Spring 5-2022

Dataset Evaluation for Data Trading Using Expected Loss and Homomorphic Encryption

Minsung Joo

Washington University in St. Louis

School of Engineering and Applied Science

Department of Computer Science and Engineering

Dataset Evaluation for Data Trading Using Expected Loss and Homomorphic Encryption

by

Minsung (Michael) Joo

A thesis presented to Washington University in partial fulfillment of the requirements for the honors of the degree of

Bachelor of Arts

April 2022
Saint Louis, Missouri

Contents

List of Tables

List of Figures

Acknowledgments

A special thanks goes to Netanel Raviv for his supervision, original idea, and support; Roman Garnett for a new scope at the problem and his Bayesian influence; Leda Liang for her help and huge contribution; Keewoo Lee for detailed discussions on HEAAN; and Youngju Kim, my mother, who first introduced me to homomorphic encryption.

Minsung (Michael) Joo

Washington University in Saint Louis
April 2022

ABSTRACT OF THE THESIS

Dataset Evaluation for Data Trading Using Expected Loss and Homomorphic Encryption

by

Minsung (Michael) Joo

Bachelor of Arts in Mathematics and Computer Science

Washington University in St. Louis, April 2022

Research Advisor: Professor Netanel Raviv

Supervised machine learning suffers from the "garbage-in garbage-out" phenomenon where the performance of a model is limited by the quality of the data. While a myriad of data is collected every second, there is no general rigorous method of evaluating the quality of a given dataset. This hinders fair pricing of data in scenarios where a buyer may look to buy data for use with machine learning. In this work, I propose using the expected loss corresponding to a dataset as a measure of its quality, relying on Bayesian methods for uncertainty quantification. Furthermore, I present a secure multi-party computation protocol with homomorphic encryption, assuming semi-honest parties, for computation of the expected loss between the buyer and the seller without compromising the data. With experimental results, I show the promise of this approach and also current limitations in real-life feasibility.

Chapter 1

Introduction

Machine learning tries to solve a problem by finding patterns in data that describe the problem (Bishop, 2006). Given the data, the performance of machine learning relies on how accurately we find patterns and how generalizable the estimated patterns are. Therefore, machine learning can succeed only if there is a valid pattern to be found in the data and if that pattern is generalizable to the problem outside the data. In other words, machine learning is only as good as the data it is given, a phenomenon referred to as "garbage-in garbage-out".

Then for one to leverage the wonders that machine learning is said to provide, one must first have informative data about their problem. Unfortunately, this is easier said than done. With a myriad of data collected every second, the problem of finding the right data becomes a daunting challenge. While simply having more data is both theoretically and empirically an answer, the cost of collecting and processing meaningful data is not insignificant. This has led to advancements in the field of active learning and semi-supervised learning that try to answer where we might try to get more data to efficiently learn more about the problem (Settles, 2009; Zhu, 2005).

In this work, I focus less on where to collect more data and instead try to answer how to evaluate and compare different datasets, a problem previously addressed by Raviv et al. (2021). Furthermore, I apply this to a real-life scenario of data trading. Consider the following scenario: assume a buyer wishes to solve a problem with machine learning, but does not have access to relevant good quality data nor a means to collect them. Then, the buyer must buy the data from an outside source, who may not be willing to share their data for free since a considerable amount of cost went into collecting them. How will the buyer

know which seller to buy data from and whether the buyer is paying a fair price considering other sellers and the marginal benefits the purchase will give to the buyer? Furthermore, how do sellers prove to the buyer that their data is better than others', without revealing their data, and thus compromising the very product they are trying to sell?

I propose using the *expected loss* of a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, as its measure of value, where $\mathbf{x}$ is the vector of features from the domain of the problem and y is the corresponding latent variable or label that we wish to predict. Expected loss is generally used in relation to models or hypotheses. Instead, I define the expected loss of a dataset as the expected loss corresponding to the posterior distribution $p(y \mid \mathbf{x}, \mathcal{D})$ induced by the dataset. The expectation is taken with respect to the domain distribution of $\mathbf{x}$ of the problem. The loss function is provided by the buyer, so that it suits the buyer's situation and allows for comparison with their marginal benefits. In this work, I focus on datasets of binary classification problems, where the label $y \in \{\pm 1\}$ or $\{0, 1\}$. However, I believe my work may be easily applied to other supervised machine learning problems, such as regression. Furthermore, I assume the data to be processed and do not consider "tidyness" of the data in measuring quality.

Addressing the problem of comparing sellers' datasets without compromising them, I present a secure multi-party computation (MPC) protocol using homomorphic encryption (HE) (Canetti et al., 1996; Rivest, 1978). The protocol involves three parties: the buyer, the seller, and an untrusted 3rd party (U3P). By keeping the encryption phase of the data within their owners and the computation of the expected loss within the U3P who has no access to decrypted data, my protocol keeps the data safe during evaluation. Here, I assume semi-honest parties, where an adversary may try to extract as much information as they can, beyond the intended amount, but follow the protocol honestly (Goldreich et al., 1987).

I begin this work in chapter 2 by formally defining the expected loss of a dataset and justifying its use as a measure of data quality. I use commonly used loss functions, but leave the definition open so that any reasonable loss function may be used. Then, in chapter 3, I explain implementation details in computing the expected loss, focusing on challenges that different data patterns present in terms of model structure uncertainty and parameter uncertainty. As a remedy, I turn to Bayesian model averaging (BMA) and Bayesian inference of model parameters and justify using them over other viable options. In chapter 4, I introduce HE,

the scheme I choose, and describe my full MPC protocol of dataset evaluation. I continue in chapter 5 by presenting experimental results on synthetic data and real-life data, using various loss functions to demonstrate the flexibility of using expected loss as a measure of data quality. Finally, I conclude this work by presenting the current limitations of this work in real-life feasibility. I also propose research questions that, once solved, would greatly improve both the performance and efficiency of my proposed protocol.

Chapter 2

Expected Loss

The idea of using expected loss for evaluating and comparing datasets is almost natural. Indeed it is how we often choose between different machine learning algorithms. For example, cross validation loss measured on an out-of-sample validation set can be seen as, albeit a crude one, a Monte Carlo approximation to the expected loss where we assume the problem and validation set to be generated by the same distribution. Using cross validation to choose a model and adjust hyperparameters is common practice in machine learning pipelines. We may simply extend this use for comparing different datasets and choose the dataset that most reduces the expected loss on the classification problem.

In this section, I first formally define the expected loss corresponding to a given dataset. Then, I look at how the expected loss of a dataset is proportional to the variance of valid hypotheses given the dataset. This part builds on previous work on data value estimation that uses the expected diameter of the space of possible linear classifiers, relating how the two are mathematically equivalent (Raviv et al., 2021). Third, I look at how minimizing the expected loss could be related to minimizing the conditional entropy of the label of points in the domain. Finally, while the idea of using expected loss to evaluate datasets is intuitively natural, it remains to see why and how a dataset with a low expected loss is valuable. Therefore, I justify the use of expected loss, presenting a scenario that this work tries to solve.

2.1 Definition

We first look at the expected loss of an individual hypothesis $h : \mathcal{X} \to \mathcal{Y}$ which we treat as a function whose input is a data point in the domain and the output is the predicted label or some relevant value. Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}, \ell(\hat{y}, y)$ be the loss of an individual prediction $\hat{y}$ when the true label is y. Throughout this work I use $p(\mathbf{x}, y)$ to mean the probability density function $p(X = \mathbf{x}, Y = y)$ where X, Y are random variables corresponding to the feature vector, or input, and the label, while $\mathbf{x}, y$ are their realizations. Furthermore, I use $\mathbb{P}(y) := \mathbb{P}(Y = y)$ to denote explicit probabilities, especially for Y, a discrete random variable. Then, we have:

$$L[h] \coloneqq \int_{\mathcal{X}\times\mathcal{Y}} \ell(h(\mathbf{x}), y) p(\mathbf{x}, y)\, d\mathbf{x}dy \tag{2.1.1}$$

Defining $\mathcal{Y} = \{0, 1\}$ for binary classification problems, we have the following:

$$\begin{aligned} L[h] &= \int_{\mathcal{X}} \big(\ell(h(\mathbf{x}), y = 1) p(\mathbf{x}, Y = 1) + \ell(h(\mathbf{x}), Y = 0) p(\mathbf{x}, Y = 0)\big) d\mathbf{x} \\ &= \int_{\mathcal{X}} \big(\ell(h(\mathbf{x}), y = 1)\mathbb{P}(y = 1 \mid \mathbf{x}) + \ell(h(\mathbf{x}), y = 0)\mathbb{P}(y = 0 \mid \mathbf{x})\big) p(\mathbf{x}) d\mathbf{x} \\ &= \mathbb{E}_{\mathbf{x}}\big[\ell(h(\mathbf{x}), y = 1)\mathbb{P}(y = 1 \mid \mathbf{x}) + \ell(h(\mathbf{x}), y = 0)\mathbb{P}(y = 0 \mid \mathbf{x})\big] \end{aligned} \tag{2.1.2}$$

Now it remains to see how we may relate $L[h]$ to a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$. Notice that upon a Bayesian treatment, a hypothesis may be seen as some function of the posterior distribution over the latent variable given an input and the dataset. Formally, for any problem, regardless of regression or classification, we may define:

$$h(\mathbf{x}) \coloneqq \phi(\mathbb{P}(y \mid \mathbf{x}, \mathcal{D})) \tag{2.1.3}$$

for some function ϕ that would depend on how we define a *prediction* and the corresponding loss ℓ. Then, observe how $h(\mathbf{x})$ is a function of $\mathcal{D}$. One final note is that we must account for how observing $\mathcal{D}$ changes the joint distribution $p(\mathbf{x}, y)$. Now, we may simply extend the definition in (2.2) to be a function of $\mathcal{D}$ as such:

Definition 2.1.1 (Expected Loss of a Dataset)**.** *The* ***expected loss*** *of a dataset $\mathcal{D}$ is the expected loss of the hypothesis induced by the posterior distribution of the latent variable given an input, that is in turn induced by the dataset. Denoted $L(\mathcal{D})$, we have:*

$$L(\mathcal{D}) \coloneqq \int_{\mathcal{X}\times\mathcal{Y}} \ell(\phi(\mathbb{P}(y \mid \mathbf{x}, \mathcal{D})), y) p(\mathbf{x}, y)\, d\mathbf{x}dy$$

where ϕ is a given function that maps the posterior distribution to a desired prediction and ℓ is a given loss function.

