= 10^{3\phi}10^{4\phi}\boldsymbol{\beta}^{[1]} + 10^{\phi}\boldsymbol{V}_{(s+1)}[10^{6\phi}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[1]}) + 10^{6\phi}\boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[1]})] \\
&= 10^{7\phi}\{\boldsymbol{\beta}^{[1]} + \boldsymbol{V}_{(s+1)}[\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[1]}) + \boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[1]})]\} \\
&\equiv 10^{7\phi}\boldsymbol{\beta}^{[2]}
\end{aligned} \tag{B.28}$$

For $k = 3$:

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[3]} &\equiv 10^{3\phi}\tilde{\boldsymbol{\beta}}^{[2]} + \tilde{\boldsymbol{V}}_{(s+1)}[\tilde{\boldsymbol{X}}^T(10^{7\phi}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[2]}) + 10^{\phi}\tilde{\boldsymbol{V}}_{(s)}^{-1}(10^{6\phi}\tilde{\boldsymbol{m}}_{(s-1)} - \tilde{\boldsymbol{\beta}}^{[2]})] \quad [9] \\
&= 10^{3\phi}10^{7\phi}\boldsymbol{\beta}^{[2]} + 10^{\phi}\boldsymbol{V}_{(s+1)}[10^{9\phi}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[2]}) + 10^{9\phi}\boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[2]})] \\
&= 10^{10\phi}\{\boldsymbol{\beta}^{[2]} + \boldsymbol{V}_{(s+1)}[\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[2]}) + \boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[2]})]\} \\
&\equiv 10^{10\phi}\boldsymbol{\beta}^{[3]}
\end{aligned} \tag{B.29}$$

By induction we obtain the following general expression:

$$\begin{aligned}
\tilde{\boldsymbol{\beta}}^{[k]} &\equiv 10^{3\phi}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{V}}_{(s+1)}[\tilde{\boldsymbol{X}}^T(10^{(3(k-1)+1)\phi}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}}^{[k-1]}) \\
&\qquad + 10^{\phi}\tilde{\boldsymbol{V}}_{(s)}^{-1}(10^{3(k-1)\phi}\tilde{\boldsymbol{m}}_{(s-1)} - \tilde{\boldsymbol{\beta}}^{[k-1]})] \quad [3k] \\
&= 10^{3\phi}10^{(3(k-1)+1)\phi}\boldsymbol{\beta}^{[k-1]} + 10^{\phi}\boldsymbol{V}_{(s+1)}[10^{3k\phi}\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) \\
&\qquad + 10^{3k\phi}\boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[k-1]})] \\
&= 10^{(3k+1)\phi}\{\boldsymbol{\beta}^{[k-1]} + \boldsymbol{V}_{(s+1)}[\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}^{[k-1]}) + \boldsymbol{V}_{(s)}^{-1}(\boldsymbol{m}_{(s-1)} - \boldsymbol{\beta}^{[k-1]})]\} \\
&\equiv 10^{(3k+1)\phi}\boldsymbol{\beta}^{[k+1]}
\end{aligned} \tag{B.30}$$

Each party s repeats this process for K iterations using the corresponding transformed inputs, or approximations, $\{\tilde{\boldsymbol{X}}_{(s)}, \tilde{\boldsymbol{y}}_{(s)}, \tilde{\boldsymbol{m}}'_{(s-1)}, \tilde{\boldsymbol{V}}'^{-1}_{(s)}, \tilde{\boldsymbol{V}}'_{(s+1)}\}$, to obtain $\tilde{\boldsymbol{m}}'_{(s)} = \tilde{\boldsymbol{\beta}}^{[K]}$. Note that, during implementation, constant factors multiplying transformed variables sent by the master to the clients can be multiplied before encryption, provided that they are iteration-independent and data-independent. For instance, the master can encrypt $10^{\phi}\tilde{\boldsymbol{V}}_{(s)}^{-1}$ instead of $\tilde{\boldsymbol{V}}_{(s)}^{-1}$, implying a savings in terms of homomorphic operations.

B.6 Cryptographic parameters: `ECBLR-QN`

Similarly to §B.3, we can derive the maximum coefficient size (MCS; `max`) for the `ECBLR-QN` algorithm as given in (5.21). Recall the rules in (B.14) and the definition $n \equiv (\phi+1)\log_2(10)$.

Maximum coefficient size

The first iteration, starting at $\tilde{\boldsymbol{\beta}}^{[0]} = 0$, simplifies to

$$\begin{aligned}\tilde{\boldsymbol{\beta}}^{[1]} &\equiv \tilde{\boldsymbol{V}}_{(s+1)}[10^{\phi}\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}} + 10^{\phi}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}] \\ &= \underbrace{10^{\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}}_{\textbf{term-1.1}} + \underbrace{10^{\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}}_{\textbf{term-1.2}}\end{aligned} \tag{B.31}$$

For **term-1.1** we have: (a) $||\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}||_\infty \leq (n+1)N$; (b) $||\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}||_\infty \leq (n+1)N(n+1)P = (n+1)^2NP$; (c) $||10^{\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{y}}||_\infty \leq \log_2(10^{\phi})(n+1)^2NP \leq (\phi+1)\log_2(10)(n+1)^2NP = n(n+1)^2NP$.

For **term-1.2** we have: (a) $||\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}||_\infty \leq (n+1)P$; (b) $||\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}||_\infty \leq (n+1)P(n+1)P = (n+1)^2P^2$; (c) $||10^{\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}||_\infty \leq \phi\log_2(10)(n+1)^2P^2 \leq n(n+1)^2P^2$.

Putting the two terms together we get $||\tilde{\boldsymbol{\beta}}^{[1]}||_\infty \leq n(n+1)^2NP + n(n+1)^2P^2 = n(n+1)^2P(P+N)$.

