Towards Model-based Pricing for Machine Learning in a Data Marketplace

Lingjiao Chen[1] Paraschos Koutris[1] Arun Kumar[2]
[1]University of Wisconsin-Madison [2]University of California, San Diego
{lchen, paris}@cs.wisc.edu, arunkk@eng.ucsd.edu

ABSTRACT

Data analytics using machine learning (ML) has become ubiquitous in science, business intelligence, journalism and many other domains. While a lot of work focuses on reducing the training cost, inference runtime and storage cost of ML models, little work studies how to reduce the cost of data acquisition, which potentially leads to a loss of sellers' revenue and buyers' affordability and efficiency. In this paper, we propose a *model-based pricing* (MBP) framework, which instead of pricing the data, directly prices ML model instances. We first formally describe the desired properties of the MBP framework, with a focus on avoiding *arbitrage*. Next, we show a concrete realization of the MBP framework via a noise injection approach, which provably satisfies the desired formal properties. Based on the proposed framework, we then provide algorithmic solutions on how the seller can assign prices to models under different market scenarios (such as to maximize revenue). Finally, we conduct extensive experiments, which validate that the MBP framework can provide high revenue to the seller, high affordability to the buyer, and also operate on low runtime cost.

CCS CONCEPTS

• **Information systems** → **Data management systems**; • **Computing methodologies** → **Machine learning**; • **Theory of computation** → *Algorithmic game theory and mechanism design*;

KEYWORDS

Machine Learning, Pricing, Data Market, Mechanism Design

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06.
https://doi.org/10.1145/3299869.3300078

ACM Reference Format:
Lingjiao Chen, Paraschos Koutris, Arun Kumar. 2019. Towards Model-based Pricing for Machine Learning in a Data Marketplace. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3299869.3300078

1 INTRODUCTION

Data analytics using machine learning (ML) is an integral part of science, business intelligence, journalism, and many other domains. Research and industrial efforts have largely focused on performance, scalability and integration of ML with data management systems [17, 30, 44]. However, limited research so far has studied the *cost of acquiring data* for ML-based data analytics.

Users often buy rich structured (*relational*) data to train their ML models, either directly through companies (e.g., Bloomberg, Twitter), or through *data markets* (e.g., BDEX [1], Qlik [2]). Such datasets are often very expensive due to the immense effort that goes into collecting, integrating, and cleaning them. Existing pricing schemes either force users to buy the whole dataset or support simplistic pricing mechanisms, without any awareness of the ML task downstream (e.g., the dataset is used to train a predictive model). This means that valuable datasets may not be affordable to potential buyers with limited budgets, and also that data sellers operate in an inefficient market, where they do not maximize their revenue. Simplistic pricing schemes may also create undesirable arbitrage opportunities, where the desired data can be acquired by combining data fragments that together cost a cheaper price. Thus, as [18] also points out, *there is a need to transition from markets that sell only data to markets that can directly sell ML models as well.*

Model-based Pricing. In this paper, we take a first step towards the long-term vision of creating a marketplace for selling and buying ML models, by presenting *a formal and practical fine-grained pricing framework for machine learning over relational data.* Our key observation is that, instead of selling raw data to the buyers, the market can directly sell *ML model instances* with different accuracy options. The price then depends on the accuracy of the model purchased, and

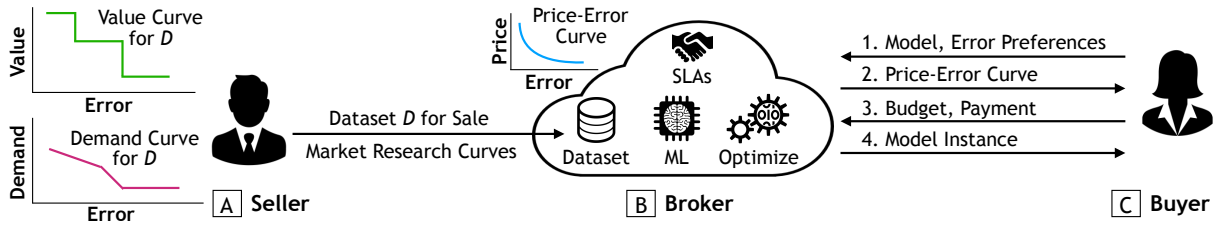

Figure 1: Model-based pricing market setup. (A) The *seller* is the agent who wants to sell ML model instances trained on their commercially valuable dataset D. (B) The *broker* is the agent that mediates the sale for a set of supported ML models and gets a cut from the seller for each sale. (C) The *buyer* is the agent interested in buying a ML model instance trained on D.

not the underlying dataset. Since the price is based on the model, we call our framework *model-based pricing* (MBP).

A high level view of MBP is demonstrated in Figure 1. The data market involves three agents, namely, the *seller* who provides the datasets, the *buyer* who is interested in buying ML model instances, and the *broker* (market) who interacts between the seller and the buyer. First, the seller and/or the broker perform market research to ascertain curves representing demand and value for the ML model instances among potential buyers. These curves plot demand and value as a function of the error/accuracy of the ML model trained. The broker uses the market research information to build price-error curves that are presented to the buyer. The buyer specifies a desired price or error budget and pays the broker, who computes an appropriate ML model instance, and returns it to the buyer.

Example 1. Alice is a journalist studying the relationship between demographics and economic indicators for an upcoming article. She wants to test how predictive some demographic features are of the average annual household income. Datasets with such information exist online, but they are expensive and exceed Alice's budget. In this scenario, a data marketplace with MBP would allow Alice to be charged only based on her ML task and desired accuracy. In particular, a linear regression model instance with a square loss might be sufficient for Alice's purposes.

Example 2 Bob is a business analyst who wants to study if a social media message is related to his own company. The data is available but expensive (*e.g.*, Twitter's GNIP API [3] enables users to pay for aggregate data about an *audience*, which is a set of users with the same age, location, etc). In such cases, MBP can not only help Bob achieve trade-off between model accuracy and budget constraints, but also increase the market size and thus profit of the data sellers.

Desiderata and Challenges. Achieving the MBP framework is a technically challenging task, from both a theoretical and practical point of view. First, in order to guarantee *affordability*, the MBP framework must allow buyers with different budgets to buy model instances with different accuracy guarantees. Second, the model instance generation and pricing should be performed *efficiently*, since model training is typically time-consuming. Third, the MBP framework must prevent *arbitrage* opportunities, where a buyer can combine model instances of low accuracy and low price to obtain a high accuracy instance for a cheaper price than the one provided by the market. For example, an instance with high accuracy should always be at least as expensive as an instance with lower accuracy. Finally, the MBP framework must provide capabilities for the broker/buyer to assign the prices such that their revenue is maximized.

Our Solution. Our key technical contribution is a simple and efficient *noise-injection mechanism* that realizes an MBP framework with formal guarantees. Specifically, the broker first trains the optimal model instance, which is a one-time cost. When a buyer requests a model instance, the broker adds random Gaussian noise to the optimal model and returns it to the buyer. Our proposed mechanism avoids training a model instance from scratch and is able to achieve real time interaction. We show that the error of the ML model instance (when certain properties are satisfied by the error function) is a monotone function of the *variance* of the noise injected in the model.

The pricing mechanism charges a price according to the variance of the noise injected to the model instance. Adding noise with low variance implies a model instance with expected low error and thus high price, while noise with high variance results in an instance with expected larger error and low price. This enables the buyer to either choose cheaper but less accurate instances or more accurate yet more expensive ones. Essentially, our mechanism provides different *versions* of the desired ML model of varying quality, in analogy to the notion of versioning in information selling [41].

Our proposed MBP mechanism comes with a concise characterization of when a pricing function is provably arbitrage-free. In the main theoretical result of this paper, we show that a pricing function is arbitrage-free if and only if the price of

a (randomized) model instance is monotone increasing and subadditive with respect to the inverse of the variance.

For revenue maximization, we establish a formal optimization framework based on the buyer's value and demand curves. We show that the optimization is a computationally hard problem even under a simple revenue model. To address this intractability, we present a novel method of relaxing the subadditive constraints, leading to polynomial time algorithms with provable approximation guarantees. Central to the revenue maximization problem is the problem of interpolating a monotone and subadditive function through given points, which could be of independent interest.

Finally, we prototype the MBP framework in standalone MATLAB, which is popular for ML-based analytics (but note that our framework is generic and applicable in other settings as well). We present an extensive empirical evaluation using both real and synthetic datasets. Our experiments validate that MBP always attains the highest revenue and provides the highest buyer affordability compared to the existing naive solutions, while simultaneously guaranteeing protection against arbitrage. We also show that our revenue maximization solution for price setting is orders of magnitude faster than brute-force search, while empirically achieving a revenue with negligible gap to the optimal revenue.

Summary of Contributions. In summary, this paper makes the following contributions.

- To the best of our knowledge, this is the first paper on a formal framework of ML-aware model-based pricing. We identify and formally characterize important properties, such as arbitrage freeness, that such a framework should satisfy.
- We propose a concrete MBP mechanism via a noise injection approach, and establish a concise characterization of the desired properties of a pricing function.
- We develop a revenue optimization framework that finds the optimal prices with the desired properties as constraints. Although it is a computationally hard problem, we provide an approximate solution which gives a pricing function with provably high revenue.
- Finally, extensive experiments validate that our proposed solution provides high revenue for the seller, large affordability ratio and thus accessibility for the buyer, and fast runtime for the broker.

Outline. Section 2 presents the problem setup and background. Section 3 introduces the MBP framework and relevant desiderata. In Section 4, we provide a concrete realization of the MBP framework through the noise injection approach. Section 5 studies the revenue optimization problem and Section 6 presents the implementation and experimental evaluation. We conclude in Section 7. All missing proofs are left to the Appendix.

2 RELATED WORK

In this section, we discuss related work on data pricing and machine learning.

Pricing Relational Queries. The problem of pricing relational data has received a lot of attention recently. The pricing schemes currently supported in data markets are typically simplistic: a buyer can either buy the whole dataset for some fixed price, or ask simple queries and be priced according to the number of output tuples. A recent line of work [5, 19–21, 25, 29, 39] has formally studied pricing schemes for assigning prices to relational queries, a setting called *query-based pricing* (QBP). In QBP, we are given a dataset D and a query Q, and the goal is to assign a price $p(Q, D)$ based on the information disclosed by the query answer. Central to it is the notion of *arbitrage*. Intuitively, whenever query Q_1 discloses more information than query Q_2, we want to ensure that $p(Q_1) \geq p(Q_2)$; otherwise, the buyer has an arbitrage opportunity to purchase the desired information for a lesser price. To formalize information disclosure, QBP uses the mathematical tool of *query determinacy* [34, 35].

At first glance, MBP seems similar to QBP. For relational queries, the price is for the information released by the query output, while for ML analytics, the price is for the information released by the model. However, there are fundamental differences: for relational queries, the buyer obtains a *deterministic* answer, while for ML analytics, the model is typically computed in an *non-deterministic* way. Also, in MBP, we enable the buyer to specify an *accuracy* constraint to control the predictive power of the model instance they buy.

Markets for ML. While several ML systems [4, 6, 26, 38] have been developed to reduce the computational cost of training ML models, there has been little attention to markets for ML until recently [18, 23]. [23] develops a market via block chain technology to allow exchange and purchasing of ML models. [18] points out the importance of creating markets for ML applications. MBP can be viewed as the first foray into how we should build such markets.

ML over Relational Data. We focus on standard supervised ML for relational/structured data, specifically, classification and regression. We are given a dataset table D with n labeled examples and d features. The target (label) is denoted Y, while the feature vector is denoted X. The labeled examples are independently and identically distributed (IID) samples from some underlying (hidden) distribution that produces the data, $P[\mathbf{X}, Y]$. An *ML model* is simply a mathematical model to approximate P in some intuitive manner. An *ML model instance* is a specific instance of that ML model that corresponds to some prediction function $f : \mathcal{D}_{\mathbf{X}} \rightarrow \mathcal{D}_Y$. The set of functions learnable (representable) by an ML model is called its *hypothesis space*. The predictive power of a model

instance is often *evaluated* using standard scores such as *hold-out test error* [12]. There are hundreds of ML models [33]; some of the most popular ones are Naive Bayes and Generalized Linear Models (GLMs). These models are popular mainly due to their *interpretability*, speed, and extensive systems support. Thus, we primarily focus on such models.