For binary classification problems, we have:

$$\begin{aligned} L(\mathcal{D}) = \mathbb{E}_{\mathbf{x}}\big[&\ell(\phi(\mathbb{P}(y \mid \mathbf{x}, \mathcal{D})), y = 1)\mathbb{P}(y = 1 \mid \mathbf{x}, \mathcal{D}) \\ &+ \ell(\phi(\mathbb{P}(y \mid \mathbf{x}, \mathcal{D})), y = 0)\mathbb{P}(y = 0 \mid \mathbf{x}, \mathcal{D})\big] \end{aligned} \tag{2.1.4}$$

For binary classification problems, we have two common configurations for ϕ and ℓ, and another configuration that is not commonly used but relevant to this work. Notice that for all loss functions, we define ϕ such that h returns the optimal prediction that minimizes ℓ. For simplicity, let $\pi(\mathbf{x}) = \mathbb{P}(y = 1 \mid \mathbf{x}, \mathcal{D})$. First is the case of using 0-1 loss upon explicit prediction of labels. In this case, for some threshold value $\rho \in [0, 1]$, we have:

$$\begin{aligned} h_0(\mathbf{x}) &= \phi_0(\pi(\mathbf{x})) \coloneqq \mathbb{1}[\pi(\mathbf{x}) \geq \rho] \\ \ell_0(\hat{y}, y) &= \mathbb{1}[\hat{y} \neq y] \end{aligned}$$

Second is the case of using log loss upon predicting the probability of each possible label. In this case, we have:

$$\begin{aligned} h_e(\mathbf{x}) &= \phi_e(\pi(\mathbf{x})) \coloneqq \pi(\mathbf{x}) \\ \ell_e(\hat{y}, y) &= -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \end{aligned}$$

Here, we look at how using the log loss translates to the expected loss of a dataset.

$$\begin{aligned} L(\mathcal{D}) &= \mathbb{E}_{\mathbf{x}}\big[\ell_e(\pi(\mathbf{x}),1)\mathbb{P}(y=1 \mid \mathbf{x},\mathcal{D}) + \ell_e(\pi(\mathbf{x}),y=0)\mathbb{P}(y=0 \mid \mathbf{x},\mathcal{D})\big] \\ &= -\mathbb{E}_{\mathbf{x}}\big[(\log \pi(\mathbf{x}))\pi(\mathbf{x}) + (\log(1-\pi(\mathbf{x})))(1-\pi(\mathbf{x}))\big] \\ &= -\mathbb{E}_{\mathbf{x}}\big[\pi(\mathbf{x})\log \pi(\mathbf{x}) + (1-\pi(\mathbf{x}))\log(1-\pi(\mathbf{x}))\big] \end{aligned} \tag{2.1.5}$$

The third rare case is using ℓ_2 loss upon predicting the probability of each possible label. In this case, we have:

$$\begin{aligned} h_2(\mathbf{x}) &= \phi_2(\pi(\mathbf{x})) \coloneqq \pi(\mathbf{x}) \\ \ell_2(\hat{y}, y) &= (y-\hat{y})^2 \end{aligned}$$

Notice again that our prediction $h_2(\mathbf{x})$ is the optimal prediction that minimizes ℓ_2. Here, we look at how using the ℓ_2 loss translates to the expected loss of a dataset.

$$\begin{aligned} L(\mathcal{D}) &= \mathbb{E}_{\mathbf{x}}\big[\ell_2(\pi(\mathbf{x}),1)\mathbb{P}(y=1 \mid \mathbf{x},\mathcal{D}) + \ell_2(\pi(\mathbf{x}),y=0)\mathbb{P}(y=0 \mid \mathbf{x},\mathcal{D})\big] \\ &= \mathbb{E}_{\mathbf{x}}\big[(1-\pi(\mathbf{x}))^2\pi(\mathbf{x}) + (0-\pi(\mathbf{x}))^2(1-\pi(\mathbf{x}))\big] \\ &= \mathbb{E}_{\mathbf{x}}\big[(1-\pi(\mathbf{x}))\pi(\mathbf{x})\big((1-\pi(\mathbf{x})) + \pi(\mathbf{x})\big)\big] \\ &= \mathbb{E}_{\mathbf{x}}\big[(1-\pi(\mathbf{x}))\pi(\mathbf{x})\big] \end{aligned} \tag{2.1.6}$$

Notice here that assuming some random variable A to have a Bernoulli distribution with parameter $\mathbb{P}(A=1) = \pi(\mathbf{x})$ gives us that $L(\mathcal{D})$ is the expected variance of A, with expectation taken with respect to $\mathbf{x}$:

$$L(\mathcal{D}) = \mathbb{E}_{\mathbf{x}}\mathrm{Var}(A) \tag{2.1.7}$$

This will be used later when showing how the idea of the expected diameter in Raviv et al. (2021) is equivalent to the variance of hypotheses induced by the dataset, and consequently, the expected loss of the dataset.

While the above are a few of the most widely used loss functions, note that there is no limit to how ℓ may be configured. This allows for any loss function for any situation to be applied in computing the expected loss.

2.2 Variance of Hypotheses

In a similar work, Raviv et al. (2021) measure the value of a dataset according to the *expected diameter* defined as such:

Definition 2.2.1 (Expected Diameter)**.** *For a given dataset $\mathcal{D}$ and a given probability distribution $\mathcal{H}$ over its set of consistent hypotheses, the* ***expected diameter*** *of $\mathcal{D}$ is $\mathbb{E}_{h_1,h_2\sim\mathcal{H}}d(h_1,h_2)$, where $d(h_1,h_2)$ is defined as the fraction of $\mathbf{x}$'s on which h_1 and h_2 disagree.*

Here, set $\mathcal{H}$ is defined as the set of valid hypotheses under $\mathcal{D}$, equipped with a probability distribution over the hypotheses. In other words, if we think of the domain as a linearly separable space according to the associated labels, $\mathcal{H}$ is the set (and distribution) of hyperplanes that separate the domain in accordance with the points in $\mathcal{D}$. Here, each hypothesis $h \in \mathcal{H}$ simply predicts the label associated with a point $\mathbf{x}$. Raviv et al. (2021) claim that a dataset with a lower expected diameter is more valuable as it means that the dataset eliminates more possible hyperplanes. In other words, a dataset with a lower expected diameter carries more information about where the true label-generating hyperplane lies. Taking a Bayesian approach, we may claim that $\mathcal{H}$ is the posterior distribution over the set of possible hyperplanes induced by $\mathcal{D}$.

While Raviv et al. (2021) focuses on binary domains where $\mathbf{x} \in \{\pm 1\}^n$, we may easily extend it for continuous domains. Let $D(\mathcal{D})$ denote the expected diameter of a dataset $\mathcal{D}$ and $H(\mathbf{x}) = \mathbb{E}_h h(\mathbf{x})$. Switching to $\mathcal{Y} = \{\pm 1\}$ labels for simplicity, we observe the following from Raviv et al. (2021):

$$\begin{aligned} D(\mathcal{D}) &= \mathbb{E}_{h_1,h_2} d(h_1,h_2) \\ &= \mathbb{E}_{h_1,h_2}\mathbb{E}_{\mathbf{x}}\left[\frac{1-h_1(\mathbf{x})h_2(\mathbf{x})}{2}\right] \\ &= \mathbb{E}_{\mathbf{x}}\mathbb{E}_{h}\left[\frac{1-h(\mathbf{x})H(\mathbf{x})}{2}\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\frac{1-H(\mathbf{x})^2}{2}\right] \end{aligned} \tag{2.2.1}$$

Furthermore, $\forall \mathbf{x} \in \mathcal{X}$, if we look at the variance of the hypothesis $h \sim \mathcal{H}$, we have:

$$\begin{aligned}
\mathrm{Var}_h(h(\mathbf{x})) &= \mathbb{E}_h[(h(\mathbf{x}) - H(\mathbf{x}))^2] \\
&= \mathbb{E}_h[h(\mathbf{x})^2 - 2h(\mathbf{x})H(\mathbf{x}) + H(\mathbf{x})^2] \\
&= 1 - 2H(\mathbf{x})^2 + H(\mathbf{x})^2 \\
&= 1 - H(\mathbf{x})^2
\end{aligned} \tag{2.2.2}$$

Therefore, using (2.2.1) and (2.2.2), we have:

$$D(\mathcal{D}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}}\mathrm{Var}_h(h(\mathbf{x})) \tag{2.2.3}$$

In other words, the expected diameter of a dataset is proportional to the expected variance of the hypothesis, which is also the expected ℓ_2 loss between a hypothesis and the expected hypothesis.

Now, we show the following:

Theorem 2.2.1. *The expected ℓ_2 loss of a dataset is proportional to the expected diameter of a dataset.*

Proof. Notice that since $h \sim \mathcal{H}$, as used by Raviv et al. (2021), where $\mathcal{H}$ is the posterior distribution over the set of valid hypotheses induced by $\mathcal{D}$, for a given point $\mathbf{x}$, we have:

$$\mathbb{P}(h(\mathbf{x}) = 1) = \mathbb{P}(y = 1 \mid \mathbf{x}, \mathcal{D}) =: \pi(\mathbf{x})$$

Then, since $h(\mathbf{x}) \in \{\pm 1\}$ we have that $h(\mathbf{x})$ is a Bernoulli random variable with parameter $\pi(\mathbf{x})$. Therefore, by (2.1.7) and (2.2.3):

$$\begin{aligned}
L(\mathcal{D}) &= \mathbb{E}_{\mathbf{x}}\mathrm{Var}_h(h(\mathbf{x})) \\
&\propto D(\mathcal{D})
\end{aligned} \tag{2.2.4}$$

□

The above theorem shows that the expected ℓ_2 loss is proportional to the expected diameter of the dataset, where both are equivalent to the expected variance of hypotheses induced by the dataset. Therefore, we may conclude that a dataset with lower expected diameter and lower expected ℓ_2 loss will induce a posterior distribution over possible hypotheses with lower variance. With lower variance, we may conclude that such a dataset will have greater confidence over where the true label-generating hyperplane lies.