For the following iterations,

$$\begin{aligned}\tilde{\boldsymbol{\beta}}^{[k]} &\equiv 10^{3\phi}\tilde{\boldsymbol{\beta}}^{[k-1]} + \tilde{\boldsymbol{V}}_{(s+1)}[\tilde{\boldsymbol{X}}_{(s)}^T(10^{(3k-2)\phi}\tilde{\boldsymbol{y}}_{(s)} - \tilde{\boldsymbol{X}}_{(s)}\tilde{\boldsymbol{\beta}}^{[k-1]}) \\ &\quad + 10^{\phi}\tilde{\boldsymbol{V}}_{(s)}^{-1}(10^{3(k-1)\phi}\tilde{\boldsymbol{m}}_{(s-1)} - \tilde{\boldsymbol{\beta}}^{[k-1]})] \\ &= \underbrace{10^{3\phi}\tilde{\boldsymbol{\beta}}^{[k-1]}}_{\textbf{term-k.1}} + \underbrace{10^{(3k-2)\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{X}}_{(s)}^T\tilde{\boldsymbol{y}}_{(s)}}_{\textbf{term-k.2}} + \underbrace{\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{X}}_{(s)}^T\tilde{\boldsymbol{X}}_{(s)}\tilde{\boldsymbol{\beta}}^{[k-1]}}_{\textbf{term-k.3}} \\ &\quad + \underbrace{10^{(3k-2)\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}}_{\textbf{term-k.4}} + \underbrace{10^{(3k-2)\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{\beta}}^{[k-1]}}_{\textbf{term-k.5}}\end{aligned} \tag{B.32}$$

For **term-k.1** we have: (a) $||10^{3\phi}\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq 3\phi\log_2(10)||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq 3n\cdot||\tilde{\boldsymbol{\beta}}^{[k-1]}||$.

For **term-k.2** we have: (a) $||\tilde{\boldsymbol{X}}_{(s)}^T\tilde{\boldsymbol{y}}_{(s)}||_\infty \leq (n+1)N$; (b) $||\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{X}}_{(s)}^T\tilde{\boldsymbol{y}}_{(s)}||_\infty \leq (n+1)N(n+1)P = (n+1)^2NP$; (c) $||10^{(3k-2)\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{X}}_{(s)}^T\tilde{\boldsymbol{y}}_{(s)}||_\infty \leq (3k-2)\phi\log_2(10)(n+1)^2NP \leq (3k-2)n(n+1)^2NP$;

For **term-k.3** we have: (a) $||\tilde{\boldsymbol{X}}_{(s)}\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq (n+1)P \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$; (b) $||\tilde{\boldsymbol{X}}_{(s)}^T\tilde{\boldsymbol{X}}_{(s)}\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq (n+1)N(n+1)P \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty = (n+1)^2NP \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$; (c) $||\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{X}}_{(s)}^T\tilde{\boldsymbol{X}}_{(s)}\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq (n+1)P(n+1)^2NP \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty = (n+1)^3NP^2 \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$;

For **term-k.4** we have: (a) $||\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}||_\infty leq (n+1)P$; (b) $||\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}||_\infty leq (n+1)P(n+1)P = (n+1)^2P^2$; (c) $||10^{(3k-2)\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{m}}_{(s-1)}||_\infty \leq (3k-2)\phi\log_2(10)(n+1)^2P^2 \leq (3k-2)n(n+1)^2P^2$;

For **term-k.5** we have: (a) $||\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq (n+1)P \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$; (b) $||\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq (n+1)P(n+1)P \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty = (n+1)^2P^2 \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$; (c) $||10^{(3k-2)\phi}\tilde{\boldsymbol{V}}_{(s+1)}\tilde{\boldsymbol{V}}_{(s)}^{-1}\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq (3k-2)\phi\log_2(10)(n+1)^2P^2 \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \leq (3k-2)n(n+1)^2P^2 \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty$.

Finally, $||\tilde{\boldsymbol{\beta}}^{[k]}||_\infty$ is equal to the sum of these 5 terms:

$$\begin{aligned} ||\tilde{\boldsymbol{\beta}}^{[k]}||_\infty &\leq 3n \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}|| + (3k-2)n(n+1)^2NP + (n+1)^3NP^2 \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \\ &\quad + (3k-2)n(n+1)^2P^2 + (3k-2)n(n+1)^2P^2 \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \\ &= (3k-2)n(n+1)^2P(P+N) \\ &\quad + [3n + (n+1)^3NP^2 + (3k-2)n(n+1)^2P^2] \cdot ||\tilde{\boldsymbol{\beta}}^{[k-1]}||_\infty \end{aligned} \tag{B.33}$$

which gives a recursion for the upper bound on the MCS of $\tilde{\boldsymbol{\beta}}^{[k]}$ at any iteration $k \in \mathbb{N}_{1:K}$, starting at $||\tilde{\boldsymbol{\beta}}^{[1]}||_\infty \leq n(n+1)^2P(P+N)$.

A bound on the cyclotomic polynomial degree follows from the alternative method derived in §B.3.

Bibliography

A. Abbas and S. U. Khan. A review on the state-of-the-art privacy-preserving approaches in the e-health clouds. *IEEE Journal of Biomedical and Health Informatics*, 18(4):1431–1441, 2014.

R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Very Large Data Bases (VLDB'02)*, pages 143–154. VLDB Endowment, 2002.

D. Allen. The relationship between variable selection and data agumentation and a method for prediction. *Technometrics*, 16(1):125–127, 1974.

M. R. Anderlik and M. A. Rothstein. Privacy and confidentiality of genetic information: what rules for the new science? *Annual Review of Genomics and Human Genetics*, 2:401–433, 2001.