Private Query Release via Noise Injection. From a technical perspective, our framework shares the idea of injecting noise with many previous studies for private query release [7, 10, 11, 13, 15, 16, 27, 27, 28, 31, 37, 40, 42, 43]. A large portion of previous studies focuses on releasing private query results to a single user for various purposes, including relational queries [10, 15, 43], data mining [11, 16], statistical queries [13, 32], and machine learning models [7, 8, 37, 40], just to name a few. To ensure privacy while releasing data to multiple users, a few studies [27, 42] propose creating different query versions by carefully adding different noise. Similar to our approach, [28] proposes to inject different levels of noise to an ML model. Their goal is to maximize the privacy while satisfying different model accuracy requirements, whereas our target is to achieve a balance between accuracy and price in a market. In addition, almost all of those noise injection approaches require complex model noise management and do not relate the noisy models to their prediction accuracy in the context of a market.

As our mechanism injects noise, there are some interesting connections between our framework and different privacy. For example, if the Gaussian mechanism is applied, then arbitrage-freeness may imply certain connections of the privacy between different model instances. Due to space constraint, we leave in-depth analysis to future work.

To the best of our knowledge, this is the first work that studies how to release ML models via noise injection in a market which quantitatively and formally connects accuracy with noise and price.

3 MODEL-BASED PRICING FRAMEWORK

In this section, we introduce the framework of model-based pricing (MBP), and then outline some basic properties that our framework must satisfy. We summarize the notations used throughout this paper in Table 1.

3.1 Market Setup and Agents

Our framework involves three types of agents that interact in the setting of a data marketplace: the *seller*, the *broker* and the *buyer*. We now introduce our market setup involving these agents and their interactions, as well as the notation and assumptions we use. Figure 1 illustrates the market setup.

Seller. The seller provides the dataset D for sale, and it is given as a pair $(D_{\text{train}}, D_{\text{test}})$, wherein D_{train} is called the *train set* (for obtaining model instances) and D_{test} is the *test set*.

Table 1: Notations.

Symbol	Meaning
$D/D_{\text{train}}/D_{\text{test}}$	dataset/train set/test set
$n_0/n_1/n_2$	number of samples in $D/D_{\text{train}}/D_{\text{test}}$
d	number of features
$\mathbf{x}/y$	feature vector/target value (label)
$\mathcal{M}/m$	a set of ML models/a specific model
$\mathcal{H}/h$	hypothesis space/some hypothesis
$\lambda(\cdot,\cdot)/\epsilon(\cdot,\cdot)$	error function for training/testing
$h^*_\lambda(D)$	optimal model instance w.r.t. λ on D
δ	noise control parameter (NCP)
W_δ	distribution generated by δ
$w \sim W_\delta$	random variable generated by W_δ
$\mathcal{K}(\cdot,\cdot)$	randomized noise mechanism
$\hat{h}^\delta_\lambda(D)$	model generated via $\mathcal{K}(h^*_\lambda(D), w)$
$p_{\epsilon,\lambda}(\delta, D)$	price of the model instance $\hat{h}^\delta_\lambda(D)$
$\bar{p}(x)$	price of the model instance $\hat{h}^{1/x}_\lambda(D)$

This train-test split is standard in ML practice [12]. For simplicity of exposition, we express a row in D as a labeled example of the form $z = (\mathbf{x}, y)$, where $\mathbf{x} = z[\mathbf{X}]$ is the feature vector and $y = z[Y]$ is the target.

Broker. The broker specifies a menu of ML models $\mathcal{M}$ she can support (*e.g.*, logistic regression for classification and least squares for regression), along with the corresponding hypothesis spaces $\mathcal{H}_m$ for each $m \in \mathcal{M}$. For now, fix an ML model, *i.e.*, the hypothesis space $\mathcal{H}$. An *error (loss) function* $\lambda(h, D)$ measures the goodness of a hypothesis $h \in \mathcal{H}$ on D_{train} and returns a real number in $[0, \infty)$. Given D and the error function λ, let $h^*_\lambda(D) = \arg\min_{h\in\mathcal{H}} \lambda(h, D)$ denote the optimal model instance, *i.e.*, the model instance that obtains the smallest error on the training dataset w.r.t. λ. We also define another error function ϵ that can operate on either D_{test} or D_{train}, based on the buyer's preference. We use D with both the error functions, with the convention that λ operates on D_{train} and ϵ on D_{test} or D_{train}.

In general, ϵ can be different from λ because that may be more meaningful from an ML accuracy standpoint. In particular, in this paper, we focus on the following types of ML models and their associated error functions. Formally, we focus on λ that is *strictly convex*. In particular, for classification, this covers the popular logistic regression and linear SVM model (with L2 regularization). For regression with a real-valued target, this covers the popular least squares linear regression model. We think it is reasonable to focus on

these well-understood ML models and leave more complex ML models and error functions to future work. However, we emphasize that our market setup, our analyses of the properties of the pricing functions, and the revenue optimization are all generic and applicable to *any* ML model. For ϵ, we use both the same loss function as λ and the commonly used *misclassification rate* error function for classification models. Table 2 summaries those notations.

ML model	Error Function(s)
For λ; $(y, \mathbf{x})$ from train set D_{train}	
Lin. reg.	$\sum_{(y,\mathbf{x})}(y - \mathbf{w}^T\mathbf{x})^2$ $[+\mu\|w\|^2]$
Log. reg.	$\sum_{(y,\mathbf{x})} log(1 + e^{-y\mathbf{w}^T\mathbf{x}})$ $[+\mu\|w\|^2]$
L2 Lin. SVM	$\sum_{(y,\mathbf{x})} max(1, -y\mathbf{w}^T\mathbf{x}) + \mu\|w\|^2$
For ϵ; $(y, \mathbf{x})$ from test set D_{test} or train set D_{train}	
Lin. reg.	Same as λ
Log. reg.	Same as λ; $\sum_{(y,\mathbf{x})} \mathbb{1}_{y=(\mathbf{w}^T\mathbf{x}>0)}$
L2 Lin. SVM	

Table 2: ML models in $\mathcal{M}$ and associated error functions. $[\cdot]$ indicates optional regularization. All errors are typically averaged by the number of examples used.

The broker releases a model instance through a *randomized mechanism* $\mathcal{K}$ that enables them to *trade off ML error for the price* the model instance is sold for. This is a key novel technical capability in our market setup that enables model-based pricing in contrast to flat pricing. This mechanism enables us to realize *versioning* in the context of ML, in analogy to the versioning of digital information goods in micro-economics literature [41].

Specifically, $\mathcal{K}$ uses a set of parametrized probability distributions $\{\mathcal{W}_\delta \mid \delta \in \mathbb{R}_+\}$. Given a dataset D, an error function λ and a noise control parameter (NCP) δ, the broker first computes the optimal model instance $h^*_\lambda(D)$. Then, they sample $w \sim \mathcal{W}_\delta$ and output a *noisy version* of the optimal model, $\hat{h}^\delta_\lambda(D) = \mathcal{K}(h^*_\lambda(D), w)$. The NCP δ will be used as a knob to control the amount of noise added, and in turn, the price of the model instance sold. We will discuss more about the noise addition mechanism's desiderata shortly, but first, we explain the other agent in our setup.

Buyer. The buyer specifies an ML model $m \in \mathcal{M}$ they are interested in learning over D, along with their preferences for the error functions λ and ϵ to use from among the ones the broker supports. After a set of interactions with the broker, which will be explained shortly, the buyer obtains an instance of m that satisfies their price and/or error constraints.

3.2 Agent Interaction Models

Having introduced the three agents in our framework, we now explain how the market operates. Figure 1 illustrates our market setup and the interactions between the seller and broker, as well as between the broker and the buyer.

Broker-Seller Interaction Model. Apart from providing D, the seller works with the broker to determine the *pricing function* p for a given ML model. The pricing function does *not* depend solely on the released model instance $\hat{h}^\delta_\lambda(D)$. Instead, it depends on D, the NCP δ, and the two error functions λ, ϵ. Hence, we express the pricing function as $p_{\epsilon,\lambda}(\delta, D) \in [0, \infty)$. The desirable properties of a pricing function, how to set them to maximize revenue for the seller but still satisfy potential customers and run a feasible market, and how to compute them efficiently are all core technical challenges that we address later in this paper.

In the context of the interaction model, the broker is able to set the pricing functions based on two curves provided by the seller based on their *market research* about D. These curves, illustrated in Figure 2(a), tell the broker how much value potential customers attach to model errors in terms of monetary worth (value curve) and how much demand there is in the market for different model errors (demand curve). Defining and using these curves as inputs for optimizing pricing is standard practice in micro-economics for data markets such as the sale of digital videos [14].

Given the demand and value curves as a function of some error function, the broker first transforms them to demand and value curves as a function of the inverse NCP, as shown in Figure 2(b). Then, the broker computes the revenue maximizing pricing function as a function of the inverse NCP (Figure 2(c)) – we defer discussion of the revenue optimization problem till Section 5.

Broker-Buyer Interaction Model. The buyer-broker interaction has four steps, as illustrated by Figure 1(C). First, the buyer specifies the ML model they are interested in ($\mathcal{H}$) and the two error functions λ, ϵ corresponding to that model. For instance, these could be the log loss for logistic regression training but the zero-one classification error for testing. Second, the broker computes a curve that plots the price together with the *expected error* for every NCP δ, given by $\mathbb{E}_{\sim\mathcal{W}_\delta}\left[\epsilon\left(\hat{h}^\delta_\lambda(D), D\right)\right]$. This curve shows to the buyer the possible price points of the different versions of this model.

For the third step, the buyer has three options. First, she can specify a particular point on the curve (*i.e.* a price-error combination); since we know that δ behaves monotonically w.r.t. the expected error, the broker can find the unique δ^* that corresponds to that point, and obtains $\hat{h}^{\delta^*}_\lambda(D)$. The second option is that the buyer specifies an *error budget* $\hat{\epsilon}$. The broker then solves the following optimization problem:

$$\delta^* = \arg\min_\delta p_{\epsilon,\lambda}(\delta, D)$$
$$\text{s.t.} \quad \mathbb{E}_{\sim\mathcal{W}_\delta}\left[\epsilon\left(\hat{h}^\delta_\lambda(D), D\right)\right] \leq \hat{\epsilon}$$

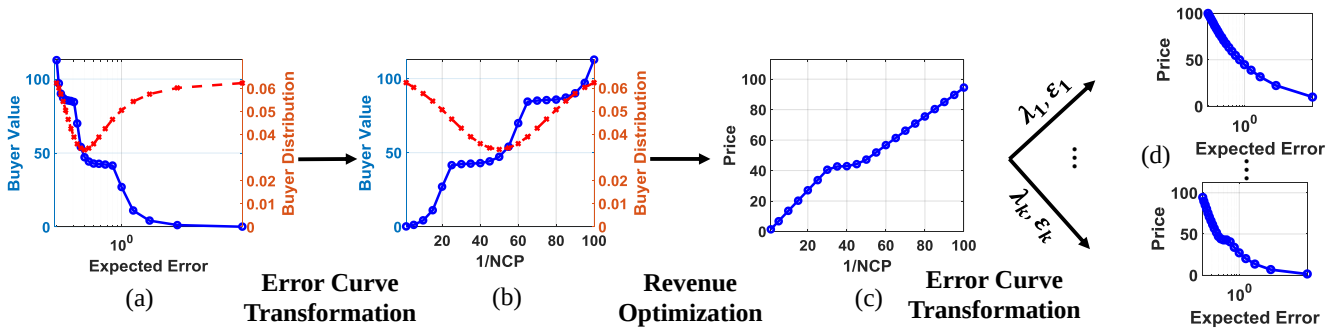

Figure 2: End-to-end model based pricing. The seller first provides the broker with the buyer value and demand curve via market research. The broker then obtains the curve v.r.t the inverse NCP via some error transformation, computes the pricing function via revenue optimization, and then returns a pricing curve to the buyers based on different error functions λ, ϵ.

The third and final option for the buyer is to specify a *price budget* $\hat{p}$ to the broker. The broker then solves

$$\delta^* = \arg\min_{\delta} \mathbb{E}_{\sim \mathcal{W}_\delta}\left[\epsilon\left(\hat{h}^{\delta}_{\lambda}(D), D\right)\right]$$
$$\text{s.t.} \quad p_\epsilon(\delta, D) \le \hat{p}$$

The third step is for the buyer to pay the price of $p_{\epsilon,\lambda}(\delta^*, D)$ to the broker. In the final step of this interaction, the broker gives the obtained model instance $\hat{h}^{\delta^*}_{\lambda}(D)$ to the buyer.