2.3 Expected Loss and Information Gain

The information gain is defined as the Kullback-Leibler divergence of the prior distribution from the posterior distribution. Using this definition for our problem, for a given point $\mathbf{x}$, we have:

$$\begin{aligned} IG(y \mid \mathbf{x}, \mathcal{D}) &= D_{\mathrm{KL}}(\mathbb{P}(y \mid \mathbf{x}, \mathcal{D}) \mid\mid \mathbb{P}(y \mid \mathbf{x})) \\ &= H(y \mid \mathbf{x}) - H(y \mid \mathbf{x}, \mathcal{D}) \end{aligned} \tag{2.3.1}$$

where H is the information entropy of a random variable. Since the entropy of a random variable measures how much *randomness* or *surprise* there is in the random variable's outcome, it makes sense that a valuable dataset that is informative should significantly reduce the entropy. Therefore, we may claim that a dataset that leads to more expected information gain over $\mathbf{x} \sim \mathcal{X}$ for the random variable $y \mid \mathbf{x}$ is also more valuable.

Now, we show the following:

Theorem 2.3.1. *Given datasets D_i, $i = 1, \cdots, n$, the dataset that minimizes the expected* log *loss is the dataset that maximizes the expected information gain of $y \mid \mathbf{x}$ given $\mathcal{D}$, with expectation taken over $\mathbf{x}$.*

Proof. Notice that each $\mathcal{D}_i$ does not affect $\mathbb{P}(y \mid \mathbf{x})$. Then, let $C := \mathbb{E}_{\mathbf{x}} H(y \mid \mathbf{x})$ a constant that depends on the prior distribution of $y \mid \mathbf{x}$ and $\mathbf{x}$. Also let $\pi_i(\mathbf{x}) = \mathbb{P}(y = 1 \mid \mathbf{x}, \mathcal{D}_i)$.

Then, by (2.1.5) and (2.3.1) we have:

$$\begin{aligned}
\arg\min_i \ L(\mathcal{D}_i) &= \arg\min_i \ -\mathbb{E}_{\mathbf{x}}\big[\pi_i(\mathbf{x})\log\pi_i(\mathbf{x}) + (1-\pi_i(\mathbf{x}))\log(1-\pi_i(\mathbf{x}))\big] \\
&= \arg\max_i \ \mathbb{E}_{\mathbf{x}}\big[\pi_i(\mathbf{x})\log\pi_i(\mathbf{x}) + (1-\pi_i(\mathbf{x}))\log(1-\pi_i(\mathbf{x}))\big] \\
&= \arg\max_i \ C + \mathbb{E}_{\mathbf{x}}\big[\pi_i(\mathbf{x})\log\pi_i(\mathbf{x}) + (1-\pi_i(\mathbf{x}))\log(1-\pi_i(\mathbf{x}))\big] \\
&= \arg\max_i \ \mathbb{E}_{\mathbf{x}} H(y \mid \mathbf{x}) - \mathbb{E}_{\mathbf{x}} H(y \mid \mathbf{x}, \mathcal{D}_i) \\
&= \arg\max_i \ \mathbb{E}_{\mathbf{x}} IG(y \mid \mathbf{x}, \mathcal{D}_i) \qquad (2.3.2)
\end{aligned}$$

□

2.4 Expected Loss as a Measure of Value

Theorem 2.1 and 2.2 both justify using expected loss to evaluate the value of a dataset. Theorem 2.1 shows how comparing datasets according to the expected diameter idea used in Raviv et al. (2021) is equivalent to using the expected ℓ_2 loss. Theorem 2.2 shows how comparing datasets according to their respective expected information gain of the latent variables associated with points in the domain is equivalent to using the expected log loss. Therefore, besides the natural argument of stating that minimizing expected loss is desirable, we have that a dataset with lower expected loss is valuable because of how it has lower expected diameter and higher information gain.

Another advantage of using expected loss is its applicability to different situations. First. consider the case when the buyer has their own data and a corresponding hypothesis and are looking to augment it by buying more data. In this case, we may slightly alter the problem as such. Instead of simply choosing the dataset with lowest expected loss, we choose the dataset with lowest expected loss once augmented onto the buyer's existing data:

$$\mathcal{D}^* = \arg\max_i \ L(\mathcal{D}_b) - L(\mathcal{D}_b \cup \mathcal{D}_i) = \arg\min_i \ L(\mathcal{D}_b \cup \mathcal{D}_i) \qquad (2.4.1)$$

where $\mathcal{D}_b$ is the buyer's existing dataset. An additional advantage of expected loss in this scenario is that a buyer may discover that all options do not significantly decrease expected

loss, and therefore they choose not to invest in augmenting their data. This allows for a reasoned purchase decision in buying data.

Another case where using expected loss is advantageous is when the data buyer has a specific loss function that meets their needs, as previously mentioned in section 2.1. For example, consider a medical diagnostic of a viral disease, where the loss of making a type II error (false negative) is much higher than that of making a type I error (false positive). In this case, a buyer may configure a loss function that heavily penalizes type II errors to evaluate datasets. Furthermore, this allows for a buyer to configure a loss function so that they can immediately compare the expected monetary gain from buying the dataset and make purchase decisions accordingly. These two cases will be further investigated with implemented examples in Chapter 5.

Finally, we look at another significance of expected loss and what a dataset with low expected loss actually implies. To do so, let us compare what the *ideal* dataset looks like and what a useless dataset looks like. Note that for both the ℓ_2 and log loss functions, simply taking the derivative shows that the expected loss decreases as each $\pi(\mathbf{x})$ gets closer to 0 or 1. On the other hand, expected loss increases as each $\pi(\mathbf{x})$ gets closer to 0.5. Then, we see that the expected loss in this case relies on how confident we are in our predictions. This, however, presents the problem that an adversarial dataset whose corresponding posterior distribution leads to all points being predicted to have label $y = +1$ with probability 1 may be judged to be *ideal*, which is why we assume that we work under semi-honest adversaries that follow the protocol honestly. In real-life cases, we could easily filter out malicious adversaries who do not follow protocol with a small amount of validation data.

Chapter 3

Different Data Patterns

The justification for using expected loss as a measure of the value of a dataset depends on the assumption that the posterior distribution $p(y \mid \mathbf{x}, \mathcal{D})$ is reliable. This depends on the choice of the model prior and model evidence, more so on the latter. Our choice of model evidence is determined by how we model the relationship between $\mathbf{x}$ and its corresponding latent variable y. This decision heavily influences the outlook we have on candidate datasets, where modeling the relationship in the wrong way could lead to a comparatively uninformative dataset being valued more than an informative one. In this section, I demonstrate such a phenomenon and discuss non-parametric modeling and Bayesian model averaging (BMA) as remedies to this problem. This addresses the problem of model uncertainty. Furthermore, I discuss maximum likelihood estimation (MLE) and maximum a posteriori (MAP) methods in parametric modeling of the model evidence and how such point estimation methods could fail to differentiate datasets of varying value, unlike Bayesian inference. This addresses the problem of parameter uncertainty. Recall in the previous chapter that the significance of expected loss lies in how a dataset reduces uncertainty about the true underlying model. Therefore, uncertainty quantification for each dataset naturally becomes a necessary part in computing the expected loss, which is ultimately what this chapter tries to solve.

3.1 BMA and Non-parametric Modeling

Consider the following toy example of comparing two datasets that were generated by the same 2-dimensional binary classification problem in Figure 3.1, where dataset $\mathcal{D}_1$ only contains sample from a specific area of the domain while dataset $\mathcal{D}_2$ captures a wider area.

Samples in one class are marked with a blue + while samples in the other are marked with a red −. Using a logistic regression model where $y = \sigma(\theta_0 + \theta_1 \cdot [\mathbf{x}]_1 + \theta_2 \cdot [\mathbf{x}]_2)$ for σ the sigmoid function, and using 0.5 as the threshold for classification, Figure 3.1 shows the decision boundaries corresponding to the mean and 50 samples from a Laplace approximation to the posterior distribution of the weights, with the intercept $-(\theta_0/\theta_2)$ set to the mean for sake of visualization.