Y. Aono, T. Hayashi, L. T. Phong, and L. Wang. Fast and secure linear regression and biometric authentication with security update. IACR Cryptology ePrint Archive, Report 2015/692: eprint.iacr.org/2015/692, 2015.

S. Arita and S. Nakasato. Fully homomorphic encryption for point numbers. IACR Cryptology ePrint Archive, Report 2016/402: eprint.iacr.org/2016/402, 2016.

M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand. A guide to fully homomorphic encryption. IACR Cryptology ePrint Archive, Report 015/1192: eprint.iacr.org/015/1192, 2015.

L. J. Aslett, P. M. Esperança, and C. C. Holmes. A review of homomorphic encryption and software tools for encrypted statistical machine learning. arXiv:1508.06574: arxiv.org/abs/1508.06574, 2015a.

L. J. Aslett, P. M. Esperança, and C. C. Holmes. Encrypted statistical machine learning: new privacy preserving methods. arXiv:1508.06845: arxiv.org/abs/1508.06845, 2015b.

A. A. Atayero and O. Feyisetan. Security issues in cloud computing: The potentials of homomorphic encryption. *Journal of Emerging Trends in Computing and Information Sciences*, 2(10):546–552, 2011.

S. Audrey, L. Brown, R. Campbell, A. Boyd, and J. Macleod. Young people's views about consenting to data linkage: findings from the PEARL qualitative study. *BMC Medical Research Methodology*, 16(1):1–13, 2016.

K. Austen. The trouble with wearables. *Nature*, 525(7567):22–24, 2015.

E. Ayday, J. L. Raisaro, J. Hubaux, and J. Rougemont. Protecting and evaluating genomic privacy in medical tests and personalized medicine. In *Workshop on Privacy in the Electronic Society (WPES'13)*, pages 95–106. ACM, 2013a.

E. Ayday, J. L. Raisaro, P. J. McLaren, J. Fellay, and J. Hubaux. Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data. In *Workshop on Health Information Technilogies (HealthTech'13)*. USENIX, 2013b.

K. Barker, M. Askari, M. Banerjee, K. Ghazinour, B. Mackas, M. Majedi, S. Pun, and A. Williams. A data privacy taxonomy. In *Dataspace: The Final Frontier*, volume 5588 of *Lecture Notes in Computer Science*, pages 42–54. Springer, 2009.

M. Barni, G. Droandi, , and R. Lazzeretti. Privacy protection in biometric-based recognition systems: a marriage between cryptography and signal processing. *Signal Processing Magazine*, 32(5):66–76, 2015.

J. Basilakis, B. Javadi, and A. Maeder. The potential for machine learning analysis over encrypted data in cloud-based clinical decision support — background and review. In *Health Informatics and Knowledge Management (HIKM'15)*, volume 164 of *Conferences in Research and Practice in Information Technology*, pages 3–13. ACS, 2015.

A. H. Beck. Open access to large scale datasets is needed to translate knowledge of cancer heterogeneity into better patient outcomes. *PLoS Medicine*, 12(2):1–4, 2015.

J. Benaloh. Dense probabilistic encryption. In *Workshop on Selected Areas of Cryptography (SAC'94)*, pages 120–128. Springer, 1994.

J. Benaloh, M. Chase, E. Horvitz, and K. Lauter. Patient controlled encryption: ensuring privacy of electronic medical records. In *Cloud Computing Security Workshop (CCSW'09)*, pages 103–114. ACM, 2009.

Åke. Björck. *Numerical methods for least squares problems*. SIAM, 1996.

M. Bobrow. Funders must encourage scientists to share. *Nature*, 552(7555):129, 2015.

D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society*, 46(2): 203–213, 1999.

D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography (TCC'05)*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography (TCC'11)*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

D. Boneh, A. Sahai, and B. Waters. Functional encryption: A new vision for public-key cryptography. *Communications of the ACM*, 55(11):56–64, 2012.

M. Bonsall, S. Wallace-Hadrill, J. Geddes, G. Goodwin, and E. Holmes. Nonlinear time-series approaches in characterizing mood stability and mood instability in bipolar disorder. *Proceedings of the Royal Society B (Biological Sciences)*, 279(1730):916–924, 2012.

J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. IACR Cryptology ePrint Archive, Report 2016/659: eprint.iacr.org/2016/659, 2016.

J. W. Bos, K. Lauter, and M. Naehrig. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics*, 50:234–243, 2014.

R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. Cryptology ePrint Archive, Report 2014/331: eprint.iacr.org/2014/331, 2014.

Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology (CRYPTO'12)*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science Conference (ITCS'12)*, pages 309–325. ACM, 2012.

Z. Brakerski, C. Gentry, and S. Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In *Public-Key Cryptography (PKC'13)*, volume 7778 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2013.

P. T. Breuer and J. P. Bowen. A fully encrypted microprocessor the: Secret computer is nearly here. *Procedia Computer Science*, 83:1282–1287, 2016.

N. Busom, R. Petrlic, F. Sebé, C. Sorge, and M. Valls. Efficient smart metering based on homomorphic encryption. *Computer Communications*, 82:95–101, 2016.

S. Carpov and R. Sirdey. Another compression method for homomorphic ciphertexts. In *Security in Cloud Computing (SCC'16)*, pages 44–50. ACM, 2016.

G. S. Çetin, W. Dai, Y. Doröz, W. J. Martin, and B. Sunar. Blind web search: How far are we from a privacy preserving search engine? IACR Cryptology ePrint Archive, Report 2016/801: eprint.iacr.org/2016/801, 2016.

K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems (NIPS'08)*, pages 289–296, 2009.