Restrictions on the Randomized Mechanism. The mechanism $\mathcal{K}$ used by the broker needs to satisfy certain properties to enable us to reason about the market's behavior and ensure it is "well-behaved" (a property we will define shortly). In particular, in this paper, we only consider randomized mechanisms that satisfy the following two conditions.

- $\mathcal{K}$ is *unbiased*, which means that:

  $$\mathbb{E}_{w\sim\mathcal{W}_\delta}\left[\mathcal{K}(h^*_\lambda(D), w)\right] = h^*_\lambda(D)$$

  In simple terms, the model instance sold after adding noise is the same as the optimal model instance *in expectation*. Only the NCP δ controls how much noise is added.
- The parameter δ behaves monotonically w.r.t. the expected error,

  $$\delta_1 \le \delta_2 \Leftrightarrow \mathbb{E}\left[\epsilon\left(\hat{h}^{\delta_1}_{\lambda}(D), D\right)\right] \le \mathbb{E}\left[\epsilon\left(\hat{h}^{\delta_2}_{\lambda}(D), D\right)\right]$$

  That is, by increasing the NCP δ we strictly increase the expected error as well, and vice versa. Its feasibility depends on the exact ϵ used. As we show later, this assumption is reasonable for many scenarios and lets us provide formal guarantees on the market's behavior.

We now present three concrete examples of how our model-based pricing framework operates based on the examples introduced in Section 1.

Example 1. *Suppose Alice obtains a schema (ID, Age, Sex, Height, AnnualIncome). Her first goal is "learning" the average annual income from a certain region (denoted by D). The hypothesis space $\mathcal{H}$ is just $\mathbb{R}$. The error functions can be simply defined as $\lambda(h, D) = (h - \bar{x})^2$, where $\bar{x}$ is the true column average from D_{train}, and similarly for ϵ on D_{train}. One possible randomized mechanism for adding noise is $\mathcal{K}_1(h^*_\lambda(D), w) = h^*_\lambda(D) + w$, where $w \sim U[-\delta, \delta]$, i.e., a uniform random distribution. Yet another possible mechanism is $\mathcal{K}_2(h^*_\lambda(D), w) = h^*_\lambda(D) \cdot w$, where $w \sim U[1-\delta, 1+\delta]$. Both of these randomized mechanisms satisfy the two restrictions listed earlier.*

Example 2. *Her second goal is to learn a least squares linear regression model to predict the income based on the features (age, sex, and height). The hypothesis space $\mathcal{H}$ is the set of all hyperplanes $h \in \mathbb{R}^d$ (where $d = 4$). The error function λ is the least squares loss defined on the training subset, i.e.,*

$$\lambda(h, D) = \frac{1}{2|D_{train}|} \sum_{z_i \in D_{train}} \left(h^T \mathbf{x}_i - y_i\right)^2.$$

The error function ϵ is the same as above, except on the test subset D_{test} of D. Given the optimal model instance $h^*_\lambda(D)$, one randomized mechanism for adding noise is as follows. Let $\mathcal{W}_\delta = \mathcal{N}(0, \delta^2)$ be the standard d-dimensional normal (Gaussian) distribution with mean 0 and variance δ^2. The noise addition mechanism is $\mathcal{K}(h^*_\epsilon(D), w) = h^*_\epsilon(D) + w$ Thus, we simply add Gaussian noise (of different magnitudes) independently to each co-efficient of the optimal model instance and return it to the buyer. Another possible mechanism is to sample noise from a zero-mean Laplace distribution. Both randomized mechanisms satisfy the two restrictions listed earlier.

Example 3. *Following the example in the introduction, suppose that Bob and the seller agree with a standard word embedding approach that maps each Twitter message to a (sparse)*

vector in a high dimensional space $\mathbb{R}^d$. Then, a standard logistic regression model is applied to determine if a message (embedded to a real vector) is related to Bob's company. Both the error function λ and ϵ are then

$$\frac{1}{2|D_{train}|} \sum_{z_i \in D_{train}} \log\left(1 + e^{-y_i h^T x_i}\right).$$

Various noise injection approaches can be used for this example scenario as well.

3.3 Pricing Function Desiderata

We now return to the concept of the pricing functions mentioned earlier when explaining the broker-seller interaction model. For the market to be able to work, these pricing functions need to satisfy a set of desiderata that provide some guarantees to both the seller and the buyer. In a sense, these guarantees act as the service-level agreement (SLA) for model-based pricing. In particular, we want the pricing functions to satisfy the following requirements.

Non-negativity. Clearly, the pricing function has to be *non-negative*, since the buyer should not be able to make money from the broker by obtaining an ML model instance.

Definition 1. *A pricing function $p_{\epsilon,\lambda}$ is* non-negative *in dataset D iff for every parameter δ (of $\mathcal{K}$),*

$$p_{\epsilon,\lambda}(\delta, D) \geq 0.$$

Error Monotonicity. Next, we want to make sure that if for a parameter δ_1, we obtain a smaller (or equal) expected error than for a parameter δ_2, then the price is larger (or equal) for the former model instance. Otherwise, a buyer that wants to buy a model instance with the smaller error can purchase it for a smaller price. This situation is illustrated in Figure 3. The formal definition is as follows.

Definition 2. *A pricing function $p_{\epsilon,\lambda}$ is* error-monotone *in dataset D if for every parameters δ_1, δ_2,*

$$\mathbb{E}\left[\epsilon(\hat{h}_\lambda^{\delta_1}(D), D)\right] \leq \mathbb{E}\left[\epsilon(\hat{h}_\lambda^{\delta_2}(D), D)\right]$$

implies that

$$p_{\epsilon,\lambda}(\delta_1, D) \geq p_{\epsilon,\lambda}(\delta_2, D).$$

Error monotonicity implies that whenever we have two parameters δ_1, δ_2 such that $\mathbb{E}\left[\epsilon(\hat{h}_\lambda^{\delta_1}(D), D)\right] = \mathbb{E}\left[\epsilon(\hat{h}_\lambda^{\delta_2}(D), D)\right]$, then the prices must be equal as well, *i.e.* $p_{\epsilon,\lambda}(\delta_1, D) = p_{\epsilon,\lambda}(\delta_2, D)$. Hence, the error monotonicity property implies that the price does not depend on the actual parameter δ of the mechanism, but on the error that this parameter induces.

Arbitrage-freeness. The final property is *arbitrage-freeness*. We first explain the importance of this property intuitively. Suppose a buyer wants to buy one model instance with a small error but large price. Suppose further she also buys more of such model instances at different prices, the sum of all of which is lower than that of the desired single model instance. Meanwhile, suppose she is able to "combine" the latter set of model instances to construct a new model instance with an error smaller than the originally desired single model instance. In this case, she would rather just buy the latter set of model instances instead of the original model instance to get an error lower than what the market is set up for. Such a situation is called *arbitrage*. For the market to work well, we need to ensure that it is *arbitrage-free*, i.e., situations such as these do not happen (or are extremely unlikely). This intuition is captured formally by the following definitions.

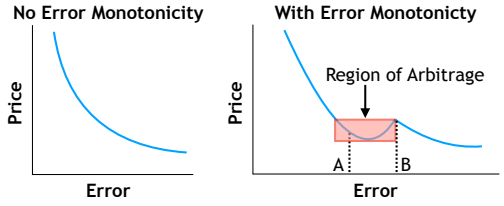

Figure 3: The pricing function on the left has error monotonicity. The one on the right does not, which leads to *arbitrage* situations. Point A has both a lower price and lower error than B. Thus, buyers are unlikely to pick B; indeed, the entire shaded region shown is useless for the seller, since they lose some potential revenue.

Definition 3 (k-Arbitrage). *We say that a pricing function $p_{\epsilon,\lambda}$ exhibits k-arbitrage in dataset D if there exist parameters $\delta_0, \delta_1, \delta_2, \cdots, \delta_k$, and a function $g : \mathcal{H}^k \to \mathcal{H}$ such that*

(1) $\sum_{i=1}^{k} p_{\epsilon,\lambda}(\delta_i, D) < p_{\epsilon,\lambda}(\delta_0, D)$, *and*

(2) $\mathbb{E}\left[\epsilon(\tilde{h}, D)\right] \leq \mathbb{E}\left[\epsilon(\hat{h}_\lambda^{\delta_0}(D), D)\right]$, *where* $\tilde{h} = g(\hat{h}_\lambda^{\delta_1}(D), \hat{h}_\lambda^{\delta_2}(D), \ldots, \hat{h}_\lambda^{\delta_k}(D))$ *s.t.* $\mathbb{E}\left[\tilde{h}\right] = h_\lambda^*(D)$.

Here, a buyer wants to combine model instances to obtain a new instance which is unbiased and has a much smaller error. This is similar to minimizing the variance while maintaining a zero bias in an ML context. Note that being unbiased is the only constraint on the power of the buyer, and there is no limit on the computational power of the buyer.

Definition 4 (Arbitrage-free). *A pricing function $p_{\epsilon,\lambda}$ is* arbitrage-free *in dataset D iff it does not exhibit k-arbitrage for any $k \in \mathbb{N}^+$.*

Not surprisingly, arbitrage-freeness implies that the pricing function is also error monotone:

Lemma 1. *If a pricing function $p_{\epsilon,\lambda}$ is* arbitrage-free *in dataset D, then it is also error-monotone in D.*

Definition 5. *We say that a pricing function $p_{\epsilon,\lambda}$ is* well-behaved *in dataset D iff it is non-negative and arbitrage-free.*

3.4 Scope and Limitations

As a first step towards a formal framework for a model-based data market, this paper focuses on the setting of supervised learning with a *fixed hypothesis space* using *a fixed set of features* and *a strictly convex model objective*. This covers a range of models including linear regression, logistic regression and linear support vector machines with 2-norm regularizers. Yet, non-convex models such as neural networks are out of scope, and feature/model selection is excluded as well.

We leave the extension of our framework to include such capabilities as future work. For instance, to enable feature selection, the market has to consider which feature combinations become available to the buyers, as well as consider arbitrage situations where models learned from different feature sets are combined together to form more accurate models. Since feature selection is a challenging problem (*e.g.*, adding more features does not necessarily imply a better model), it may be infeasible to formally characterize how model combinations behave in such a setting. Instead, we would need to weaken the arbitrage-free guarantees, and aim for approximate or probabilistic guarantees. The same challenges occur when a buyer can choose to purchase from multiple hypothesis spaces. In such cases, transfer learning techniques [36] can also be useful to measure how much information is shared across different hypothesis spaces.

4 NOISY MODEL GENERATION

So far we have presented a general framework for pricing ML models, and the interactions between the three agents. In this section, we describe a concrete instantiation of this framework, and present specific mechanisms for noise addition and price computation.

4.1 The Gaussian Mechanism

Let us fix a hypothesis space $\mathcal{H}$, such that model instances in $\mathcal{H}$ are vectors in $\mathbb{R}^d$.[1] We will focus on a specific randomized mechanism, denoted $\mathcal{K}_G$, which uses additive Gaussian noise. In particular, define $\mathcal{W}_\delta = \mathcal{N}(\mathbf{0}, (\delta/d) \cdot \mathbf{I}_d)$, for any $\delta \in \mathbb{R}_+$. Here $\mathbf{0}$ is the d-dimensional vector with all 0 entries, and $\mathbf{I}_d$ is the identity matrix with dimensions $d \times d$.

Given the two error functions ϵ, λ, a dataset D and a parameter δ, the Gaussian mechanism first computes the optimal model $h^*_\lambda(D)$ for the given error function λ and dataset D, samples a vector $w \sim \mathcal{W}_\delta$, and finally outputs $h^*_\lambda(D) + w$.

[1]For simplicity, we assume that the dimension of the models equals to the number of features d, but note that our framework works in general even if they are not equal.

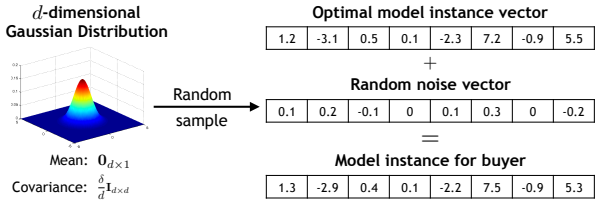

Figure 4: The Gaussian Mechanism for adding random noise to an optimal model instance.