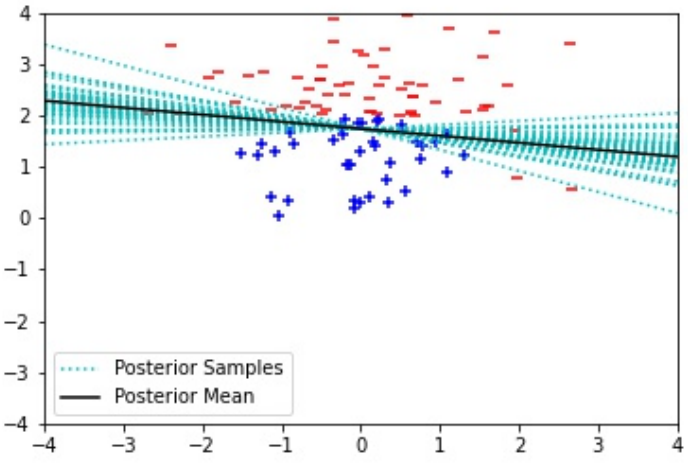
Due to the data disagreeing with the structure of a linear model, the posterior distribution induced by $\mathcal{D}_2$ has high variance, and therefore is associated with high uncertainty about the true decision boundary. However, the posterior distribution induced by $\mathcal{D}_1$ is relatively confident about where the true decision boundary lies. Indeed, when using the respective posterior distributions for computing the expected loss with the log loss function, assuming a uniform distribution over the domain $\mathcal{X} = [-4, 4] \times [-4, 4]$, we have $L(\mathcal{D}_1) = 0.1442$ while $L(\mathcal{D}_2) = 0.6588$. On the other hand, as expected, the actual expected log loss with known labels over the domain for $\mathcal{D}_1$ is very high at $\mathcal{L}(\mathcal{D}_1) = 5.2133$ while $\mathcal{L}(\mathcal{D}_2) = 0.6209$. Here, the actual expected log loss with known labels was taken as usual:

$$\mathcal{L}(\mathcal{D}) = -\mathbb{E}_{\mathcal{X}}[y \log \pi(\mathbf{x}) + (1 - y) \log(1 - \pi(\mathbf{x}))] \tag{3.1.1}$$

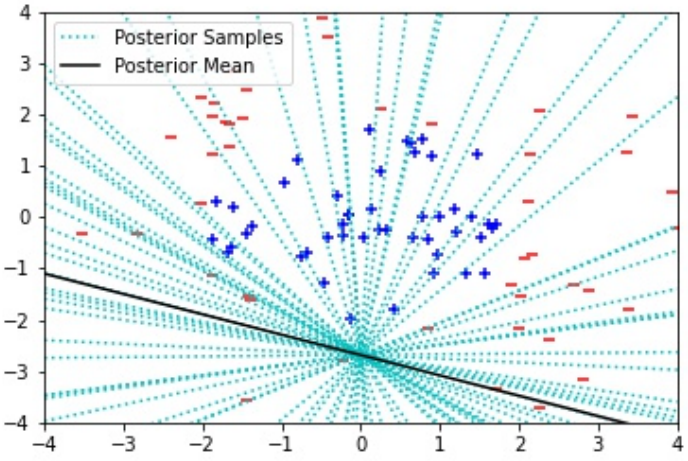
As it can be seen from the above toy example, using the right model in computing the posterior distribution is imperative for using the expected loss in evaluating datasets. However, knowing a priori which model to use in computing the expected loss is not always feasible. To overcome this, two methods are proposed: Bayesian model averaging (BMA) and non-parametric models.

3.1.1 Bayesian Model Averaging

BMA is a soft model selection method that factors in uncertainty about the problem's structure. Given a set of models $M = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ with a prior distribution over each model, it computes the posterior distribution for each model given the data, and uses it to output a weighted average for each prediction. The posterior distribution $\mathbb{P}(\mathcal{M}_i \mid \mathcal{D})$ is



(a) $\mathcal{D}_1$



(b) $\mathcal{D}_2$

Figure 3.1: Two datasets with posterior distribution of slope of decision boundary assuming a linear model

computed as:

$$\mathbb{P}(\mathcal{M}_i \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \mathcal{M}_i)\mathbb{P}(\mathcal{M}_i)}{p(\mathcal{D})} \tag{3.1.2}$$

where the likelihood of the data given a model is computed by marginalizing the relevant parameters $\boldsymbol{\theta}_i$ as:

$$p(\mathcal{D} \mid \mathcal{M}_i) = \int_{\Theta} p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}_i, \mathcal{M}_i) p(\boldsymbol{\theta}_i \mid \mathcal{M}_i) d\boldsymbol{\theta}_i \tag{3.1.3}$$

and the denominator, or normalizing factor is simply computed as:

$$p(\mathcal{D}) = \sum_{i=1}^{n} p(\mathcal{D} \mid \mathcal{M}_i)\mathbb{P}(\mathcal{M}_i) \tag{3.1.4}$$

With the posterior computed, the predictive posterior distribution becomes:

$$p(y \mid \mathbf{x}, \mathcal{D}) = \sum_{i=1}^{n} p(y \mid \mathbf{x}, \mathcal{D}, \mathcal{M}_i)\mathbb{P}(\mathcal{M}_i \mid \mathcal{D}) \tag{3.1.5}$$

However, notice that the posterior distribution for each model also depends on the data. In other words, going back to the toy example, if the model posterior distributions were computed separately, then $\mathcal{D}_1$ would output a linear model to be close to the true model, and BMA would fail in discovering that a linear model is not to be trusted. Therefore, with various sellers, it is necessary that the model posterior probabilities are computed with all the data combined. In other words, if there are datasets $\mathcal{D}_1, \ldots, \mathcal{D}_m$, model posterior distributions should be computed as:

$$\mathbb{P}(\mathcal{M}_i \mid \mathcal{D}_1, \ldots, \mathcal{D}_m) \tag{3.1.6}$$

for each possible model. Using BMA in computing the expected loss is demonstrated in Chapter 5 with synthetic data and encryption.

One potential problem for this method is the need for a set of predetermined models and a prior distribution over them. That said, I believe the need for a prior distribution could even be an advantage that reflects the buyer's prior knowledge about the problem, especially that reflects the buyer's specific needs. Furthermore, the problem of model selection is not particular just to this scenario, but for the entire field of machine learning and statistics.

Finally, I add that when computing the model posterior probabilities in real-life, computing the integral in (3.1.3) is often intractable, especially for classification problems using a logistic link function. Therefore, in practice, I use the Bayesian Information Criterion (BIC) as a surrogate to the model posterior probabilities. The BIC is defined as such:

$$\text{BIC}(\mathcal{M}_i) = \log p(\mathcal{D} \mid \hat{\theta}_{\text{MAP}}) - \frac{d}{2} \log N \tag{3.1.7}$$

where d denotes the dimensionality of θ and N denotes the number of samples in $\mathcal{D}$. Notice that the BIC penalizes complex models with many parameters, appealing to Occam's Razor.

3.1.2 Non-parametric Models

Another solution to the problem of model selection in computing expected loss is to use non-parametric models that do not assume a predetermined structure in the pattern of the data. Gaussian process (GP) models are a natural choice as a non-parametric probabilistic model due to its theoretical guarantees (Neal, 1996) and great interpretability (Rasmussen and Williams, 2005). Furthermore, GPs exhibit uncertainty in areas of the domain far from where the data lies. Therefore, using GPs for computing the expected loss favors data that are relevant to the buyer's problem. Finally, GPs naturally work as a Bayesian model that automatically factors in uncertainty about predictions. This proves to be important in computing the expected loss as it is shown in the next section.

We may see how using GPs perform in the above problem. Using a GP classification model with the RBF kernel, the logistic link function, and a Laplace approximation to compute the expected log losses L and the actual log losses $\mathcal{L}$, we have:

$$\begin{aligned} L(\mathcal{D}_1) &= 0.6296 \quad \mathcal{L}(\mathcal{D}_1) = 0.5356 \\ L(\mathcal{D}_2) &= 0.5063 \quad \mathcal{L}(\mathcal{D}_2) = 0.4048 \end{aligned}$$

Indeed, we see that the expected loss is a closer approximation to the actual log loss and preserves the order of quality between $\mathcal{D}_1$ and $\mathcal{D}_2$. Note that this toy experiment was done without careful hyperparameter tuning to emulate real-life scenarios.

3.2 Point Estimation and Bayesian Inference

As seen in the previous section, considering model uncertainty is crucial in computing the expected loss between datasets that may fit each assumed model differently. Now, we look at a case where two datasets may follow the same model structure, but with different levels of confidence. This addresses parameter uncertainty in computing the expected loss with parametric models. Before looking at a demonstration, we may look at how point estimation and Bayesian inference differ in computing the posterior predictive distributions. An MAP point estimation simply uses the following:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{3.2.1}$$

$$p(y \mid \mathbf{x}, \mathcal{D}) = p(y \mid \mathbf{x}, \boldsymbol{\theta}^*) \tag{3.2.2}$$

whereas a Bayesian treatment uses:

$$p(y \mid \mathbf{x}, \mathcal{D}) = \int_{\Theta} p(y \mid \mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) d\boldsymbol{\theta} \tag{3.2.3}$$

Notice that the point estimate does not consider the variance of the parameters' posterior distribution, and therefore the amount of uncertainty that the data induces is ignored. On the other hand, Bayesian inference, albeit with the use of an often intractable integral, marginalizes the parameters, fully incorporating its uncertainty.

Consider the following toy example of two datasets of different quality in Figure 3.2 (plotted similarly to Figure 3.1), where $\mathcal{D}_1$ has closely packed clusters of points farther away from the true latent decision boundary while $\mathcal{D}_2$ has wide clusters of points closer to the latent boundary. Intuitively, $\mathcal{D}_2$ is more informative than $\mathcal{D}_1$ about where the latent boundary lies. Indeed, the samples from the Laplace approximation of the Bayesian posterior shows a lot of uncertainty in where the boundary lies for $\mathcal{D}_1$ compared to that for $\mathcal{D}_2$. However, using a point estimate for the model to compute $p(y \mid \mathbf{x}, \mathcal{D})$ would lead to both datasets using almost the same parameters, and consequently, lead to similar expected loss. Again,

computing the expected log loss and actual log loss with a point estimate, we have:

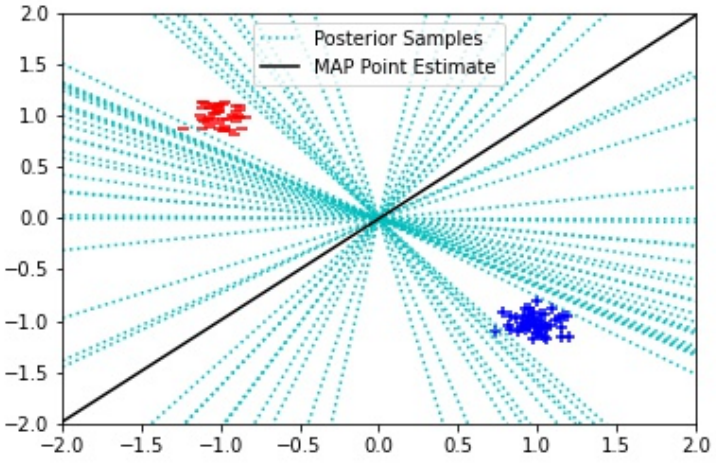
$$L_{\text{MAP}}(\mathcal{D}_1) = 0.2331 \quad \mathcal{L}_{\text{MAP}}(\mathcal{D}_1) = 0.1814$$
$$L_{\text{MAP}}(\mathcal{D}_2) = 0.2437 \quad \mathcal{L}_{\text{MAP}}(\mathcal{D}_2) = 0.1440$$