D. D. Chen, eleMenten, F. Vercauteren, S. S. Roy, R. C. C. Cheung, D. Pao, and I. Verbauwhede. High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems. *IEEE Transactions on Circuit and Systems*, 62 (1):157–166, 2015.

J. H. Cheon, A. Kim, M. Kim, , and Y. Song. Floating-point homomorphic encryption. IACR Cryptology ePrint Archive, Report 2016/421: eprint.iacr.org/2016/421, 2016.

A. Choudhury, J. Loftus, E. Orsini, A. Patra, and N. P. Smart. Between a rock and a hard place: Interpolating between MPC and FHE. IACR Cryptology ePrint Archive, Report 2013/085: eprint.iacr.org/2013/085, 2013.

A. Cilardo and D. Argenziano. Securing the cloud with reconfigurable computing: An FPGA accelerator for homomorphic encryption. In *Design, Automation and Test in Europe (DATE'16)*, pages 1622–1627. IEEE, 2016.

A. Costache, N. P. Smart, S. Vivek, and A. Waller. Fixed point arithmetic in SHE schemes. IACR Cryptology ePrint Archive, Report 2015/889: eprint.iacr.org/2015/889, 2016.

W. Dai and B. Sunar. cuHE: A homomorphic encryption accelerator library. IACR Cryptology ePrint Archive, Report 2015/818: eprint.iacr.org/2015/818, 2015.

I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology (CRYPTO'12)*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

F. K. Dankar and K. El Emam. Practicing differential privacy in health care: a review. *Transactions on Data Privacy*, 5:35–67, 2013.

Y. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3:1376, 2013.

Y. de Montjoye, L. Radaelli, V. K. Singh, and A. S. Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, 2015.

J. Domingo-Ferrer and V. Torra. A critique of k-anonymity and some of its enhancements. In *Availability, Reliability and Security (ARES'08)*, pages 990–993. IEEE, 2008.

P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3):103–130, 1997.

N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. Technical Report: web.math.princeton.edu/ ndowlin/website-assets/Cryptonets.pdf, 2015.

C. Du, G. Bai, and X. Wu. High-speed polynomial multiplier architecture for ring-LWE based public key cryptosystems. In *Great Lakes Symposium on VLSI (GLSVLSI'16)*, pages 9–14. ACM, 2016.

W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *New Security Paradigms Workshop (NSPW'01)*, pages 13–22. ACM, 2001.

L. Ducas and A. Durmus. Ring-LWE in polynomial rings. In *Public Key Cryptography (PKC'12)*, volume 7293 of *Lecture Notes in Computer Science*, pages 34–51. Springer, 2012.

C. Dwork. Differential privacy. In *Automata, Languages and Programming (ICALP'06)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science. Now Publisher, 2014.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

K. El Emam, E. Jonker, L. Arbuckle, and B. Malin. A systematic review of re-identification attacks on health data. *PLoS One*, 6(12):e28071, 2011.

T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology (CRYPTO'84)*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1985.

M. Enserink and G. Chin. The end of privacy. *Science*, 347(6221):453–580, 2015.

J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, Report 2012/144: eprint.iacr.org/2012/144, 2012.

J. L. Fernández-Alemán, I. C. S. nor, P. Ángel Oliver Lozoya, and A. Toval. Security and privacy in electronic health records: A systematic literature review. *Journal of Biomedical Informatics*, 46(3):541–562, 2013.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(10):3133–3181, 2014.

L. Fousse et al. *GNU MPFR: The GNU The Multiple Precision Floating-Point Reliable Library*, 3.1.4 edition, 2016. mpfr.org.

W. J. Fu. Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3): 397–416, 1998.

W. Fulton. Eigenvalues, invariant factors, highest weights, and Schubert calculus. *Bulletin of the American Mathematical Society*, 37(3):209–249, 2000.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 2nd edition, 2004.

C. Gentry. Fully homomorphic encryption using ideal lattices. In *Symposium on Theory of Computing (STOC'09)*, pages 169–178. ACM, 2009a.

C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009b. Available at: crypto.stanford.edu/craig/craig-thesis.pdf.

C. Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010.

C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology (EUROCRYPT'11)*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.

C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology (EUROCRYPT'12)*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012a.

C. Gentry, S. Halevi, and N. P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography (PKC'12)*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012b.

C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology (CRYPTO'13)*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

J. Ghosh and J. P. Reiter. Secure bayesian model averaging for horizontally partitioned data. *Statistics and Computing*, 23(3):311–322, 2013.

S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Symposium on Theory of Computing (STOC'82)*, pages 365–377. ACM, 1982.

T. Graepel, K. Lauter, and M. Naehrig. ML Confidential: Machine learning on encrypted data. In *Information Security and Cryptology (ICISC'12)*, volume 7839 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2013.

T. Granlund et al. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.1.1 edition, 2016. gmplib.org.

E. Green, M. Guyer, and National Human Genome Research Institute. Charting a course for genomic medicine from base pairs to bedside. *Nature*, 470(7333):204–213, 2011.

D. Greenbaum, A. Sbone, X. J. Mu, and M. Gerstein. Genomics and privacy: Implications of the new reality of closed data for the field. *PLoS Computational Biology*, 7(12):e1002278, 2011.

M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

S. Halevi and V. Shoup. Algorithms in `HElib`. In *Advances in Cryptology (CRYPTO'14)*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2014.

S. Halevi and V. Shoup. Bootstrapping for `HElib`. In *Advances in Cryptology (EUROCRYPT'15)*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670. Springer, 2015.

R. Hall, S. E. Fienberg, and Y. Nardi. Secure multiple linear regression based on homomorphic encryption. *Journal of Official Statistics*, 27(4):669–691, 2011.