This is illustrated in Figure 4. Formally:

$$\mathcal{K}_G(h^*_\lambda(D), w) = h^*_\lambda(D) + w, \quad w \sim \mathcal{N}(\mathbf{0}, (\delta/d) \cdot \mathbf{I}_d) \tag{1}$$

Lemma 2. *$\mathcal{K}_G$ is an unbiased randomized mechanism.*

We next analyze the Gaussian mechanism $\mathcal{K}_G$. Let us first consider a particular instantiation of the error function ϵ, the *square loss*, defined as $\epsilon_s(h, D) \triangleq \left\|h - h^*_\lambda(D)\right\|_2^2$. For square loss, we can show that the parameter δ is exactly equal to the expected error of the mechanism (and hence the parameter δ behaves monotonically w.r.t. the expected error).

Lemma 3. *Let λ, D, and $\hat{h}^\delta_\lambda(D) = \mathcal{K}_G(h^*_\lambda(D), w)$. Then:*

$$\mathbb{E}\left[\epsilon_s\left(\hat{h}^\delta_\lambda(D), D\right)\right] = \delta$$

We can show that other types of error functions behave monotonically w.r.t. the expected error (and thus δ).

Theorem 4. *Let ϵ be convex as a function of the model instance h. Let $\hat{h}^\delta_\lambda(D) = \mathcal{K}_G(h^*_\lambda(D), w)$. Then, for any two parameters δ_1, δ_2, we have*

$$\mathbb{E}\left[\epsilon(\hat{h}^{\delta_1}_\lambda(D), D)\right] \geq \mathbb{E}\left[\epsilon(\hat{h}^{\delta_2}_\lambda(D), D)\right]$$

if and only if $\delta_1 \geq \delta_2$. If ϵ is additionally strictly convex, the above holds with strict inequality ($>$).

4.2 Arbitrage for the Gaussian Mechanism

We now turn our attention to the pricing function that corresponds to the Gaussian mechanism. We show the central theoretical result of this paper, which gives us a concise characterization of an arbitrage-free pricing function when we use the Gaussian mechanism.

Theorem 5. *Let D be a dataset, and λ be an error function. A pricing function $p_{\epsilon_s,\lambda}$ is arbitrage free for the Gaussian mechanism $\mathcal{K}_G$ if and only if the following two conditions hold for every $\delta_1, \delta_2, \delta_3$:*

(1) *If $1/\delta_1 = 1/\delta_2 + 1/\delta_3$, then*

$$p_{\epsilon_s,\lambda}(\delta_1, D) \leq p_{\epsilon_s,\lambda}(\delta_2, D) + p_{\epsilon_s,\lambda}(\delta_3, D).$$

(2) *If $\delta_1 \leq \delta_2$, then $p_{\epsilon_s,\lambda}(\delta_1, D) \geq p_{\epsilon_s,\lambda}(\delta_2, D)$.*

Theorem 5 indicates that if the two conditions hold, the buyer is not able to recover the noise or the model by paying less. This is because all noise is generated independently, and therefore the Cramer-Rao Lower Bound (which we use in the proof) puts a statistical limit on how much information can be released. If a buyer was able to recover the noise or true model exactly, this would have contradicted the bound.

Theorem 5 also tells us that arbitrage-freeness is equivalent to the function $\bar{p}(x) = p_{\epsilon_s,\lambda}(1/x, D)$ being *subadditive* and *monotone* over its domain. Hence, we have a concise criterion to check for the arbitrage freeness property.

Although the square loss gives us a compact theoretical characterization of arbitrage-freeness, it is not typically used to measure the error of the model instance returned to the user. However, in the case where ϵ is a strictly convex function, we can still characterize arbitrage-freeness by applying Theorem 4. Indeed, as a corollary of Theorem 4, if ϵ is strictly convex, there exists a bijection between the expected error and the parameter δ. Thus, there exists a function ϕ, which we call the *error-inverse* of ϵ, such that:

$$\delta = \phi\left(\mathbb{E}\left[\epsilon(\hat{h}^{\delta}_{\lambda}(D), D)\right]\right)$$

Combining the above insight with Theorem 5, one can easily obtain the following result.

Theorem 6. *Let D be a dataset, and λ, ϵ be error functions. Suppose that ϵ is strictly convex, and let ϕ be its error-inverse. A pricing function $p_{\epsilon,\lambda}$ is arbitrage free for the Gaussian mechanism $\mathcal{K}_G$ if and only if the function $\bar{p}(x) = p_{\epsilon,\lambda}(1/\phi(x), D)$ is monotone and subadditive.*

In other words, arbitrage-freeness is still characterized through monotonicity and subadditivity once we view the pricing function through the transformation of the inverse map ϕ. A natural question here is how one can compute the error inverse ϕ of a given ϵ. In general, we can always compute ϕ empirically, but in several cases it is possible to compute it analytically.

5 REVENUE OPTIMIZATION

So far we have introduced the Gaussian mechanism to offer for sale noisy ML models to the buyer, and showed a simple characterization of when a pricing function is arbitrage-free under this mechanism. In this section, we will study the question of how to assign arbitrage-free prices to maximize the seller's revenue.

In this section, we fix a dataset D, the two error functions ϵ (strictly convex) and λ, and consider only the Gaussian mechanism $\mathcal{K}_G$. Instead of dealing directly with the pricing function $p_{\epsilon,\lambda}$, it will be more convenient to express the price as $\hat{p}(x) = p_{\epsilon,\lambda}(1/\phi(x), D)$, where ϕ is the error-inverse of ϵ.

Now we describe two specific scenarios of price setting, and then provide a general formalism that captures both.

Price Interpolation. Suppose that the seller wants to set the pricing function such that it takes specific prices for a set of parameters. In particular, the seller provides *n price points* of the form (a_j, P_j), where a_j is a parameter value, and P_j is its desired price. The goal is to find an arbitrage-free and non-negative pricing function such that the values $\hat{p}(a_j)$ are as close as possible to P_j.

We can capture the above setting by solving the problem of finding an arbitrage-free and non-negative pricing function that maximizes the following objective:

$$T_{\text{PI}}(x_1, \ldots, x_n) = -\sum_{j=1}^{n} \ell(x_j, P_j)$$

where $x_j = \hat{p}(a_j)$. Here, $\ell(x, y)$ can be any loss function such that $\ell(x, y) \geq 0$ and $\ell(x, y) = 0$ if and only if $x = y$. For example, we can choose $\ell(x, y) = |x - y|$, or also $\ell(x, y) = (x - y)^2$, in which case we obtain the functions T^{∞}_{PI} and T^{2}_{PI} respectively.

Revenue Maximization from Buyer Valuations. Assume that the buyers who are interested in buying a model with parameter x have a *valuation* v_x for this model. This implies that they will buy the model only if $\hat{p}(x) \leq v_x$. Moreover, we can capture "how many" buyers are interested in the particular model with parameter a_j through a parameter b_j. In this setting, the profit of the seller setting the price at $\hat{p}(a_j) = x$ is $b_j x \cdot \mathbf{1}_{x \leq v_j}$, where $\mathbf{1}_{x \leq v_j}$ is an indicator variable that takes value 1 if $x \leq v_j$, otherwise 0.

Suppose that the seller through market research has obtained the values v_j, b_j for n of these parameters (which correspond to the demand and value curves in Figure 2(a)). We can capture this setting by solving the problem of finding an arbitrage-free and non-negative pricing function that maximizes the following objective:

$$T_{\text{BV}}(x_1, \ldots, x_n) = \sum_{j=1}^{n} b_j x_j \cdot \mathbf{1}_{x_j \leq v_j}$$

where again $x_j = \hat{p}(a_j)$.

We can capture both scenarios by a general optimization problem. Specifically, *n parameter points* $\{a_1, \ldots, a_n\}$ and an objective function T are given. The goal is to find a function $\hat{p}$ that maximizes the quantity $T(\hat{p}(a_1), \ldots, \hat{p}(a_n))$ such that $\hat{p}$ is a well-behaved pricing function, *i.e.*, it is arbitrage-free and non-negative. Formally, we want to solve the following optimization problem:

$$\begin{aligned} \textbf{max}_{\hat{p}} \quad & T(\hat{p}(a_1), \ldots, \hat{p}(a_n)) \\ \textbf{subject to} \quad & \hat{p}(x+y) \leq \hat{p}(x) + \hat{p}(y), \qquad x, y \geq 0 \\ & \hat{p}(y) \geq \hat{p}(x), \qquad y \geq x \geq 0 \\ & \hat{p}(x) \geq 0, \qquad x \geq 0 \end{aligned} \tag{2}$$

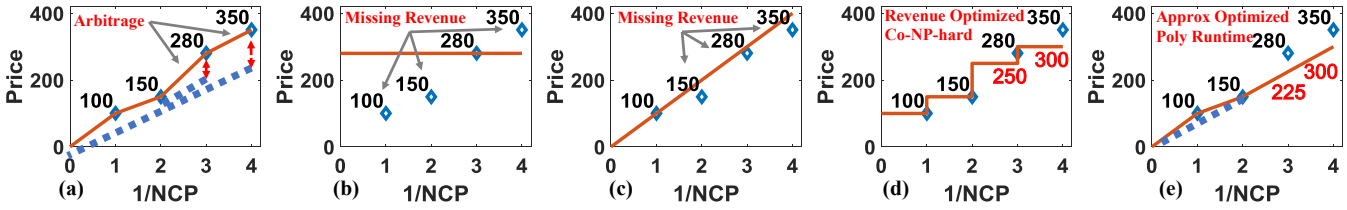

Figure 5: Illustrating example of revenue optimization. Consider a revenue maximization problem with 4 points. $a_1 = 1, a_2 = 2, a_3 = 3, a_4 = 4, b_1 = b_2 = b_3 = b_4 = 0.25, v_1 = 100, v_2 = 150, v_3 = 280, v_4 = 350$**. (a) sets all prices equal to the valuation, but it has arbitrage issue. (b) and (c) use constant and linear pricing functions, respectively. They avoid arbitrage, but they lose revenue. (d) gives the revenue-optimal pricing function, which is coNP-hard to compute. (e) is the proposed pricing function that approximates the optimal revenue well while it can be efficiently computed.**

The first two constraints in (2) capture the subadditivity and monotonicity constraints that result from the arbitrage-freeness requirement. The third constraint corresponds to the non-negative requirement. Observe that the above optimization problem is not over a set of variables, but over the space of all functions $\hat{p}$.

5.1 Hardness Results

We now study the computational complexity of solving the optimization problem (2). We will show that the problem is intractable for all objective functions $T_{\text{BV}}, T_{\text{PI}}^{\infty}, T_{\text{PI}}^{2}$. To show this hardness result, we first consider a decision problem that we call SUBADDITIVE INTERPOLATION.

DEFINITION 6 (SUBADDITIVE INTERPOLATION). *Given as input a set of points* $\{(a_j, P_j)\}_{j=1}^{n}$*, where* a_j, P_j *are non-negative rational numbers, does there exists a function* $\hat{p}$ *that* (i) *is positive, monotone and subadditive, and* (ii) *ensures* $\hat{p}(a_j) = P_j$*.*

We show in the Appendix that SUBADDITIVE INTERPOLATION is a computationally hard problem.

THEOREM 7. *The problem* SUBADDITIVE INTERPOLATION *is coNP-hard.*

Equipped with Theorem 7, we can show that the other optimization problems are also hard. Suppose that the objective function T has a unique maximizer, $(\theta_1, \ldots, \theta_n)$. Then, the optimization problem (2) will return the maximum value of T if and only if for every $j = 1, \ldots, n$ we have $\hat{p}(a_j) = \theta_j$. We can use this observation to prove the following result:

COROLLARY 7.1. *The optimization problem* (2) *with objective functions any of* $\{T_{BV}, T_{PI}^{\infty}, T_{PI}^{2}\}$ *is coNP-hard.*

5.2 Approximating Subadditivity

In order to overcome the hardness of the original optimization problem (2), we seek to approximately solve it by modifying the subadditivity constraint. In particular, we replace the subadditive constraints $\hat{p}(x + y) \leq \hat{p}(x) + \hat{p}(y)$ by the constraints $\hat{q}(x)/x \geq \hat{q}(y)/y$ for every $0 < x \leq y$. In other words, we want to find $\hat{q}$ such that $\hat{q}(x)/x$ is a decreasing function of x. The reformulated optimization problem is as follows:

$$\begin{aligned} \textbf{max}_{\hat{q}} \quad & T(\hat{q}(a_1), \ldots, \hat{q}(a_n)) \\ \textbf{subject to} \quad & \hat{q}(y)/y \leq \hat{q}(x)/x, \qquad y \geq x > 0 \\ & \hat{q}(y) \geq \hat{q}(x), \qquad y \geq x \geq 0 \\ & \hat{q}(x) \geq 0, \qquad x \geq 0 \end{aligned} \tag{3}$$

It is easy to show that for any feasible solution $\hat{q}$ of (3), the pricing function $\hat{p}(x) = \hat{q}(x)$ is also a feasible solution of (2) and hence $\hat{q}$ is a well-behaved pricing function as well.