Notice that the expected loss is almost equal to each other where as the actual loss favors $\mathcal{D}_2$. Using a Laplace approximation to the Bayesian posterior to incorporate parameter uncertainty, we have:

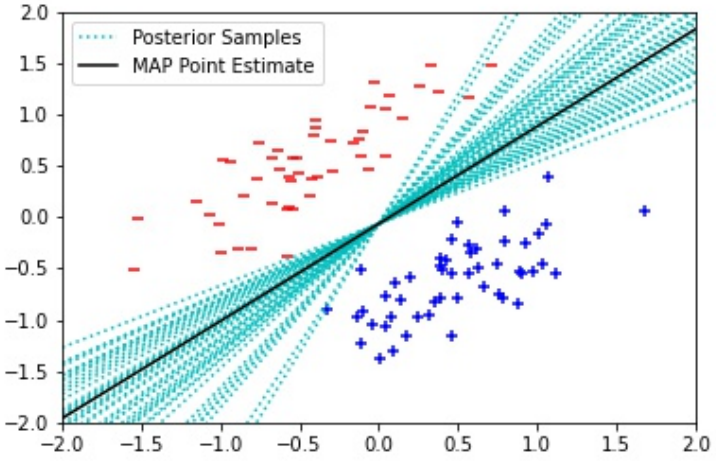
$$L(\mathcal{D}_1) = 0.4453 \quad \mathcal{L}(\mathcal{D}_1) = 0.3963$$
$$L(\mathcal{D}_2) = 0.1751 \quad \mathcal{L}(\mathcal{D}_2) = 0.1525$$

which successfully incorporates parameter uncertainty in measuring and differentiating the quality of the two datasets.

This phenomenon of parameter uncertainty affecting dataset quality will be much more significant in real-life higher dimensional data due to the curse of dimensionality. Therefore, despite the computational challenges in incorporating Bayesian inference, especially over an encrypted domain which will be discussed in the next chapter, using Bayesian inference for parameter uncertainty quantification is necessary in computing expected loss.



(a) $\mathcal{D}_1$



(b) $\mathcal{D}_2$

Figure 3.2: Two datasets with posterior distribution of slope of decision boundary assuming a linear model

Chapter 4

Homomorphic Encryption

As we assume to work under a semi-honest protocol, we are presented the challenge of computing the expected loss without compromising the data. With an untrusted 3rd party (U3P) computing the expected loss on behalf of the buyer and the seller, I propose using homomorphic encryption (HE) for encrypting both the buyer and seller's data and computing the expected loss under encryption. In this chapter, I will first briefly explain HE and the implementation I chose to use along with how logistic regression is performed. Then, despite the argument made in section 3.4, I show why Bayesian inference is infeasible with HE, and propose bootstrapping for approximate Bayesian inference. Finally, putting everything together, I outline the full pipeline of dataset evaluation that may be applied to a real-life data market.

4.1 HE for Arithmetic of Approximate Numbers (HEAAN)

HE is a cryptographic scheme that allows homomorphic operations on encrypted data to be preserved after decryption (Rivest, 1978). This allows for untrusted parties to be able to process the encrypted data without the decryption key. In relation to our problem, this allows the U3P to compute the expected loss on encrypted data without ever accessing the decrypted data itself. Using the notation used by Gentry (2009) in the introduction of Fully HE (FHE), a public key HE scheme $\mathcal{E}$ has four algorithms: $\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$, and $\mathsf{Evaluate}$. KeyGen is the usual algorithm that generates a public key pk and private key sk where the public key is used by $\mathsf{Encrypt}$ for encrypting data (or plaintext) into encrypted data (or ciphertext) and the private key is used by $\mathsf{Decrypt}$ for decrypting ciphertext into plaintext.

Evaluate is the algorithm of interest, where once provided a public key (or evaluation key evk as we will later see), a circuit of operations, and a tuple of ciphertexts, performs the circuit on the ciphertexts to output an encrypted version of the output whose corresponding decrypted plaintext should equal to the output if the circuit were performed on the original plaintexts themselves. The following shows a brief pseudocode illustrating the process:

Algorithm 4.1 Homomorphic Public Key Encryption Scheme

Input: plaintext tuple $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$, circuit C, secruity parameter λ
Output: output $\pi = C(\boldsymbol{\pi})$
(sk, pk) $\leftarrow \mathsf{KeyGen}(\lambda)$
$(\psi_1, \ldots, \psi_n) =: \boldsymbol{\Psi} \leftarrow \mathsf{Encrypt}(\mathrm{pk}, \boldsymbol{\pi})$
$\psi \leftarrow \mathsf{Evaluate}(\mathrm{pk}, C, \boldsymbol{\Psi})$
$\pi \leftarrow \mathsf{Decrypt}(\mathrm{sk}, \psi)$
return π

Note that a scheme $\mathcal{E}$ is a HE scheme if the ciphertext size and decryption time is polynomial in λ. A FHE, as introduced by Gentry (2009), is a scheme that is homomorphic for all circuits. While many FHE schemes exist, I chose to use the HEAAN scheme by Cheon et al. (2016) that performs efficient approximate computation over complex numbers, allowing for dealing with real-valued continuous data. In this section, I will briefly introduce the ring learning with errors (RLWE) problem, on which the HEAAN scheme, along with many other FHE schemes, is based on, and then the HEAAN scheme itself.

4.1.1 RLWE Problem

The RLWE problem, introduced by Lyubashevsky et al. (2013), extends the learning with errors (LWE) problem, introduced by Regev (2005). While Regev first introduced the search version of the problem, we present the decision version which is more relevant to this work. The decision LWE problem is as follows:

Definition 4.1.1 (Learning with Errors Decision Problem)**.** *Let secret* $\mathbf{s} \in \mathbb{Z}_q^n$ *with size* $n \geq 1$*, modulus* $q \geq 2$*. Let* χ *on* $\mathbb{Z}_q$ *be the probability distribution of the error* e*. Then, let* $A_{\mathbf{s},\chi}$ *on* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *be the probability distribution of* $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e \mod q)$ *from sampling*

$\mathbf{a} \sim$ *Uniform*$(\mathbb{Z}_q^n)$ *and* $e \sim \chi$. *The LWE decision problem is to distinguish between samples from* $A_{\mathbf{s},\chi}$ *and uniformly random samples from* $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Notice that this problem would be easy to solve if it were not for the $e \sim \chi$ term that introduces the "error" to the linear equations.

The RLWE problem substitutes $\mathbb{Z}_q^n$ with the ring $\mathcal{R}_q := \mathbb{Z}_q[x]/\Phi_n(x)$, where $\Phi_n(x)$ is the n^{th} cyclotomic polynomial, switching from working with vectors to polynomials. Note that for $n = 2^h$, we simply have $\Phi_n(x) = x^{n/2} + 1$. Furthermore, notice that the transition from vectors to polynomials is natural once we consider representing each polynomial in terms of its coefficients. The RLWE problem as used in Cheon et al. (2016) in their description of the HEAAN scheme follows this definition, using a discretized rounded Gaussian distribution for χ. Formally written, we have the following definition:

Definition 4.1.2 (Ring Learning with Errors Decision Problem)**.** *Let secret* $s \in \mathcal{R}_q$ *and* χ *be the error distribution. Then, let* $A_{s,\chi}$ *on* $\mathcal{R}_q \times \mathcal{R}_q$ *be the probability distribution of* $(a, \langle a, s\rangle + e)$ *from sampling* $a \sim$ *Uniform*$(\mathcal{R}_q)$ *and* $e \sim \chi$. *The RLWE decision problem is to distinguish between samples from* $A_{s,\chi}$ *and uniformly random samples from* $\mathcal{R}_q \times \mathcal{R}_q$.

The hardness of this problem is discussed in detail by Lyubashevsky et al. (2013), where they show the equivalence of the decision problem with the search problem, and prove a quantum reduction from the worst-case shortest vector problem on ideal lattices in $\mathcal{R} := \mathbb{Z}[x]/\Phi_n(x)$ to the search problem. Regev (2010) also discusses the advantages that RLWE has over LWE, especially in cryptography, with regards to size and speed.

4.1.2 HEAAN Scheme

The philosophy behind HEAAN is that the encryption noise should be ignored as an error occuring in approximate computation, similarly to how observation noise, which occurs when data are often not a perfect representation of the true values, and finite precision arithmetic in computers, which is a result of representing real numbers with a finite number of bits, are treated. This is where the connection to the RLWE problem lies: the $e \sim \chi$ in RLWE problems that ensures its hardness is treated as an encryption noise and kept small such that it does not significantly perturb the original data.

In fact, HEAAN's novelty compared to other FHE schemes is motivated by the idea of how numbers are stored with a finite number of bits. The challenge in past FHE schemes was identifying the least significant bits (LSBs) so that they may be removed in a rounding process, especially in computing multiplication. This challenge was caused by the fact that the "rounding" of a high degree polynomial cannot be easily represented as a lower degree polynomial. To bypass this problem, HEAAN uses a rescaling technique that manipulates the ciphertexts by rescaling them as such:

$$\mathbf{c}' \leftarrow \lfloor p^{-1} \cdot \mathbf{c} \rceil \mod q/p \tag{4.1.1}$$

where $\langle \mathbf{c}, sk \rangle = m + e \mod q$ for $\mathbf{c}$ the ciphertext, sk the private key, m the plaintext, e the encryption noise, and p a rescaling factor, such that $\mathbf{c}'$ becomes an encryption of m/p with noise e/p both $\mod q/p$. Doing so for $\mathbf{c}_1, \mathbf{c}_2$ when multiplying the two reduces the size of the ciphertext modulus $q \to q/p$, mimicking fixed floating-point arithmetic for reasonable precision and outstanding efficiency. The technical implementation of how rescaling is done with an additional "evaluation key" is described in depth in Cheon et al. (2016).