J. D. Hamilton. *Time series analysis*, volume 2. Princeton University Press, 1994.

D. J. Hand and K. Yu. Idiot's Bayes — not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

W. Hart, F. Johansson, and S. Pancratz. *FLINT: Fast Library for Number Theory*, 2.4.5 edition, 2016. flintlib.org.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, New York, NY, USA, 2009.

E. C. Hayden. Informed consent: a broken contract. *Nature*, 486(7403):312–314, 2013.

R. Hazin, K. B. Brothers, B. A. Malin, B. A. Koenig, S. C. Sanderson, M. A. Rothstein, M. S. Williams, E. W. Clayton, and I. J. Kullo. Ethical, legal, and social implications of incorporating genomic information into electronic health records. *Genetics in Medicine*, 15(10):810–816, 2013.

R. Heatherly. Privacy and security within biobanking: The role of information technology. *The Journal of Law, Medicine & Ethics*, 44(1):156–160, 2016.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *TheJournal*, Technometrics(12):1, 1970.

E. Holmes, M. Bonsall, S. Hales, H. Mitchell, F. Renner, S. Blackwell, P. Watson, G. Goodwin, and M. Di Simplicio. Applications of time-series analysis to mood fluctuations in bipolar disorder to promote treatment innovation: a case series. *Translational Psychiatry*, 6:e720, 2016.

L. Jamal, J. C. Sapp, K. Lewis, T. Yanes, F. M. Facio, L. G. Biesecker, and B. B. Biesecker. Research participants' attitudes towards the confidentiality of genomic sequence information. *European Journal of Human Genetics*, 22 (8):964–968, 2014.

P. B. Jensen, L. J. Jensen, and S. Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405, 2012.

Z. Ji, X. Jiang, S. Wang, L. Xiongn, and L. Ohno-Machado. Differentially private distributed logistic regression using private and public data. *BMC Medical Genomics*, 7(Suppl 1):S14, 2014.

D. J. Kaufman, J. Murphy-Bollinger, J. Scott, and K. L. Hudson. Public opinion about the importance of privacy in biobank research. *American Journal of Human Genetics*, 85(5):643–654, 2009.

G. C. Kessler. An overview of cryptography. Computer Security Commons: commons.erau.edu/publication/127, 2016.

A. Khedr and G. Gulak. SecureMed: Secure medical computation using gpu-accelerated homomorphic encryption scheme. IACR Cryptology ePrint Archive, Report 2016/445: eprint.iacr.org/2016/445, 2016.

D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *International Conference on Management of data (SIGMOD'11)*, pages 193–204. ACM, 2011.

M. Kim and K. Lauter. Private genome analysis through homomorphic encryption. *BMC Medical Informatics and Decision Making*, 15(Suppl 5):S3, 2015.

O. Kocabaş and T. Soyata. Utilizing homomorphic encryption to implement secure and private medical cloud computing. In *Cloud Computing (CLOUD'15)*, pages 540–547. IEEE, 2015.

O. Kocabaş, T. Soyata, J. Couderc, M. Aktas, J. Xia, and M. Huang. Assessment of cloud-based health monitoring using homomorphic encryption. In *Computer Design (ICCD'13)*, pages 443–446. IEEE, 2013.

R. Lagendijk, Z. Erkin, and M. Barni. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine*, 30(1):82–105, 2013.

K. Laine and R. Player. Simple Encrypted Arithmetic Library - SEAL (v2.0). Microsoft Research Technical Report MSR-TR-2016-52: www.microsoft.com/en-us/research/wp-content/uploads/2016/09/sealmanual.pdf, 2016.

K. Lauter, M. Naehrig, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Cloud computing security workshop (CCSW'11)*, pages 113–124. ACM, 2011.

K. Lauter, A. López-Alt, and M. Naehrig. Private computation on encrypted genomic data. In *Progress in Cryptology (LATINCRYPT'14)*, volume 8895 of *Lecture Notes in Computer Science*, pages 3–27. Springer, 2014.

M. S. Lee, Y. Lee, J. H. Cheon, and Y. Paek. Accelerating bootstrapping in FHEW using GPUs. In *Application-specific Systems, Architectures and Processors (ASAP'15)*, pages 128–135. IEEE, 2015.

T. Lepoint and M. Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In *Progress in Cryptology (AFRICACRYPT'14)*, volume 8469 of *Lecture Notes in Computer Science*, pages 318–335. Springer,

2014.

M. Lichman. UCI Machine Learning Repository. archive.ics.uci.edu/ml/index.html, 2013.

Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *The Journal of Privacy and Confidentiality*, 1(1):59–98, 2009.

R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Topics in Cryptology (CT-RSA'11)*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.

H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discover*, 6(4):393–423, 2002.

X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin. Privacy-preserving clinical decision support system using gaussian kernel-based classification. *IEEE Journal of Biomedical and Health Informatics*, 20(2):655–668, 2016.

A. López-Alt. *Cryptographic Algorithms for the Secure Delegation of Multiparty Computation*. PhD thesis, New York University, 2014. www.cs.nyu.edu/media/publications/AdrianaLopezalt.pdf.

V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *Advances in Cryptology (EUROCRYPT'10)*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *Advances in Cryptology (EUROCRYPT'13)*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013.

L. Malina, J. Hajny, R. Fujdiak, and J. Hosek. On perspective of security and privacy-preserving solutions in the internet of things. *Computer Networks*, 102:83–95, 2016.

R. Matovu and A. Serwadda. Your substance abuse disorder is an open secret! Gleaning sensitive personal information from templates in an EEG-based authentication system. In *Biometrics: Theory, Applications, and Systems (BTAS'16)*. IEEE, 2016.

J. Mayer, P. Mutchler, and J. C. Mitchell. Evaluating the privacy properties of telephone metadata. *PNAS*, 113(20): 5536–5541, 2016.