LEMMA 8. *Any pricing function* $\hat{q}$ *that satisfies the constraints of* (3) *is arbitrage-free and non-negative.*

Approximation Guarantees. We can show that for any pricing function $\hat{p}$ that is a feasible solution of (2), we can find a feasible solution $\hat{q}$ of (3) that is not too far away from $\hat{p}$. We will use this fact later to show that our approximation does not lose too much from the optimal objective value. More precisely:

LEMMA 9. *Let* $\hat{p}$ *be a feasible solution of* (2)*. Then, there exists a feasible solution* $\hat{q}$ *of* (3) *such that for every* $x > 0$*:*

$$\hat{p}(x)/2 \leq \hat{q}(x) \leq \hat{p}(x)$$

From Functions to Variables. Problem (3) is still over the space of all possible functions. However, it turns out that we can equivalently rewrite it so that it searches over variables instead of functions. The key observation is that we only need to find the values of the function $\hat{q}$ for the n parameter points $a_1, \ldots, a_n$. In particular, consider

$$\begin{aligned} \textbf{max}_{\mathbf{z}} \quad & T(z_1, \ldots, z_n) \\ \textbf{subject to} \quad & z_j/a_j \leq z_i/a_i, \qquad a_j \geq a_i \\ & z_j \geq z_i, \qquad a_j \geq a_i \\ & z_j \geq 0, \qquad 1 \leq j \leq n \end{aligned} \tag{4}$$

The next proposition tells us that the two formulations are essentially equivalent. In particular, if $\mathbf{z}^*$ is an optimal

solution for (4), then we can use it to construct an optimal solution $\hat{q}$ for (3). function that goes through the points (a_i, z_i).

Proposition 1. *For every feasible solution of* (3)*, there exists a feasible solution of* (4) *with the same value of the objective function, and vice versa.*

We conclude this section by providing an example of the construction of the approximate optimization program.

Example 4. *Consider the revenue maximization problem with parameters as given in Figure 5. In this scenario, the optimization problem 4 can be written as follows:*

$$\begin{aligned} \textbf{max} \quad & T_{BV}(z_1, z_2, z_3, z_4) \\ \textbf{subject to} \quad & z_1 \geq z_2/2 \geq z_3/3 \geq z_4/4 \\ & z_4 \geq z_3 \geq z_2 \geq z_1 \geq 0 \end{aligned}$$

5.3 Algorithms for Revenue Optimization

In this section, we show how to leverage the approximate optimization problem (4) in order to obtain efficient algorithms with some approximation guarantees. The advantage of (4) is that it is an optimization problem with linear constraints. Depending on the objective function T, this problem can be tractable. For example, if T is concave, then (4) is a linear program with a concave objective function, which can be solved in polynomial time.

We will focus on the two problems we introduced before: price interpolation, and revenue maximization from buyer valuations.

Price Interpolation. It is easy to see that both objective functions $T_{\text{PI}}^{\infty}, T_{\text{PI}}^{2}$ are concave. This implies immediately that (4) with the above two objective functions can be solved in polynomial time.

We next show that we can also obtain an (additive) approximation guarantee. For this, we need the following general result:

Proposition 2. *Let C_{MPB}, C_{SA} be the optimal values of* (4) *and* (2) *respectively. Further let the objective function $T(z_1, \ldots, z_n) = \sum_{i=1}^{n} T_i(z_i)$, where each T_i is concave and non-positive. Then,*

$$C_{SA} + \sum_i T_i(0)/2 \leq C_{MBP} \leq C_{SA}$$

We can apply the above proposition for the objective function T_{PI}^{∞}: this implies that the optimal value of the approximate solution will be at most $(\sum_j P_j)/2$ away from the optimal solution. Similarly, we can obtain an additive approximation guarantee of $(\sum_j P_j^2)/2$ for T_{PI}^{2}.

Revenue Maximization from Buyer Valuations. In contrast to price interpolation, T_{BV} is not a concave function. However, it turns out that (4) can still be solved in polynomial time, and moreover, that the optimal solution is within a constant factor of the optimal solution of (2). Indeed:

Table 3: Dataset Statistics.

Task	DataSet	n_1	n_2	d
Regression	Simulated1	7500000	2500000	20
	YearMSD	386509	128836	90
	CASP	34298	11433	9
Classification	Simulated2	7500000	2500000	20
	CovType	435759	145253	54
	SUSY	3750000	1250000	18

Proposition 3. *Let C_{MPB}, C_{SA} be the optimal values of* (4) *and* (2) *respectively under the objective function T_{BV}. Then,*

$$C_{SA}/2 \leq C_{MBP} \leq C_{SA}$$

To solve the optimization problem (4), we resort to dynamic programming. We defer to all the details in the Appendix; our result can be stated as follows.

Theorem 10. *There exists a dynamic programming algorithm that computes the optimal solution of* (4) *with objective function T_{BV} in running time $O(n^2)$.*

6 EXPERIMENTS

The goal of the experimental section is three-fold: (*i*) validate that the ML model accuracy/error is monotone with respect to the inverse of the noise control parameter 1/NCP (which is the variance of the Gaussian noise), (*ii*) show that the MBP framework can generate more revenue for the sellers, while more buyers have access to purchasing ML models, and (*iii*) demonstrate that our algorithms run much faster than a naive brute-force search for the revenue optimization problem, while still providing near-optimal revenue.

Experimental Setup. All experiments were run on a machine with 4 Intel i5-6600 3.3 GHz cores, 16 GB RAM, and 500 GB disk with Ubuntu 14.04 LTS as the OS. We have prototyped the MBP framework in Matlab 2017b.

6.1 Expected Error to 1/NCP Transformation

The first question is whether it is true that the expected model accuracy/error behaves always monotonically as a function of 1/NCP, *i.e.*, the inverse of the noise control parameter. In Section 4, Theorem 4 gives a positive answer for the case of strictly convex error function. Here we present an empirical study on different error functions.

We use six datasets that are summarized in Table 3. The first three datasets, Simulated1, YearMSD, and CASP are for the regression task (Linear Regression), while the next three

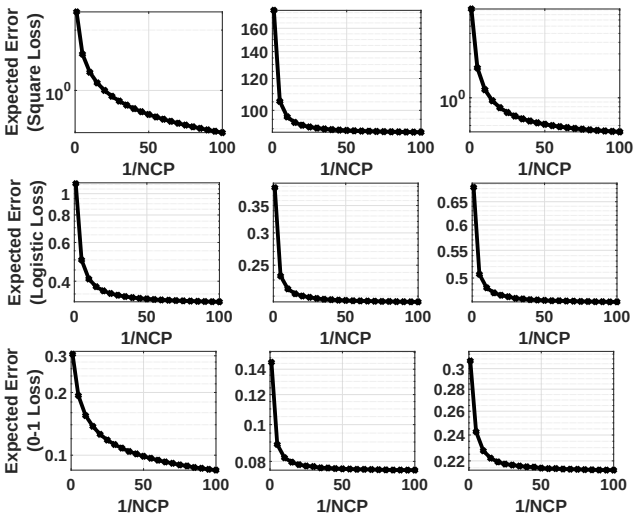

Figure 6: Error Transformation Curve. All errors are measured on the testing datasets. The first row corresponds to the square loss for Simulated1, YearMSD, and CASP, respectively. The second row represents the logistic loss, while the third row shows the 0/1 classification erro for Simulated2, CovType, and SUSY.

datasets, Simulated2, CovType and SUSY are for the classification task (Logistic Regression). The feature vectors of the dataset Simulated1 and Simulated2 are generated from a normal distribution. The target values of Simulated1 are simply the inner product of the feature vectors and a hyperplane vector. The label value of a data point from Simulated2 is 1 with probability 0.95 if it is above a given hyperplane, and 0 otherwise. The other datasets are all from the UCI machine learning repository [9]. For each value of the NCP, we generate 2000 random models, each of which is equal to the optimal model plus an independently randomly generated vector with the same variance.

As shown in Figure 6, the testing error decreases as 1/NCP increases. This verifies that there is a monotone mapping and hence the error transformation is feasible. Interestingly, even when the error is not strictly convex, such as the 0/1 error, the expected error still decreases as 1/NCP increases. This might be because all model instances are trained and tested on a relatively large datasets and thus have good generalization error. Thus, the 0/1 classification error can be well approximated by the strictly convex loss function.

In the following part of the experiments, we will focus on the parameter 1/NCP, which is monotone with and thus represents the expected error.

6.2 Revenue and Affordability Gain

Next we study the benefits of our proposed MBP approach on the seller's revenue and buyer's affordability ratio (fraction

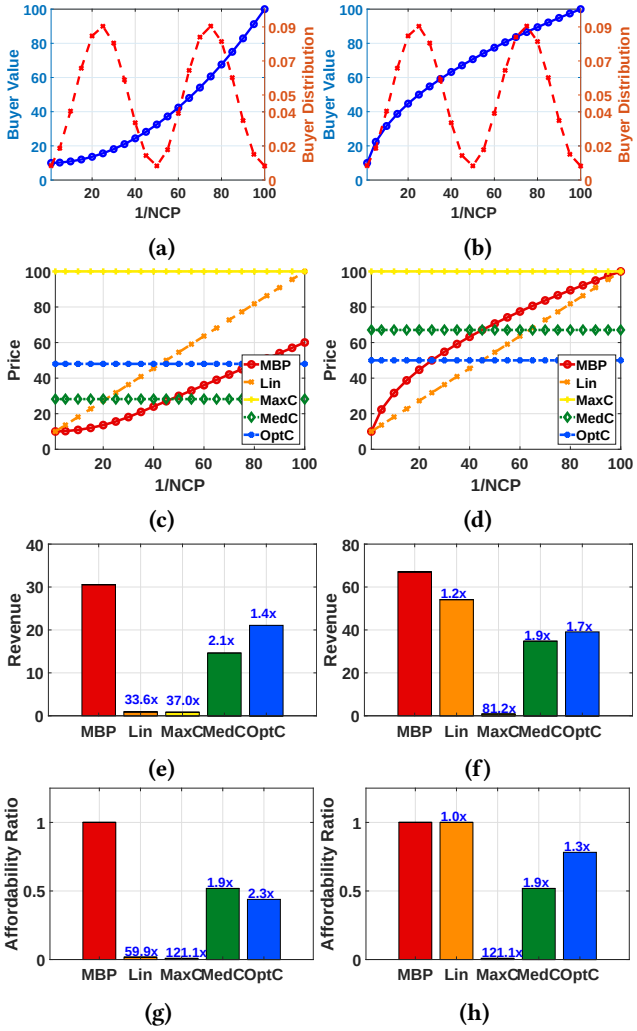

Figure 7: Revenue and Affordability Gain. The buyer distribution is fixed and we vary the buyer value curve.

of the buyers that can afford to buy a model instance) compared to other approaches of pricing ML models. Consider the setting of revenue maximization from buyer valuations as described in Section 5, *i.e.*, a buyer would pay for a model instance if and only if the price is less than the buyer's valuations. We compare MBP with four pricing approaches, all of which obtain well-behaved pricing functions.

- **Lin**, the linear approach, uses a linear interpolation of the smallest and largest value in the buyer's value curve to set the price.
- **MaxC** sets a single price to all instances, using the highest value in the buyer's value curve.
- **MedC** sets a single price to all instances such that at least half of the buyers afford to buy an instance.