This allows HEAAN to efficiently implement the operations $\mathsf{Add}(\mathbf{c}_1, \mathbf{c}_2)$ and $\mathsf{Mulitply}(\mathbf{c}_1, \mathbf{c}_2)$ for evaluating circuits in ciphertext while maintaining reasonable accuracy, even when working with continuous real numbers. Then, this allows us to perform any polynomial operations on the data. Again, Cheon et al. (2016) go in depth on how HEAAN implements polynomials, along with approximate polynomials, multiplicative inverses, and fast Fourier transforms. For the purposes of this work, I will focus on polynomials.

4.2 Logistic Regression with HE

Unfortunately, the fact that HEAAN only allows for polynomials is limiting. When computing expected loss, this becomes a problem once we are compelled to compute $p(y \mid \mathbf{x}, \mathcal{D})$. Logistic regression, for example, requires evaluation of a sigmoid function σ which is not a polynomial of finite degree:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad \sigma'(z) = \sigma(z)(1 - \sigma(z)) \tag{4.2.1}$$

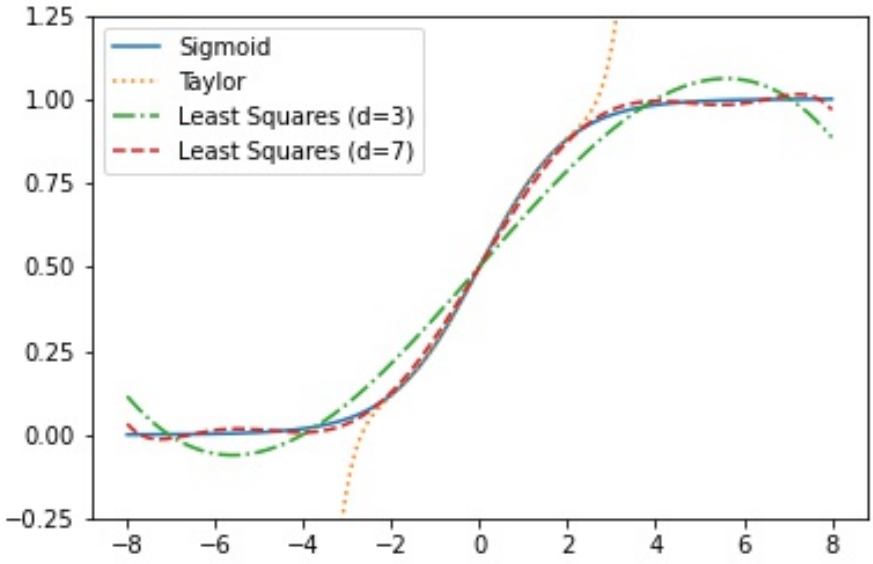


Figure 4.1: Approximations of the Sigmoid Function

On the other hand, apart from the sigmoid function and its derivative, using logistic regression and samples of the domain distribution $\mathbf{x} \sim \mathcal{X}$ to compute the expected loss with a polynomial loss function may be done all with ciphertext until we decrypt the final output to get a plaintext version of the expected loss (the challenge of using Bayesian logistic regression discussed in section 3.4 will be addressed in section 4.3).

While Cheon et al. (2016) mention using a Taylor approximation to the sigmoid function, later works including Kim et al. (2018) and Han et al. (2019) propose using a least squares approximation over the interval $z \in [-8, 8]$. Indeed, a Taylor approximation of degree 9 has a rapidly increasing error from $|z| > 2$, while a least squares approximation of just degree 3 maintains reasonable errors as shown in Figure 4.1 and Figure 4.2. The polynomials are numerically:

$$
\begin{aligned}
T_9(z) &= \frac{1}{2} + \frac{1}{4}z - \frac{1}{48}z^3 + \frac{1}{480}z^5 - \frac{17}{80640}z^7 + \frac{31}{1451520}z^9 \\
g_3(z) &\approx 0.5 + 0.1501z - 0.001593z^3 \\
g_7(z) &\approx 0.5 + 0.2169z - 0.008191z^3 + 0.0001658z^5 - 0.000001196z^7
\end{aligned}
$$

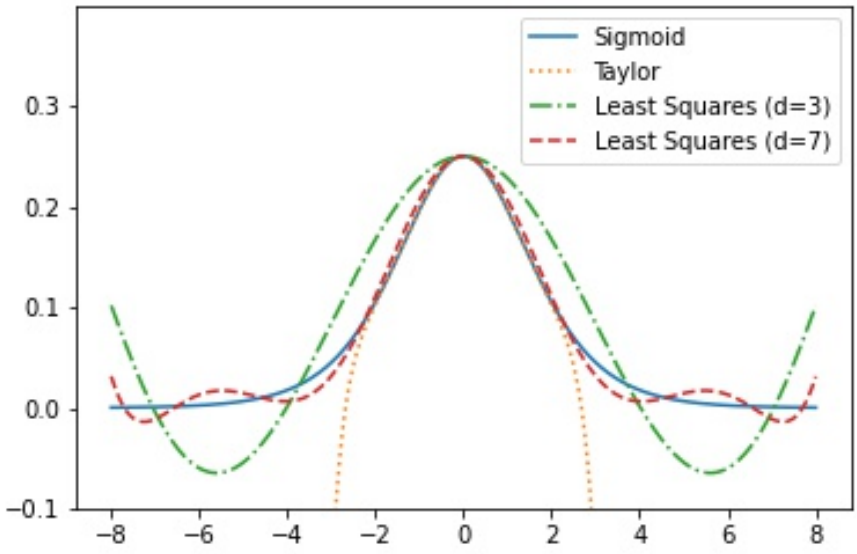


Figure 4.2: Approximations of the Derivative of the Sigmoid Function

where T_9 is the degree 9 Taylor approximation and g_d is the least squares approximation of degree d.

Using g_d polynomial approximations, we are able to evaluate the sigmoid function and actually perform gradient descent in ciphertext to approximate $p(y \mid \mathbf{x}, \mathcal{D})$ with reasonable performance. A detailed explanation of the implementation of logistic regression under HE that I used is described in Kim et al. (2018).

To verify that this polynomial approximation is reasonable in the context of logistic regression, I show how it performs both under encryption and without encryption. The following results in Table 4.1 report the accuracy, log loss, and the area under the receiver operating characteristic curve (AUC) score for the MNIST dataset for digits 3 and 8 with HEAAN and g_3, g_7 along with the results for a non-encrypted logistic regression model, where one uses the original sigmoid function and the others each use g_3, g_7. For all models, I use 42 iterations with a learning rate of 1.0, and a mini-batch of 1024 for each gradient descent step. I use a compressed version of the MNIST dataset, which is originally a 28×28 pixel image, into a 14×14 pixel image by taking the mean value of each 2×2 pixel square. The dataset has $11,982$ training samples and $1,984$ testing samples. I ran the experiment 10 times and

Encryption	Encrypted		Unencrypted		
Link	g_3	g_7	σ	g_3	g_7
Accuracy	0.9622	0.9602	0.9545	0.9630	0.9579
log loss	0.5124	0.5046	0.5012	0.5107	0.5085
AUC	0.992	0.991	0.990	0.991	0.991

Table 4.1: Performance Comparison on MNIST

report only the mean. Notice that the experiments show that the g_3 and g_7 polynomials are reasonable estimates for σ in the context of logistic regression. Furthermore, we see that encryption does not significantly change the results of the optimization process. The log loss is moderately high across all models for a binary classification problem, a result of only making 42 steps of gradient descent.

4.3 Bootstrapping for Approximate Bayesian Inference

As demonstrated in section 3.4, a point estimate of the weight vector in using logistic regression for estimating $p(y \mid \mathbf{x}, \mathcal{D})$ cannot entirely capture the value of a dataset. A point estimate cannot represent the posterior variance of the weight vector given the dataset, possibly estimating two datasets of different value to be similar. Uncertainty quantification is crucial in computing expected loss as expected loss, for certain loss functions, itself may be seen as a measure of uncertainty given the data.

Nonparametric probabilistic models such as GPs could be considered, but they involve computing the covariance matrix, taking the inverse of the covariance matrix of size $n \times n$, where n is the number of samples in $\mathcal{D}$, and an approximation method for the posterior. While Lu et al. (2017) and Hall et al. (2011) discuss iterative methods for matrix inversion under FHE for the purposes of linear regression, the above three operations in GP classification present a computational bottleneck for a reasonable implementation under HEAAN.

The computational bottleneck of matrix inversion presents a challenge for using the Laplace approximation for Bayesian logistic regression as well. Furthermore, even with a Laplace

approximation on the weight vector, computing the Gaussian cumulative density function (CDF) under FHE for $p(y \mid \mathbf{x}, \mathcal{D})$ remains a challenge as well. Using Markov chain Monte Carlo (MCMC) methods for approximate Bayesian inference is also infeasible due to most methods using computationally heavy operations or infeasible ones such as inequality operations in Metropolis-Hastings.

However, using Monte Carlo (MC) samples provides an inspiration for how to proceed with approximate Bayesian inference in our problem. Bootstrapping samples of $\mathcal{D}$ to compute samples of the posterior weight distribution $p(\mathbf{w} \mid \mathcal{D})$ and then using them for approximate Bayesian inference of $p(y \mid \mathbf{x}, \mathcal{D})$ is a viable option due to its simplicity and ease of parallel implementation.