D. Micciancio. A first glimpse of cryptography's holy grail. *Communications of the ACM*, 53(3):96, 2010.

D. Micciancio and O. Regev. Lattice-based cryptography. In D. J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, 2009.

M. F. M. Mursi, G. M. R. Assassa, A. Abdelhafez, and K. M. A. Samra. On the development of electronic voting: A survey. *International Journal of Computer Applications*, 61(16):1–11, 2013.

A. Narayanan and V. Shmatikov. Robust de-anonymization of large datasets (how to break anonymity of the netflix prize dataset). arXiv:0610105: arxiv.org/abs/0610105, 2006.

A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Symposium on Security and Privacy (SP'08)*, pages 111–125. IEEE, 2008.

A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Symposium on Security and Privacy (SSP'09)*, pages 173–187. IEEE, 2009.

A. Narayanan and V. Shmatikov. Myths and fallacies of "personally identifiable information". *Communications of the ACM*, 53(6):24–26, 2010.

M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang. Privacy in the genomic era. *ACM Computing Surveys*, 48(1):Article 6, 2015.

S. Nepal, R. Ranjan, and K. R. Choo. Trustworthy processing of healthcare big data in hybrid clouds. *Cloud Computing*, 2(2):78–84, 2015.

Y. Nesterov. A method for solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS'01)*, volume 14, pages 841–848. MIT Press, 2002.

A. O'Hagan and J. Forster. *Kendall's Advanced Theory of Statistics, volume 2B: Bayesian Inference*. Arnold, 2nd edition, 2004.

P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Review*, 57 (6):1701–1777, 2010.

A. Oppermann, J.-P. Seifert, and F. Thiel. Secure cloud reference architectures for measuring instruments under legal control. In *Cloud Computing and Services Science (CLOSER'16)*, pages 289–294. SCITEPRESS Digital Library, 2016.

E. Öztürk, Y. Doröz, B. Sunar, and E. Savaş. Accelerating somewhat homomorphic evaluation using FPGA. IACR Cryptology ePrint Archive, Report 2015/294: eprint.iacr.org/2015/294, 2015.

P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology (EUROCRYPT'99)*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

G. Paul and J. Irvine. Privacy implications of wearable health devices. In *Security of Information and Networks (SIN'14)*, pages 117–121. ACM, 2014.

C. Peikert. Lattice cryptography for the internet. In *Post-Quantum Cryptography (PQCrypto'14)*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219. Springer, 2014.

A. Rajkumar and S. Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *Artificial Intelligence and Statistics (AISTATS'12)*, pages 933–941. JMLR, 2012.

O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Symposium on Theory of Computing (STOC'05)*, pages 84–93. ACM, 2005.

O. Regev. The learning with errors problem. In *Computational Complexity (CCC'10)*, pages 191–204. IEEE, 2010.

E. C. Reinke. Classical cryptography. *The Classical Journal*, 58(3):113–121, 1962.

J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *International Conference on Machine Learning (ICML'03)*, pages 616–623. AAAI, 2003.

R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180. Academic Press, 1978a.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978b.

D. Rogith, R. A. Yusuf, S. R. Hovick, S. K. Peterson, A. M. Burton-Chase, Y. Li, and F. Meric-Bernstamand Elmer V Bernstam. Attitudes regarding privacy of genomic information in personalized cancer therapy. *Journal of the American Medical Informatics Association*, 21(e2):e320–e325, 2014.

S. S. Roy, K. Järvinen, F. Vercauteren, V. Dimitrov, and I. Verbauwhede. Modular hardware architecture for somewhat homomorphic function evaluation. In *Cryptographic Hardware and Embedded Systems (CHES'15)*, volume 9293 of *Lecture Notes in Computer Science*, pages 164–184. Springer, 2015.

A. J. Rush, M. H. Trivedi, H. M. Ibrahim, T. J. Carmody, B. Arnow, D. N. Klein, J. C. M. nad Philip T. Ninan, S. Kornstein, R. Manber, M. E. Thase, J. H. Kocsis, and M. B. Keller. The 16-item Quick Inventory of Depressive Symptomatology (QIDS), Clinician Rating (QIDS-C), and Self-Report (QIDS-SR): A psychometric evaluation in patients with chronic major depression. *Biological Psychiatry*, 54(5):573–583, 2003.

V. S. Ryaben'kii and S. V. Tsynkov. *A Theoretical Introduction to Numerical Analysis*. Chapman and Hall/CRC, 2006.

S. Saeb, M. Zhang, J. C. Karr, M. S. Schueller, E. M. Corden, P. K. Kording, and C. D. Mohr. Mobile phone sensor correlates of depressive symptom severity in daily-life behavior: An exploratory study. *Journal of Medical Internet Research*, 17(7):e175, 2015.

S. S. Sahoo, C. Jayapandian, G. Garg, F. Kaffashi, S. Chung, A. Bozorgi, C. Chen, K. Loparo, S. D. Lhatoo, and G. Zhang. Heart beats in the cloud: distributed analysis of electrophysiological 'big data' using cloud computing for epilepsy clinical research. *Journal of the American Medical Informatics Association*, 21(2):263–271, 2014.

S. C. Sanderson, M. D. Linderman, S. A. Suckiel, G. A. Diaz, R. E. Zinberg, K. Ferryman, M. Wasserstein, A. Kasarskis, and E. E. Schadt. Motivations, concerns and preferences of personal genome sequencing research participants: baseline findings from the HealthSeq project. *European Journal of Human Genetics*, 24:14–20, 2015.

X. Shi and X. Wu. An overview of human genetic privacy. *Annals of the New York Academy of Sciences*, early publication online, 2016.

A. Silverberg. Fully homomorphic encryption for mathematicians. IACR Cryptology ePrint Archive, Report 2013/250: eprint.iacr.org/2013/250, 2013.