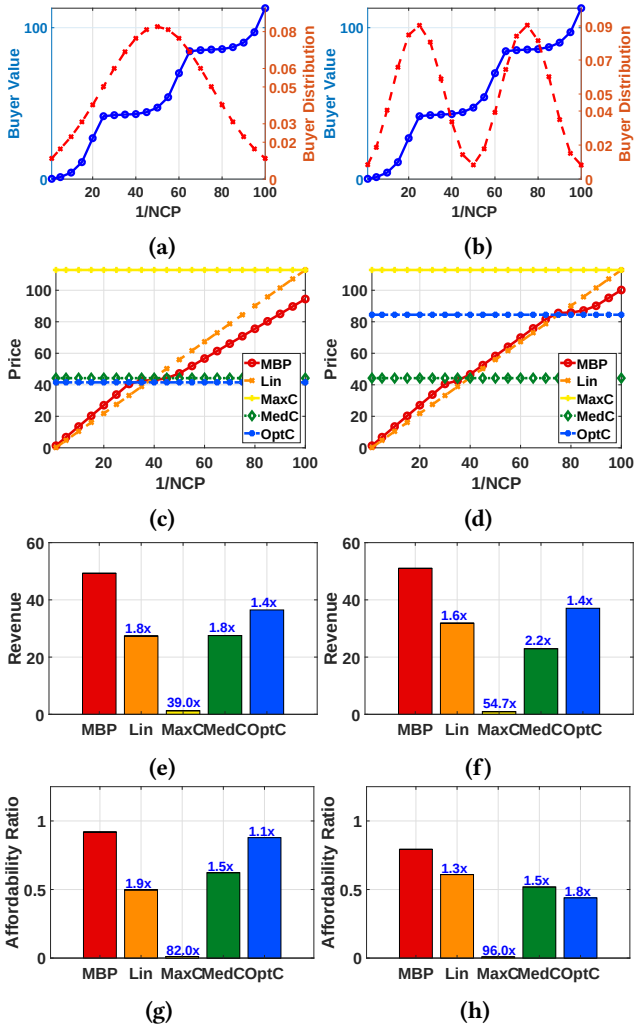

Figure 8: Revenue and Affordability Gain. We fix the buyer valuation and vary the buyer distribution.

- **OptC** sets a single price to all instances, using the one that maximizes the seller's profit.

Figure 7 and 8 show the results under different buyer value and demand curves, respectively. Overall, MBP can achieve up to 81.2x revenue gains and up to 121.1x affordability gains compared to the four baseline approaches.

We first fix the buyer distribution and vary the buyer valuation. As shown in Figure 7(a), when the value curve is convex, MBP obtains significantly more revenue and affordability compared to the linear approach. This is because the linear approach misses the opportunities to sell model instances to buyers interested in buying model instances with medium accuracy. When the buyer curve becomes concave as shown in Figure 7(b), the other approaches lose more revenue, since they set a single price, and cannot accordingly change the price for different buyers. Meanwhile, MBP achieves the largest revenue gains and affordability, as a concave function is also a subadditive function and thus MBP can match exactly the value curve.

Next, we fix the buyer value curve and vary the demand curve. As shown in Figure 8, when most of the buyers are interested in buying model instances with medium accuracy, MBP tends to produce a price function that ties close to the price for model instance with medium accuracy. When most buyers are interested in buying extremely low and extremely high accurate model instances, MBP can accordingly change the price function to follow the different requirement. As shown in Figure 8(c) and 8(d), none of the other methods is able to capture this. The ability to adjust to different buyer valuations and distribution curves explains why MBP can achieve the largest revenue gain and affordability ratio.

6.3 Runtime Performance

Finally, we present experimental results on the runtime performance of the revenue optimization algorithms under MBP. Fixing the buyer curve, we vary the number of pricing points and compare the runtime and revenue gains of our MBP proposed, versus the optimal yet expensive optimal algorithm MILP (a multiple-integer-linear programming based approach) as well as all the other four baseline methods.

Figures 9 and 10 present how the runtime, revenue, and affordability ratio vary as the buyer distribution and value curve change. Overall, MBP is always more than several orders of magnitude faster than the naive MILP. This is because MILP requires solving integer linear programming exponentially many times. Since MBP is an algorithm requiring only quadratic runtime, its runtime is much smaller. While other naive pricing methods are slightly faster than MBP due to their simplicity, they almost always suffer from either reduced revenue gains or reduced affordability ratio, or both.

Finally, as shown in Figures 9(g), 9(h), 10(g) and 10(h), while MILP and MBP do not explicitly optimize the affordability ratio, they almost always produce a pricing curve with highest affordability ratio. This is because, informally speaking, optimizing revenue can be achieved by selling models to as many as possible buyers, which implicitly optimizes the affordability ratio. Nonetheless, there are a few cases (such as Figure 10(h)), where MedC has slightly higher affordability ratio since it explicitly optimizes it.

7 CONCLUSION AND FUTURE WORK

In this work, we initiate the formal study of data markets that sell directly ML models to buyers. We propose a model-based pricing (MBP) framework, which instead of pricing

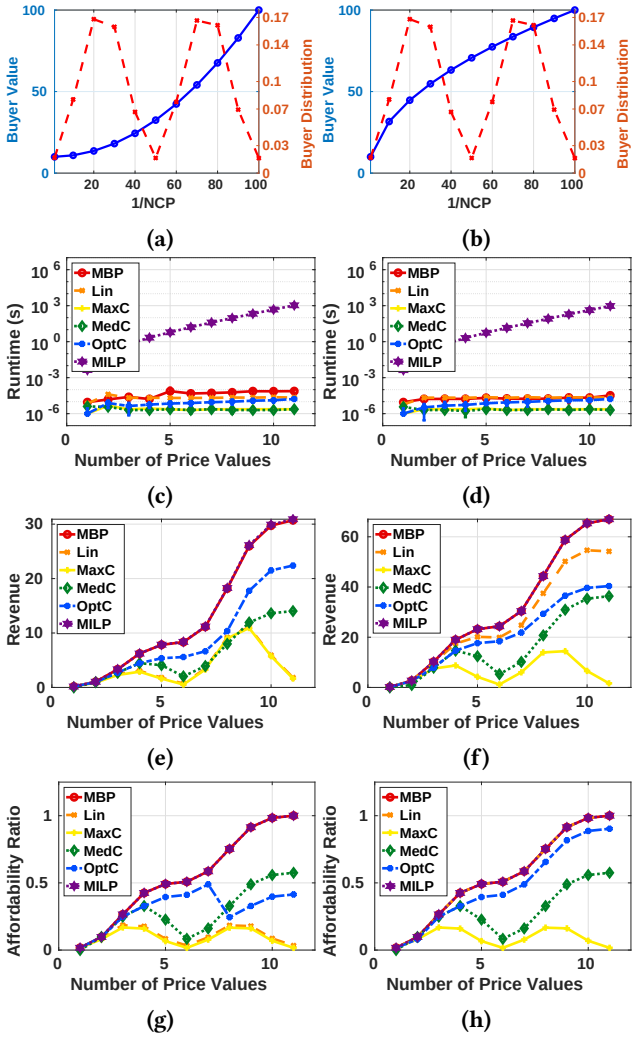

Figure 9: Runtime performance of MBP. We fix the buyer distribution and vary buyer valuation.

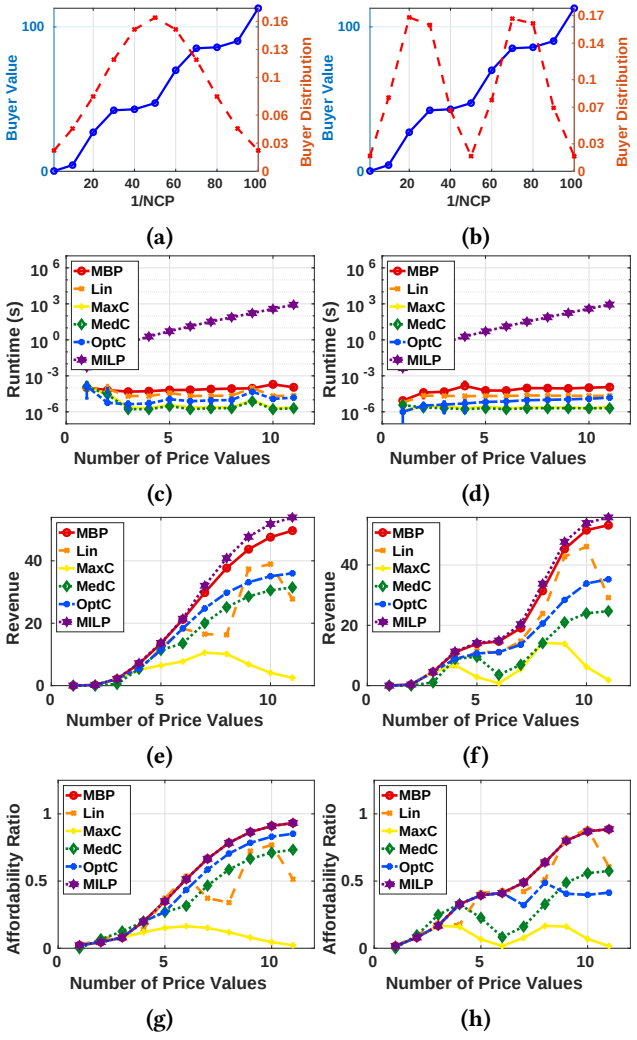

Figure 10: Runtime performance of MBP. We fix buyer value and vary buyer distribution.

the data, directly prices ML model instances. We show that a concrete realization of the MBP framework via a random noise injection approach provably satisfies several desired formal properties, including preventing arbitrage opportunities. Based on the proposed framework, we then provide algorithmic solutions on how sellers can assign prices to models under different market scenarios (such as to maximize revenue). Extensive experiments validate that the MBP framework can provide high revenue to the sellers and high affordability to the buyers with low runtime cost.

There are several other exciting directions for future work. First, more complex ML models such as Bayesian networks and artificial neural networks, are also frequently used. Non-relational data (images, text, etc.) might require complex feature extraction, possibly implicitly within an ML model (as in deep learning [24]). Second, we assumed that buyers know which ML model/hypothesis they want. This is a reasonable starting point because most existing cloud ML platforms assume the buyer picks the ML model. But in practice, users often perform *model selection* and explore different ML models [12, 33] and refine their choices iteratively [22]. Handling complex models and feature/model selections can be challenging, as it may require a revised arbitrage-freeness notion based on probabilistic guarantee and is left to future work. Third, in many cases data comes with privacy constraints, and therefore integrating model-based pricing with data privacy is also a core future challenge. Finally, more complicated buyer models as well as trade-offs between revenue and fairness can be further explored in the revenue optimization.

ACKNOWLEDGEMENT

This work is partially supported by the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation, a gift from Google and a Google PhD Fellowship. We thank Jeffrey Naughton and Xi Wu for invaluable discussions and feedback on earlier drafts of this paper.