For a reasonable approximation to the true posterior, I use three bootstrapping methods in conjunction. The first is the weighted likelihood bootstrap (WLB), or as further developed, the weighted Bayesian bootstrap (WBB). WLB was first introduced by Newton and Rafferty (1993) and was developed into WBB by Newton et al. (2021). Notice that, assuming a Gaussian prior on $\mathbf{w}$, finding the MAP of the weights in logistic regression is simply an optimization problem as such:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \ell_e(y_i, \sigma(\mathbf{w}^\top \mathbf{x}_i)) + \lambda ||\mathbf{w}||^2 \tag{4.3.1}$$

Newton et al. (2021) uses this problem formulation by simply reweighting each data point with random samples from an Exponential distribution. The resulting algorithm is given as such:

Algorithm 4.2 Weighted Bayesian Bootstrap for Logistic Regression

Input: data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, number of weight samples T

Output: weight samples from approximate posterior $\{\mathbf{w}_t\}_{t=1}^T$

for $t = 1, \ldots, T$ **do**

  Sample $(u_0, \ldots, u_n) \sim \text{Uniform}(0, 1)$

  $\theta_i \leftarrow \log(1/u_i)$ for $i = 0, \ldots, n$

  $\mathbf{w}_t \leftarrow \arg\min_{\mathbf{w}} \sum_{i=1}^n \theta_i \ell_e(y_i, \sigma(\mathbf{w}^\top \mathbf{x}_i)) + \theta_0 \lambda ||\mathbf{w}||^2$

return $(\mathbf{w}_1, \ldots, \mathbf{w}_T)$

Despite being a simple algorithm, the WBB converges asymptotically to a Gaussian approximate of the posterior, effectively returning samples from a Laplace approximation of the posterior. Details of this convergence analysis can be seen in Newton and Rafferty (1993) and Newton et al. (2021). The second bootstrapping method I use is bootstrapping with noise, appealing to Raviv and Intrator (1996), but with a simpler implementation. With a small percentage $\epsilon \in (0, 1)$, I flip the labels y_i for each bootstrapped sample to force the optimization loop to output a weight vector that is not too close to the MAP. After various experiments, I found $\epsilon = 0.01$ to work best, but this may vary for each situation. Finally, I also use the m out of n bootstrap, appealing to Cheung et al. (2005). I found that using between $n^{2/3}$ to $n^{3/4}$ samples for each bootstrapped sample works best.

4.4 Data Evaluation Pipeline

Using logistic regression over HEAAN with bootstrapping for approximate Bayesian inference, we are ready to finalize the secure multi-party computation (MPC) protocol with which datasets may be evaluated in a market setting. We have three parties: the buyer, the seller, and the U3P. In this section, we denote the encrypted version of any data or number with $'$ unless noted otherwise. First of all, the buyer generates a public key pk, a private key sk, and an evaluation key evk, sharing the public key to the seller and the evaluation key to the U3P. The key generation service may be provided by the U3P, but parameterized by the buyer's own secure passcode so that the U3P cannot reverse engineer the keys. Then, if the buyer has their own dataset $\mathcal{D}_b$ they would like to improve, they encrypt it to $\mathcal{D}_b' = \mathsf{Encrypt}(pk, \mathcal{D}_b)$ and send it to the U3P who computes the encrypted expected loss $L'(\mathcal{D}_b) = \mathsf{Evaluate}(evk, L, \mathcal{D}_b')$. Then, the seller encrypts their dataset $\mathcal{D}_s$ into $\mathcal{D}_s' = \mathsf{Encrypt}(pk, \mathcal{D}_s)$ which is sent to the U3P. Next, the U3P computes the encrypted expected loss of $\mathcal{D}_b \cup \mathcal{D}_s$, denoted $L'(\mathcal{D}_b \cup \mathcal{D}_s) = \mathsf{Evaluate}(evk, L, \mathcal{D}_b' \cup \mathcal{D}_s')$. Finally, the U3P sends $L'(\mathcal{D}_b)$ and $L'(\mathcal{D}_b \cup \mathcal{D}_s)$ to the buyer, who then decrypts the values for evaluation: $L(\mathcal{D}_b) = \mathsf{Decrypt}(sk, L'(\mathcal{D}_b)), L(\mathcal{D}_b \cup \mathcal{D}_s) = \mathsf{Decrypt}(sk, L'(\mathcal{D}_b \cup \mathcal{D}_s))$. This protocol is illustrated in Figure 4.3. Note that the $\mathsf{Evaluate}(evk, L, \mathcal{D}_b' \cup \mathcal{D}_s')$ operation involves the steps discussed in chapters 3 and 4, namely Bayesian model averaging (BMA) and bootstrapping for approximate Bayesian inference. For computing the BIC for BMA under encryption, I again use a least-squares approximation of degree 3 over $[0, 1]$ for the log function.

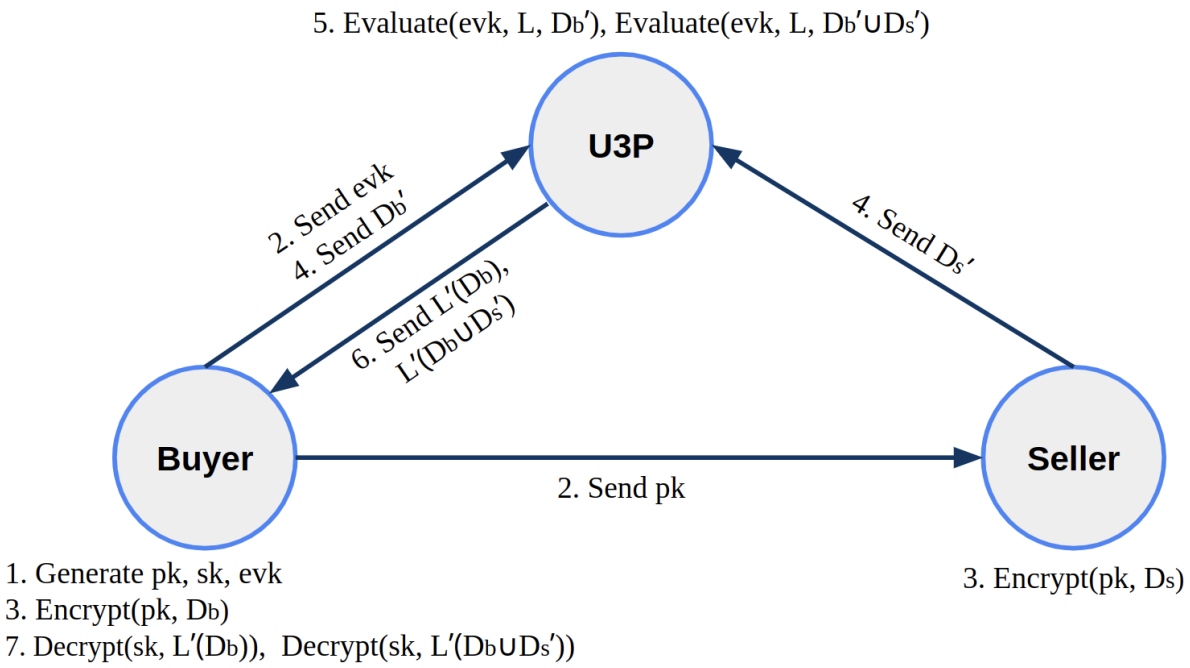


Figure 4.3: Secure MPC Protocol for Computing Expected Loss

Notice that the U3P only works with encrypted data, without any access to the public key nor the private key, preventing from any of the data being compromised. While the U3P has access to the evaluation key, which is essential for computing multiplication under HEAAN, the hardness of the RLWE problem prevents the U3P from being able to solve the private key. Furthermore, the buyer is only given the encrypted expected losses without ever accessing the seller's data directly, which is exactly what we want in this scenario. Finally, since the seller only really provides their data without anything in return, they cannot infer anything. Notice that both the buyer and the seller cannot exploit this protocol by carefully manipulating their datasets, preventing this pipeline being susceptible to adversarial buyers and sellers too. Therefore, we see that the given MPC protocol is secure under a semi-honest actors assumption.

Chapter 5

Experiments

In this chapter, I demonstrate several experiments that use the full MPC protocol outlined in Chapter 4. First, I perform experiments on synthetic data, primarily focusing on showing the effects of BMA and bootstrapping for approximate Bayesian inference. Second, I use a real-life medical diagnostic dataset, dividing it into numerous datasets according to the distance to the maximal-margin hyperplane. For the second experiment, I implement a customized loss function and describe what decision a buyer would make according to the results.

For all experiments, I report the expected loss L, the actual loss $\mathcal{L}$, and real time spent. I use 100 bootstrap runs to approximate Bayesian inference. Furthermore, to emulate real life situations, instead of explicitly defining a distribution over the domain with which to take the expectation, I use a test dataset to serve as a MC approximation to the expected value. In HEAAN, I use g_3, the 3rd degree polynomial least-squares approximation to the sigmoid function. For all experiments, I use 6 cores (12 threads) of Intel(R) Xeon(R) CPU @ 2.30Hz with 32GB of memory.