N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1): 420–443, 2014.

S. Smith. Data and privacy: Now you see me, now you don't. *Significance*, 11(4):10–15, 2014.

P. Snyder. Yao's garbled circuits: Recent directions and implementations. www.cs.uic.edu/Bits/PeterSnyder, 2014.

S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP'13)*, pages 245–248. IEEE, 2013.

T. Stamey, J. Kabalin, J. McNeal, I. Johnstone, F. Freiha, E. Redwine, and N. Yang. Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate. II. radical prostatectomy treated patients. *Journal of Urology*, 141(5):1076–1083, 1989.

E. Strickland. Wireless earbuds will record your EEG, send brainwave data to your phone. IEEE Spectrum: spectrum.ieee.org, 2016.

L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

L. Sweeney. Matching known patients to health records in Washington state data. White Paper 1089-1, Data Privacy Lab, Harvard University: dataprivacylab.org/projects/wa/1089-1.pdf, 2013.

L. Sweeney, A. Abu, and J. Winn. Identifying participants in the Personal Genome Project by name. White Paper 1021-1, Data Privacy Lab, Harvard University: dataprivacylab.org/projects/pgp/1021-1.pdf, 2013.

Z. Tari, X. Yi, U. S. Premarathne, P. Bertok, and I. Khalil. Security and privacy in cloud computing: Vision, trends, and challenges. *Cloud Computing, IEEE*, 2(2):30–38, 2015.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

E. J. Topol. The big medical data miss: challenges in establishing an open medical resource. *Nature Reviews Genetics*, 16(5):253–254, 2015.

J. Vaidya, B. Shafiq, A. Basu, and Y. Hong. Differentially private naive Bayes classification. In *Web Intelligence and Intelligent Agent Technologies (WI-IAT'13)*, pages 571–576. IEEE, 2013.

V. Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In *Foundations of Computer Science (FOCS'11)*, pages 5–16. IEEE, 2011.

M. van Dijk and A. Juels. On the impossibility of cryptography alone for privacy-preserving cloud computing. In *Hot Topics in Security (HotSec'10)*, pages 1–8. USENIX, 2010.

M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology (EUROCRYPT'10)*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

A. van Wijngaarden. Cursus: Wetenschappelijk Rekenen B: Process Analyse, 1965.

R. S. Varga. *Matrix iterative analysis*, volume 27 of *Springer Series in Computational Mathematics*. Springer, 2nd edition, 2000.

P. Waggett. Risk-based authentication: biometrics' brave new world. *Biometric Technology Today*, 2016(6):5–7, 2016.

W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar. Accelerating fully homomorphic encryption using GPU. In *High Performance Extreme Computing (HPEC'12)*, pages 1–5. IEEE, 2012.

W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar. Exploring the feasibility of fully homomorphic encryption. *IEEE Transactions on Computers*, 64(3):698–706, 2015a.

Y. Wang, C. Si, and X. Wu. Regression model fitting under differential privacy and model inversion attack. In *International Joint Conference on Artificial Intelligence (IJCAI'15)*, pages 25–31. AAAI Press, 2015b.

S. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

D. J. Wu. Fully homomorphic encryption: Cryptography's holy grail. *XRDS: Crossroads*, 21(3):24–29, 2015.

D. J. Wu and J. Haven. Using homomorphic encryption for large scale statistical analysis. Technical Report: cs.stanford.edu/people/dwu4/papers/FHE-SI_Report.pdf., 2012.

D. J. Wu, T. Feng, M. Naehrig, and K. Lauter. Privately evaluating decision trees and random forests. *Proceedings on Privacy Enhancing Technologies*, 2016(4):1–21, 2016.

A. C. Yao. Protocols for secure computations. In *Symposium on Foundations of Computer Science (FOCS'82)*, pages 160–164. IEEE, 1982.

M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. Packed homomorphic encryption based on ideal lattices and its application to biometrics. In *Security Engineering and Intelligence Informatics*, volume 8128 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2013.

M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, and T. Koshiba. New packing method in somewhat homomorphic encryption and its applications. *Security and Communication Networks*, 8(13):2194–2213, 2015.

G. Yuan, Y. Yang, Z. Zhang, and Z. Hao. Convex optimization for linear query processing under approximate differential privacy. In *Knowledge Discovery and Data Mining (SIGKDD'16)*, pages 2005–2014. ACM, 2016.

J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional mechanism: regression analysis under differential privacy. In *Very Large Data Bases (VLDB'12)*, volume 5, pages 1364–1375. VLDB Endowment, 2012.

M. T. I. Ziad, A. Alanwar, M. Alzantot, and M. Srivastava. `CryptoImg`: Privacy preserving processing over encrypted images. arXiv:1609.00881: arxiv.org/abs/1609.00881, 2016.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Methodological)*, 67(2):301–320, 2005.

Glossary

Asymmetric cryptosystem ▷*Public key cryptosystem*

Ciphertext is an encrypted message.

Ciphertext noise is injected during encryption to guarantee semantic security. The noise grows with homomorphic operations, and if it grows too large correct decryption becomes impossible and the underlying message is lost.

Ciphertext space defines the domain of ciphertexts. For instance, in the `FV` cryptosystem this is $R_q = \mathbb{Z}_q[x]/\Phi_{2^d}(x)$, the cyclotomic polynomial ring with coefficients from an integer ring.

Circular security is a property of some FHE schemes which means that it is safe to encrypt the secret key under its own public key. It is required for Gentry's bootstrapping procedure.

Circuit depth ▷*Maximum multiplicative depth*

Circuit privacy is a property of FHE schemes which means that freshly encrypted ciphertexts and those that result from some circuit evaluation (homomorphic computation) are indistinguishable.