REFERENCES

[1] [n. d.]. Big Data Exchange. www.bigdataexchange.com.
[2] [n. d.]. QLik Data Market. www.qlik.com/us/products/qlik-data-market.
[3] [n. d.]. Twitter GNIP Audience API. gnip.com/insights/audience.
[4] Martín Abadi et al. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR* abs/1603.04467 (2016).
[5] Magdalena Balazinska et al. 2011. Data Markets in the Cloud: An Opportunity for the Database Community. In *PVLDB*.
[6] Matthias Boehm et al. 2016. SystemML: Declarative Machine Learning on Spark. In *PVLDB*.
[7] Kamalika Chaudhuri and Claire Monteleoni. 2008. Privacy-preserving logistic regression. In *NIPS*.
[8] Yan Chen et al. 2018. Is my model any good: differentially private regression diagnostics. *Knowl. Inf. Syst.* (2018).
[9] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[10] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014).
[11] Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *KDD*.
[12] Jerome H. Friedman et al. 2001. *The Elements of Statistical Learning: Data mining, Inference, and Prediction.* Springer-Verlag.
[13] Joseph Geumlek, Shuang Song, and Kamalika Chaudhuri. 2017. Renyi Differential Privacy Mechanisms for Posterior Sampling. In *NIPS*.
[14] Mankiw Gregory. 2008. *Principles of Microeconomics* (5nd ed.).
[15] Michael Hay et al. 2010. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *PVLDB* (2010).
[16] Michael Hay, Liudmila Elagina, and Gerome Miklau. 2017. Differentially Private Rank Aggregation. In *ICDM*.
[17] Joseph M. Hellerstein et al. 2012. The MADlib Analytics Library or MAD Skills, the SQL. In *VLDB*.
[18] Michael Jordan. 2018. Artificial Intelligence: The Revolution Has Not Happened Yet. https://medium.com/@mijordan3/artificial-intelligence-the-revolution-hasnt-happened-yet-5e1d5812e1e7.
[19] Paraschos Koutris et al. 2012. Query-based data pricing. In *PODS*.
[20] Paraschos Koutris et al. 2012. QueryMarket Demonstration: Pricing for Online Data Markets. In *PVLDB*.
[21] Paraschos Koutris et al. 2013. Toward Practical Query Pricing with QueryMarket. In *SIGMOD*.
[22] Arun Kumar et al. 2015. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *SIGMOD Record* (2015).
[23] A. Besir Kurtulmus and Kenny Daniel. 2018. Trustless Machine Learning Contracts; Evaluating and Exchanging Machine Learning Models on the Ethereum Blockchain. *CoRR* abs/1802.10185 (2018).
[24] Yann LeCun et al. 2015. Deep Learning. *Nature* (2015).
[25] Chao Li and Gerome Miklau. 2012. Pricing Aggregate Queries in a Data Marketplace. In *WebDB*.
[26] Xupeng Li et al. 2017. MLog: Towards Declarative In-Database Machine Learning. In *PVLDB*.
[27] Yaping Li et al. 2011. Enabling Multi-level Trust in Privacy Preserving Data Mining. *CoRR* abs/1104.0459 (2011).
[28] Katrina Ligett et al. 2017. Accuracy First: Selecting a Differential Privacy Level for Accuracy Constrained ERM. In *NIPS*.
[29] Bing-Rong Lin and Daniel Kifer. 2014. On Arbitrage-free Pricing for General Data Queries. In *PVLDB*.
[30] Yucheng Low et al. 2010. GraphLab: A New Framework For Parallel Machine Learning. In *UAI*.
[31] Ashwin Machanavajjhala et al. 2017. Differential Privacy in the Wild: A Tutorial on Current Practices & Open Challenges. In *SIGMOD*.
[32] Ryan McKenna et al. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *PVLDB* (2018).
[33] Tom M. Mitchell. 1997. *Machine Learning*. McGraw Hill.
[34] Alan Nash et al. 2007. Determinacy and Rewriting of Conjunctive Queries Using Views: A Progress Report. In *ICDT*.
[35] Alan Nash et al. 2010. Views and Queries: Determinacy and Rewriting. *ACM Trans. Database Syst.* (2010).
[36] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191
[37] Nicolas Papernot et al. 2016. Towards the Science of Security and Privacy in Machine Learning. *CoRR* (2016).
[38] Neoklis Polyzotis et al. 2017. Data Management Challenges in Production Machine Learning. In *SIGMOD*.
[39] Aaron Roth. 2017. Pricing Information (and its Implications): Technical Perspective. *Commun. ACM* (2017).
[40] Anand D. Sarwate and Kamalika Chaudhuri. 2013. Signal Processing and Machine Learning with Differential Privacy: Algorithms and Challenges for Continuous Data. *IEEE Signal Process. Mag.* (2013).
[41] Carl Shapiro and Hal R. Varian. 1998. Versioning: The Smart Way to Sell Information. *Harvard Business Review* (1998).
[42] Xiaokui Xiao, Yufei Tao, and Minghua Chen. 2009. Optimal Random Perturbation at Multiple Privacy Levels. *PVLDB* (2009).
[43] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2011. Differential Privacy via Wavelet Transforms. *IEEE TKDE* (2011).
[44] Ce Zhang et al. 2014. Materialization Optimizations for Feature Selection Workloads. In *SIGMOD*.

A MISSING PROOFS

Proof of Lemma 1 . Suppose that $p_{\epsilon,\lambda}$ is not error-monotone in D. This implies that there exist parameters δ_1, δ_2 such that $\mathbb{E}\left[\epsilon(\hat{h}_\lambda^{\delta_1}(D), D)\right] \leq \mathbb{E}\left[\epsilon(\hat{h}_\lambda^{\delta_2}(D), D)\right]$ and $p_{\epsilon,\lambda}(\delta_1, D) < p_{\epsilon,\lambda}(\delta_2, D)$. It is easy to see that in this case $p_{\epsilon,\lambda}$ exhibits 1-arbitrage, since we can simply pick the function g to be the identity function. Thus, $p_{\epsilon,\lambda}$ cannot be arbitrage-free. □

Proof of Lemma 2. Indeed, we have $\mathbb{E}\left[\mathcal{K}_G(h_\lambda^*(D), w)\right] = \mathbb{E}\left[h_\lambda^*(D) + w\right] = h_\lambda^*(D) + \mathbb{E}[w] = h_\lambda^*(D)$, where the last equality comes from the fact that the Gaussian noise we add has mean 0 in every dimension. □

Proof of Lemma 3. $\mathbb{E}\left[\epsilon_s\left(\hat{h}_\lambda^\delta(D), D\right)\right] = \mathbb{E}\left\|\hat{h}_\lambda^\delta(D) - h_\lambda^*(D)\right\|_2^2$ $= \mathbb{E}\left[\|w\|_2^2\right] = \sum_{i=1}^p \mathbb{E}\left[w_i^2\right] = \delta$ finishes the proof. □

Proof of Theorem 4. We will use the following property of a strictly convex function ϵ. For every $x, y \in \mathbb{R}^p$,

$$\epsilon(x + \sigma_1 y) + \epsilon(x - \sigma_1 y) > \epsilon(x + \sigma_2 y) + \epsilon(x - \sigma_2 y)$$

if and only if $\sigma_1 > \sigma_2$, where σ_1, σ_2 are scalars. To prove the above property, we apply the definition of strict convexity twice with $t = \frac{1}{2}(1 + \frac{\sigma_2}{\sigma_1})$ when $\sigma_1 > \sigma_2$

$$t\epsilon(x + \sigma_1 y) + (1-t)\epsilon(x - \sigma_1 y) > \epsilon(x + \sigma_2 y)$$
$$(1-t)\epsilon(x + \sigma_1 y) + t\epsilon(x - \sigma_1 y) > \epsilon(x - \sigma_2 y)$$

and then sum up the two inequalities. The "if" part is a result of symmetry. Let $h^* = h^*_\lambda(D)$, and w_1, w_2 the two Gaussian noise vectors. We can now compute the expectations by $\mathbb{E}\left[\epsilon(\hat{h}^{\delta_1}_\lambda(D), D)\right] - \mathbb{E}\left[\epsilon(\hat{h}^{\delta_2}_\lambda(D), D)\right] = \mathbb{E}[\epsilon(h^* + w_1)] - \mathbb{E}[\epsilon(h^* + w_2)] = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}}(\epsilon(h^* + \delta_1 y) - \epsilon(h^* + \delta_2 y))e^{-\frac{1}{2}y^2}dy = \int_0^{+\infty} \frac{1}{\sqrt{2\pi}}(\epsilon(h^* + \delta_1 y) + \epsilon(h^* - \delta_1 y) - \epsilon(h^* + \delta_2 y) - \epsilon(h^* - \delta_2 y))e^{-\frac{1}{2}y^2}dy$, where the last equality comes from splitting the interval $[-\infty, +\infty]$ to two smaller intervals, $[-\infty, 0]$ and $[0, +\infty]$ and then changing the sign of y in the first term. It is now easy to see that the above quantity is strictly positive if and only if $\delta_1 > \delta_2$. □

Proof of Theorem 5. We next prove the two directions of the theorem.

($\Longrightarrow$) Suppose that the pricing function $p_{\epsilon_s,\lambda}$ is arbitrage free. Then, by Lemma 1 the pricing function is also error-monotone in D, so condition (2) holds. To show that condition (1) holds as well, consider parameters $\delta_1, \delta_2, \delta_3$ such that $1/\delta_1 = 1/\delta_2 + 1/\delta_3$ and

$$p_{\epsilon_s,\lambda}(\delta_1, D) > p_{\epsilon_s,\lambda}(\delta_2, D) + p_{\epsilon_s,\lambda}(\delta_3, D).$$

We will show in this case that the pricing function violates k-arbitrage for $k = 2$. We define the following function g that combines two models: $g(\hat{h}^{\delta_2}_\lambda(D), \hat{h}^{\delta_3}_\lambda(D)) = \frac{\delta_1}{\delta_2} \cdot \hat{h}^{\delta_2}_\lambda(D) + \frac{\delta_1}{\delta_3} \cdot \hat{h}^{\delta_3}_\lambda(D)$. Now, observe that: $\tilde{h} = g(\hat{h}^{\delta_2}_\lambda(D), \hat{h}^{\delta_3}_\lambda(D)) = \frac{\delta_1}{\delta_2} \cdot (h^*_\lambda(D) + w_2) + \frac{\delta_1}{\delta_3} \cdot (h^*_\lambda(D) + w_3) = h^*_\lambda(D) + \frac{\delta_1}{\delta_2} \cdot w_2 + \frac{\delta_1}{\delta_3} \cdot w_3$. Hence, we can compute the expectation $\mathbb{E}\left[\epsilon_s(\tilde{h}, D)\right] = \frac{\delta_1^2}{\delta_2^2} \cdot \mathbb{E}\left[w_2^2\right] + \frac{\delta_1^2}{\delta_3^2} \cdot \mathbb{E}\left[w_3^2\right] = \delta_1^2\left(\frac{1}{\delta_2} + \frac{1}{\delta_3}\right) = \delta_1 = \mathbb{E}\left[\epsilon_s(\hat{h}^{\delta_1}_\lambda(D), D)\right]$, and therefore, the pricing function indeed violates 2-arbitrage, a contradiction.

($\Longleftarrow$) We now show the opposite direction of the theorem, *i.e.*, that conditions (1) and (2) imply arbitrage freeness. To show this, we will use the Cramér-Rao inequality, which provides a lower bound on the variance of an unbiased estimator of a deterministic parameter. To apply the inequality, notice that the function g in the definition of k-arbitrage is essentially an estimator of the optimal model $h^*_\lambda(D)$. Hence, for any function g and $\tilde{h} = g(\hat{h}^{\delta_1}_\lambda(D), \hat{h}^{\delta_2}_\lambda(D), \ldots, \hat{h}^{\delta_k}_\lambda(D))$, we have:

$$\mathbb{E}\left[\epsilon_s(\tilde{h}, D)\right] \geq \frac{1}{\sum_{j=1}^k \frac{1}{\delta_j}} \tag{5}$$

Suppose that the pricing function $p_{\epsilon_s,\lambda}$ exhibits 1-arbitrage. Then, there must exist parameters δ_1, δ_2 with $p_{\epsilon_s,\lambda}(\delta_1, D) < p_{\epsilon_s,\lambda}(\delta_2, D)$, and a function g that returns a model $\tilde{h} = g(\hat{h}^{\delta_1}_\lambda(D))$ with $\mathbb{E}\left[\epsilon_s(\tilde{h}, D)\right] \leq \mathbb{E}\left[\epsilon_s(\hat{h}^{\delta_2}_\lambda(D), D)\right] = \delta_2$. However, Eq. (5) implies that $\mathbb{E}\left[\epsilon_s(\tilde{h}, D)\right] \geq \delta_1$. Thus, we obtain $\delta_1 \leq \delta_2$, which makes condition (2) false. Next, suppose that the pricing function exhibits k-arbitrage for $k \geq 2$. Using the same argument as above, we can show that there exist parameters $\delta_0, \delta_1, \ldots, \delta_k$ such that: (i) $1/\delta_0 = \sum_{j=1}^k 1/\delta_j$; and (ii) $\sum_{j=1}^k p_{\epsilon_s,\lambda}(\delta_j, D) < p_{\epsilon_s,\lambda}(\delta_0, D)$. We will show that the above 2 properties imply that condition (1) is false by contradiction. Assume that condition (1) is true, and also that $1/\delta_0 = \sum_{j=1}^k 1/\delta_j$. For $j = 1, \ldots, k-1$, let us define Δ_j such that $1/\Delta_j = \sum_{c=j+1}^k 1/\delta_c$. Observe that $1/\delta_0 = 1/\delta_1 + 1/\Delta_1$, and also $1/\Delta_j = 1/\delta_{j+1} + 1/\Delta_{j+1}$. Then, we can write: $p_{\epsilon_s,\lambda}(\delta_0, D) \leq p_{\epsilon_s,\lambda}(\delta_1, D) + p_{\epsilon_s,\lambda}(\Delta_1, D) \leq p_{\epsilon_s,\lambda}(\delta_1, D) + p_{\epsilon_s,\lambda}(\delta_2, D) + p_{\epsilon_s,\lambda}(\Delta_2, D) \leq \cdots \leq \sum_{j=1}^k p_{\epsilon_s,\lambda}(\delta_j, D)$. This contradicts the second property, and hence condition (1) must be false. □

Proof of Theorem 7. We will prove the theorem by showing a reduction from the unbounded subset-sum problem. In this problem, we are given as input a set of positive integers $\{w_1, w_2, \ldots, w_n\}$, and a positive number K. We then want to decide whether there exist non-negative integers k_i such that $\sum_{i=1}^n k_i w_i = K$. In other words, we are asking whether we can achieve sum K using each w_i zero or more times. It is known that unbounded subset-sum is $\mathcal{NP}$-hard. Consider an instance of the unbounded subset-sum problem, with positive integers $\{w_1, w_2, \ldots, w_n\}$, and a positive number K. Without any loss of generality, suppose that $w_1 < w_2 < \cdots < w_n < K$. We now construct an instance for price interpolation as follows: let $P_j = a_j = w_j$ for $j = 1, \ldots, n$, and $a_{n+1} = K$, $P_{n+1} = K + 1/2$. We will prove that there exists a subadditive and monotone function that interpolates the points (a_j, P_j) if and only if there exists no (unbounded) subset sum with value K.