5.1 Synthetic Data with Different Patterns

For the synthetic dataset for $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$, the underlying function that decides the label is:

$$y = \mathbb{1}[x_1^2 + x_2^2 + x_3^2 + \varepsilon < 3^2]$$

where $\varepsilon \sim \mathcal{N}(0, 0.1^2)$, making the underlying boundary the surface of a 3-dimensional sphere with radius 3 centered at 0 with small error. Then, the following datasets $\mathcal{D}_i$ were generated:

$\mathcal{D}_1$: 50 points with $x_1 \sim \mathcal{N}(1, 0.5^2); x_2, x_3 \sim \mathcal{N}(0, 0.5^2)$, 50 points with $x_1 \sim \mathcal{N}(5, 0.5^2); x_2, x_3 \sim \mathcal{N}(0, 0.5^2)$

$\mathcal{D}_2$: 100 points with $x_1 \sim \mathcal{N}(3, 1^2); x_2, x_3 \sim \mathcal{N}(0, 1^2)$

$\mathcal{D}_3$: 100 points with $x_1, x_2, x_3 \sim \mathcal{N}(0, 2^2)$

$\mathcal{D}_4$: 100 points uniformly distributed over the surface $x_1^2 + x_2^2 + x_3^2 = 3$ with each term perturbed by $\delta \sim \mathcal{N}(0, 0.1^2)$

Notice both $\mathcal{D}_1$ and $\mathcal{D}_2$ have points from a specific part of the domain such that the data cannot capture the entire sphere shape of the true boundary. On the other hand, $\mathcal{D}_3$ and $\mathcal{D}_4$ are distributed well enough such that the data captures the sphere shape, but $\mathcal{D}_3$ has points throughout the domain while $\mathcal{D}_4$ has points closely packed around the boundary. The expected order of quality is: $\mathcal{D}_4 > \mathcal{D}_3 > \mathcal{D}_2 > \mathcal{D}_1$.

The test dataset $\mathcal{D}_t$ which will serve as a sample of the test distribution, is 1000 points with $x_1 \sim \mathcal{N}(-3, 1^2)$, $x_2 \sim \mathcal{N}(1, 1^2)$, $x_3 \sim \mathcal{N}(0, 3^2)$. Furthermore, for this problem, I assume the buyer has no prior dataset. Finally, I use the square-root of the ℓ_2 loss function, with the square-root taken so that the loss is in the same units as the probability.

For BMA, we use the following models with an uninformative prior that places equal probability to each model. Since all models are generalized linear models (GLM) with the logistic link function, I identify each model $\mathcal{M}_j$ by the corresponding feature transformation $\phi_j : \mathbb{R}^3 \to \mathbb{R}^{d_j}$:

$\phi_1(x_1, x_2, x_3) = [x_1, x_2, x_3]^\top$

$\phi_2(x_1, x_2, x_3) = [x_1^2, x_2^2, x_3^2]^\top$

$\phi_3(x_1, x_2, x_3) = [x_i x_j \text{ for } i, j = 1, 2, 3]^\top$

	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$
Expected Loss	0.2091	0.1697	0.1479	0.1104
Actual Loss	0.2154	0.1527	0.1150	0.1178
Time	752 mins	749 mins	750 mins	753 mins

Table 5.1: Synthetic Dataset

First, running BMA with $\mathcal{D} = \bigcup_i \mathcal{D}_i$ with encryption, using the BIC as an approximation for the posterior probabilities, we have:

$$\mathbb{P}(\phi_1 \mid \mathcal{D}) = 0.0001 \quad \mathbb{P}(\phi_2 \mid \mathcal{D}) = 0.9989 \quad \mathbb{P}(\phi_3 \mid \mathcal{D}) = 0.0010$$

Notice that the model ϕ_2, which is what the true boundary uses, has the highest model posterior probability by a large margin due to having a reasonable model likelihood with a few number of parameters. Due to the large gap between the posterior probabilities, I resort to use Bayesian model selection instead, choosing ϕ_2. Then, using these results, we get the following results in Table 5.1 for each dataset. The time reported excludes the time used for executing BMA.

Here, we see that the order of both expected loss and actual loss follow the expected order of quality, except for the actual loss for $\mathcal{D}_3$ being lower than that for $\mathcal{D}_4$ by a small amount. However, we also see that even for a small dataset of low dimensionality, the process of bootstrapping and computing under encryption leads to the computation time being very long.

5.2 Medical Diagnostic Data with Variable Loss Function

For a real life dataset, I use the Breast Cancer dataset that classifies breast cancer cells according to whether they are malignant (1) or benign (0) (Zwitter and Soklic, 1992). For fast implementation, using the knowledge that a linear model fits the data well, I do not

	$\mathcal{D}_b$	$\mathcal{D}_b \cup \mathcal{D}_1$	$\mathcal{D}_b \cup \mathcal{D}_2$	$\mathcal{D}_b \cup \mathcal{D}_3$	$\mathcal{D}_b \cup \mathcal{D}_4$
Expected Loss	29978.25	10766.92	15982.82	19945.82	19979.60
Actual Loss	28653.56	12260.91	16200.54	17278.23	18723.13
Time	570 mins	910 mins	912 mins	911 mins	910 mins

Table 5.2: Breast Cancer Dataset

perform BMA and simply use a GLM with the logistic link function for evaluating the posterior predictive distribution. I add that this will likely be the case for many real-life situations involving tabular data.

Using a dataset of a total of 569 samples, 149 samples were randomly chosen as the test dataset that will serve as a MC approximation to the domain distribution. Then, the sellers' datasets and the buyer's dataset were built as such. First, a support-vector machine (SVM) model is fitted to the entire data. Then, the points excluding the points in the test dataset are sorted in increasing order according to their distance to the maximal margin hyperplane found by the SVM. Then, the closest points in each target class are added in order to the following datasets: 100 to $\mathcal{D}_1$, 100 to $\mathcal{D}_2$, 100 to $\mathcal{D}_3$, 20 to $\mathcal{D}_b$, and the last 100 to $\mathcal{D}_4$, where $\mathcal{D}_b$ corresponds to the buyer's dataset. Note that the expected order of quality is $\mathcal{D}_1 > \mathcal{D}_2 > \mathcal{D}_3 > \mathcal{D}_4$. Furthermore, we expect $\mathcal{D}_4$ to not significantly reduce the expected loss once augmented onto $\mathcal{D}_b$. This entire process is done such that the label ratio of the entire dataset $(357 : 212)$ is preserved as much as possible for each dataset.

For this problem, I use a special loss function that heavily penalizes type II errors compared to type I errors, and also factors in the monetary cost for each error. The custom loss function is given as such:

$$\ell(y = 1, \pi(\mathbf{x})) = 100000 \cdot \pi(\mathbf{x})$$
$$\ell(y = 0, \pi(\mathbf{x})) = 20000 \cdot (1 - \pi(\mathbf{x}))$$

Then, we have the following results in Table 5.2.

Notice again that the order of quality according to expected loss and actual loss both follow the expected order of quality. Furthermore, we are able to see that all datasets provide a

meaningful decrease in loss once augmented onto the buyer's dataset. Finally, we are able to make a reasoned decision to buy $\mathcal{D}_1$ as long as the price is at most the expected decrease in loss, which is 19211.33. The actual marginal gain comes out to be 16392.65, which is lower than the expected decrease in loss and therefore the willingness to pay, but such errors are cannot be fully prevented.

Chapter 6

Conclusion

The expected loss corresponding to the posterior predictive distribution induced by a dataset is a flexible and theoretically sound method of estimating the value of that dataset. When using common loss functions, we are able to relate the expected loss to either the posterior variance of hypotheses or the information gain of the dataset. Furthermore, the definition allows for flexible configurations of the loss function that fits particular situations. However, computing it requires careful consideration of uncertainty, which can be addressed by using Bayesian inference with Bayesian model averaging (BMA) or non-parametric models. In real-life implementations, homomorphic encryption is a promising encryption method that allows for computing the expected loss without the judged data being compromised. Seeing this in action with synthetic and real-life data, we are able to see reasonable results that correctly determine the qualities of datasets, albeit with some error which is to be expected.

While a promising idea and possibly an opportunity for a new industry to bloom, there are two limitations in this proposed idea, both related to homomorphic encryption (HE). The first limitation is the lack of complicated operations feasible with HE. While the logistic function was approximated, the limit to polynomial functions greatly hinders various interesting loss functions from being used in this framework. This is also why Chapter 5 had to resort to simple polynomial loss functions, instead of a more commonly used classification loss function such as the log loss function. Another method that is not feasible due to limitations in the type of computation that HE allows is Gaussian process (GP) models. As mentioned in section 4.3, the inversion of the Gram matrix is not feasible for moderately sized datasets as current matrix inversion implementations under HE rely on iterative methods (Hall et

al., 2011). In general, more approximate matrix operations being implemented with great precision is required for the use of non-parametric methods and interesting loss functions.

The second, and perhaps the more important limitation is the time spent on computing the expected loss. Without HE, each experiment in Chapter 5 takes less than 10 seconds to run, whereas using HE leads to evaluating one dataset to take over 12 hours for very small datasets. This is why this work is limited to datasets of small sample size and dimensionality. For this work to be feasible, much faster schemes in HE must be implemented so that datasets may be evaluated promptly. This would also allow room for using more complicated models, such as neural networks.

References

[1] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[2] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648, 1996.

[3] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437, 2017.

[4] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[5] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. 1987.

[6] Rob Hall, Stephen E Fienberg, and Yuval Nardi. Secure multiple linear regression based on homomorphic encryption. *Journal of Official Statistics*, 27(4):669, 2011.

[7] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Logistic regression on homomorphic encrypted data at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9466–9471, 2019.

[8] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, Xiaoqian Jiang, et al. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics*, 6(2):e8805, 2018.

[9] Wen-jie Lu, Shohei Kawasaki, and Jun Sakuma. Using fully homomorphic encryption for statistical analysis of categorical, ordinal and numerical data. *Cryptology ePrint Archive*, 2016.

[10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 1–23, 2010.

[11] David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2005.

[12] Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[13] Michael A Newton, Nicholas G Polson, and Jianeng Xu. Weighted bayesian bootstrap for scalable posterior distributions. *Canadian Journal of Statistics*, 49(2):421–437, 2021.

[14] Michael A Newton and Adrian E Raftery. Approximate bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(1):3–26, 1994.

[15] Netanel Raviv, Siddharth Jain, and Jehoshua Bruck. What is the value of data? on mathematical methods for data quality estimation. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2825–2830, 2020.

[16] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[17] Oded Regev. The learning with errors problem. *Invited survey in CCC*, 7(30):11, 2010.

[18] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[19] Burr Settles. Active learning literature survey. 2009.

[20] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[21] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.