Client refers to an individual or organisation that provides data (e.g., a patient or research institution) to a server, and who is typically the subject of the data or acts in the subject's behalf. (see also ▷*Server*.)

Cryptographic attack is a algorithm designed to break an encryption scheme.

Cryptographic parameters are a set of parameters that regulate the security and computational properties of a cryptosystem, and are required for its instantiation.

Cryptosystem (a.k.a. encryption scheme) is a set of algorithms which together allow information to remain secure/confidential. Usually these include key generation, encryption and decryption algorithms.

Decryption is a cryptographic algorithm which transforms a ciphertext into a plaintext using a valid secret key.

Differential privacy is a notion of privacy based on perturbation methods. By perturbing the output of an algorithm, it masks the presence or absence of an individual from the dataset and thus prevents identification.

Encryption is a cryptographic algorithm which transforms a plaintext into a ciphertext using a valid public key.

Encryption scheme ▷*Cryptosystem*

Fully Homomorphic Encryption (FHE) is a form of HE which allows for unlimited homomorphic additions and multiplications such that the homomorphic principle holds.

`FV` **cryptosystem** is a modern FHE scheme based on the learning with errors problem.

Homomorphic computation denotes any computation using ciphertexts under the framework of HE, and with the correctness guarantees given by the homomorphic principle.

Homomorphic Encryption (HE) is a form of encryption which allows for some operations to be carried out directly on encrypted data (i.e., without decrypting it), such that the homomorphic principle holds.

Homomorphic operation is an algebraic operation which can be performed on ciphertexts, under a given homomorphic cryptosystem, such that the homomorphic principle holds. Note that homomorphic operations (e.g., multiplication in the ciphertext space) are generally more complex than the corresponding non-homomorphic (regular) operations (e.g., multiplication in the plaintext space) because ciphertext spaces are typically more complex. Homomorphic operations frequently involve additional complexities such as "relinearisation" or "flattening".

Homomorphic principle is a theoretical property of HE schemes which guarantees that the result of homomorphic operations applied to ciphertexts, once decrypted, match the result of equivalent operations applied to corresponding plaintexts.

Honest-but-curious party (as opposed to ▷*Malicious party*) is a type of adversary in secure computation. It is assumed that this party follows the protocols exactly as established but can make use of any information that it has access to.

Indistinguishability is a property of ciphertexts which means that they are difficult to distinguish from other ciphertexts generated under the same cryptosystem, that is, any ciphertext 'looks like' any other, independently of the message it encrypts and, in the case of homomorphic encryption, the circuit which has generated it. It is considered a basic security requirement to protect against cryptographic attacks.

Key generation is a cryptographic function which produces cryptographic keys from a set of cryptographic parameters.

Learning With Erros (LWE) is a hard problem to solve in number theory. Because of this, it serves as a foundation of many modern FHE schemes such as `FV` and can be used to prove its security.

Leveled Fully Homomorphic Encryption (leveled FHE) is a form of HE where the cryptographic parameters are set so as to allow a certain prespecified circuit to be evaluated homomorphically. It is, therefore, fully homomorphic in that it allows homomorphic addition and multiplication, but only in a restricted sense because the depth of the circuit that it can evaluate is restricted. It is a SHE scheme with parameters chosen according to the algorithm that is to be implemented homomorphically.

Malicious party (as opposed to ▷*Honest-but-curious party*) is a type of adversary in secure computation. This party can maliciously manipulate the protocols established to obtain sensitive information in an unwarranted manner.

Maximum multiplicative depth (MMD) of an algorithm is the largest number of consecutive multiplications (without other intermediate operation) that is required to evaluate that algorithm. For example, $y_1 \times y_2 \times y_3$ has $\mathtt{L} = 2$ while $\sum_{i=1}^{10} y_i^2$ has $\mathtt{L} = 1$, even though the later has a larger number of total multiplications.

Maximum coefficient size (MCS) is the largest value (in magnitude) that each coefficient of $\mathring{m}(x)$ can take, where $\mathring{m}(x)$ is the polynomial representation of plaintexts.

Plaintext is an unencrypted message.

Plaintext space defines the domain of plaintexts. For instance, in the `FV` cryptosystem this is $\mathbb{Z}_q$, the set of integers in the interval $(-q/2, q/2]$, for some $q \in \mathbb{N}$.

Private key ▷*Secret key*

Private-key cryptosystem (a.k.a. symmetric cryptosystem) is one where the encryption and decryption keys are the same.

Public key is the cryptographic key used for encryption.

Public-key cryptosystem (a.k.a. asymmetric cryptosystem) is one where there are two distinct keys, one for encryption—which can be distributed to the public—and one for decryption—which must remain private.

Ring Learning With Erros (RLWE) is the ring version of ▷*Learning With Erros.*

Secret key (a.k.a. private key) is the cryptographic key used for decryption. In private-key cryptosystems it is also used for encryption as it is the only key in that case.

Secure Multi-Party Computation (SMPC) (a.k.a. secure function evaluation) is a method in which multiple parties compute some value, based on individually held secret bits of information, without revealing their secrets to one another in the process.

Semantic security is a property of encryption schemes which means that, for a fixed public key, several encryption of the same plaintext produce different ciphertexts with extremely high probability. It is considered a basic security requirement to protect against cryptographic attacks.

Server refers to an individual or organisation that collects, stores and uses data (e.g., a cloud server) from a client, and who is typically not the subject of the data. (see also ▷*Client.*)

Somewhat Homomorphic Encryption (SHE) is a form of homomorphic encryption capable of evaluating "low-degree" circuits homomorphically before the ciphertext noise grows too large and renders decryption impossible. The specific depth depends one the cryptographic parameters.

Symmetric cryptosystem ▷*Private key cryptosystem*