$\Rightarrow$ For the first direction, suppose that there exists an unbounded subset sum with value K. In other words, there exist positive integers k_j such that $\sum_{j=1}^n k_j w_j = K$. For the sake of contradiction, suppose that we can interpolate a subadditive and monotone function $\hat{p}$. Then we have: $K + 1/2 = P_{n+1} = \hat{p}(K) = \hat{p}\left(\sum_{j=1}^n k_j w_j\right) \leq \sum_{j=1}^n k_j \hat{p}(w_j) = \sum_{j=1}^n k_j w_j = K$ which is a contradiction.

$\Leftarrow$ For the reverse direction, suppose that there exists no unbounded subset sum K; we will show that we can construct a subadditive and monotone function f that interpolates the $(n+1)$ points. For every $x \geq 0$, define $\mu(x)$ to be the smallest possible unbounded subset sum that is at least x. Notice that $\mu(x) \geq x$ for every $x \geq 0$. Then, we define

$f(x) = \min\{\mu(x), K + 1/2\}$. It is straightforward to see that $f(x)$ is monotone by construction. We next show that f interpolates the points. Indeed, for $j = 1, \ldots, n$, we have that $\mu(a_j) = a_j < K + 1/2$ (since a_j by itself gives a sum of a_j), and hence $f(a_j) = a_j$. For $j = n + 1$, observe that by our starting assumption there is no sum of K, and hence $\mu(a_{n+1}) \geq K + 1$, which implies that $f(a_{n+1}) = K + 1/2$. Finally, we show that f is a subadditive function. Let $x, y \geq 0$. If $\mu(x) \geq K+1$ then, $f(x)+f(y) \geq f(x) = K+1/2 \geq f(x+y)$. A symmetric argument holds if $\mu(y) \geq K + 1$. Now, suppose that $\mu(x), \mu(y) \leq K$. Then, there exists k_j, k'_j such that $f(x) = \sum_{j=1}^{n} k_j w_j$ and $f(y) = \sum_{j=1}^{n} k'_j w_j$. Now we have: $x + y \leq f(x) + f(y) = \sum_{j=1}^{n} k_j w_j + \sum_{j=1}^{n} k'_j w_j = \sum_{j=1}^{n} (k_j + k'_j) w_j$ Hence, if we pick $k''_j = k_j + k'_j$, we obtain a subset sum that is at least $x + y$. We can then write: $f(x + y) \leq \mu(x + y) \leq \sum_{j=1}^{n} (k_j + k'_j) w_j = f(x) + f(y)$. This concludes our proof. □

Proof of Corollary 7.1. Observe that the objective functions $T^{\infty}_{\text{PI}}, T^{2}_{\text{PI}}$ are maximized if and only if $x_j = P_j$ for $j = 1, \ldots, n$. Hence, we can reduce SUBADDITIVE INTERPOLATION to (2). Similarly, the objective functions T_{BV} is maximized at the unique point where $x_j = v_j$, *i.e.* the price at a_j equals the valuation v_j. Hence, we can reduce SUBADDITIVE INTERPOLATION to the revenue maximization with buyer valuations problem by setting $v_j = P_j$ and $b_j = 1$ for every $j = 1, \ldots, n$. □

Proof of Lemma 8. Let $\hat{q}$ be a feasible solution to (3). We will show that $\hat{q}$ satisfies the subadditivity condition as well. Let $x, y > 0$. Since $\hat{q}$ satisfies the constraints in (3), we have $\frac{\hat{q}(x)}{x} \geq \frac{\hat{q}(x+y)}{x+y}$ and $\frac{\hat{q}(y)}{y} \geq \frac{\hat{q}(x+y)}{x+y}$. Thus: $\hat{q}(x) + \hat{q}(y) \geq \frac{x}{x+y}\hat{q}(x + y) + \frac{y}{x+y}\hat{q}(x + y) = \hat{q}(x + y)$. which concludes the proof. □

Proof of Lemma 9. Let $\hat{p}$ be a feasible solution of (2). We construct a function $\hat{q}$ such that for every $x > 0$: $\hat{q}(x) = x \cdot \min_{0<y\leq x}\{\hat{p}(y)/y\}$. We first show that $\hat{q}$ is a feasible solution of (3). Note that $\hat{q}$ is positive. Consider $0 < x \leq x'$. Then we have: $\hat{q}(x)/x = \min_{0<y\leq x}\{\hat{p}(y)/y\} \geq \min_{0<y\leq x'}\{\hat{p}(y)/y\} = \hat{q}(x')/x'$. To show that $\hat{q}(x) \leq \hat{q}(x')$, let $y_m = \text{argmin}_{0<y\leq x'}\{\hat{p}(y)/y\}$. If $y_m \leq x$, we have $\min_{0<y\leq x}\{\hat{p}(y)/y\} = \min_{0<y\leq x'}\{\hat{p}(y)/y\}$, and the desired result comes from $x \leq x'$. Otherwise, if $y_m > x$, we have: $\hat{q}(x) = x \cdot \min_{0<y\leq x}\{\hat{p}(y)/y\} \leq \hat{p}(x) \leq \hat{p}(y_m) = y_m\{\hat{p}(y_m)/y_m\} \leq x' \min_{0<y\leq x'}\{\hat{p}(y)/y\} = \hat{q}(x')$. Finally, we show that $\hat{p}(x)/2 \leq \hat{q}(x) \leq \hat{p}(x)$ for every $x > 0$. We have already shown that $\hat{q}(x) \leq \hat{p}(x)$. For the first inequality, let as before $y_m = \text{argmin}_{0<y\leq x}\{\hat{p}(y)/y\}$ and define $\Delta = x/y_m \geq 1$. If $\Delta = 1$, then $\hat{q}(x) = \hat{p}(x)$, so the result holds trivially. So, assume that $\Delta > 1$. The key observation is that $\hat{p}(x) = \hat{p}(y_m\Delta) \leq \hat{p}(y_m\lceil\Delta\rceil) \leq \lceil\Delta\rceil\hat{p}(y_m)$ where the second inequality holds from the subadditivity constraint for $\hat{p}$. Thus we have: $\hat{q}(x) = x\{\hat{p}(y_m)/y_m\} \geq \frac{\Delta}{\lceil\Delta\rceil}\hat{p}(x) \geq \frac{\Delta}{\Delta+1}\hat{p}(x) > \hat{p}(x)/2$ where the last inequality follows from the fact that $\Delta > 1$. This concludes the proof. □

Proof of Proposition 1. Let $\hat{p}$ be a feasible solution of (3) of objective value M. It is immediate that $x_i = \hat{p}(a_i)$ is a feasible solution for (4) with the same objective value.

For the opposite direction, suppose that $\mathbf{x}$ is a feasible solution to problem (4) with value M. Without any loss of generality, assume that $a_1 \leq a_2 \leq \cdots \leq a_n$. Let us define $\hat{p}$ to be a piecewise linear function such that: $\hat{p}(x) = \begin{cases} \frac{x_{j+1}}{a_{j+1}}x, & x \in [0, a_1] \\ x_j + \frac{x_{j+1}-x_j}{a_{j+1}-a_j}(x - a_j), & x \in [a_j, a_{j+1}] \\ x_n, & x \in [a_n, \infty) \end{cases}$. It is easy to see that $\hat{p}$ is non-negative, and that for every $i = 1, \ldots, n$ we have $\hat{p}(a_i) = x_i$. Additionally, $\hat{p}$ is monotone, since it is a piecewise linear function where $x_1 \leq x_2 \leq \cdots \leq x_n$. Finally, we show that for any $y \geq x > 0$ we have $\hat{p}(y)/y \leq \hat{p}(x)/x$.

First, assume that x, y are in the same interval $[a, b]$ of the piecewise linear function (which takes values x_a, x_b). Since in this interval we have $\hat{p}(x) = x_a + \frac{x_b-x_a}{b-a}(x - a)$, we want to equivalently show that:

$$\frac{x_a}{x} + \frac{x_b - x_a}{b - a}\left(1 - \frac{a}{x}\right) \geq \frac{x_a}{y} + \frac{x_b - x_a}{b - a}\left(1 - \frac{a}{y}\right) \quad \Leftrightarrow$$

$$\frac{bx_a - ax_b}{x} \geq \frac{bx_a - ax_b}{y} \quad \Leftrightarrow \quad (y - x)(bx_a - ax_b) \geq 0$$

The last inequality holds because $y \geq x$, and also $x_a/a \geq x_b/b$ (which follows from the constraints).

Now, if x, y are not in the same interval, assume that x falls in the i-th interval $[a_i, a_{i+1}]$, and y in the j-th interval $[a_j, a_{j+1}]$, where $j \geq i$. Then we have: $\hat{p}(x)/x \geq \hat{p}(a_{i+1})/a_{i+1} \geq \hat{p}(a_{i+2})/a_{i+2} \geq \cdots \geq \hat{p}(a_j)/a_j \geq \hat{p}(y)/y$. The first inequality comes from the fact that x, a_{i+1} are in the same interval, the last from the fact that y, a_j are in the same interval, and all the intermediate inequalities from the constraints in (4). □

Proof of Proposition 2. Let $\hat{p}^*$ denote the optimal solution of (2) with optimal value C_{SA}, and $\mathbf{x}^*$ the optimal solution of (4) with optimal value C_{MBP}. From Proposition 3, there exists a solution $\hat{q}^*$ of (3) that achieves the same value C_{MBP}. From Lemma 8, we obtain that $\hat{q}^*$ is also a solution to (2), and hence it must be that $C_{MBP} \leq C_{SA}$. Additionally, Lemma 9 tells us that there exists $\tilde{q}$ that is a feasible solution of (3) such that for every $x > 0$, $\hat{p}^*(x)/2 \leq \tilde{q}(x) \leq \hat{p}^*(x)$. If C' is the objective value for $\tilde{q}$, we then have that $C' \leq C_{MBP}$. We next show that $C' \geq C_{SA} + \sum_j T_j(0)/2$. We first claim that for every x, i, we have that $T_i(\tilde{q}(x)) \geq \min\{T_i(\hat{p}^*(x), T_i(\hat{p}^*(x)/2)\}$. Indeed, suppose that this is not true. Then, since $\hat{p}^*(x)/2 \leq \tilde{q}(x) \leq \hat{p}^*(x)$, there exists $\lambda \in [0, 1]$ such that $\tilde{q}(x) = \lambda\hat{p}^*(x)/2 + (1 - \lambda)\hat{p}^*(x)$. By the concavity of T_i, we now have $T_i(\tilde{q}(x)) = T_i(\lambda\hat{p}^*(x)/2 + (1 - \lambda)\hat{p}^*(x)) \geq \lambda T_i(\hat{p}^*(x)/2) + (1 - \lambda)T_i(\hat{p}^*(x)) > \lambda T_i(\tilde{q}(x)) +$

$(1-\lambda)T_i(\tilde{q}(x)) = T_i(\tilde{q}(x))$ which is a contradiction. Next, we bound $T_i(\hat{p}^*(x)/2)$ as follows using concavity: $T_i(\hat{p}^*(x)/2) = T_i\left(\hat{p}^*(x)/2 + 0/2\right) \geq T_i(\hat{p}^*(x))/2 + T_i(0)/2$. So we have:

$$\begin{aligned} T_i(\tilde{q}(x)) &\geq \min\{T_i(\hat{p}^*(x), T_i(\hat{p}^*(x))/2 + T_i(0)/2\} \\ &\geq \min\{T_i(\hat{p}^*(x), T_i(\hat{p}^*(x)) + T_i(0)/2\} \\ &= T_i(\hat{p}^*(x)) + T_i(0)/2 \end{aligned}$$

where the last inequality follows from the fact that T_i is non-positive. Finally, by $C' = \sum_{j=1}^n T_j(\tilde{q}(a_j)) \geq \sum_{j=1}^n T_j(\hat{p}^*(a_j)) + \sum_{j=1}^n T_j(0)/2 = C_{SA} + \sum_{j=1}^n T_j(0)/2$, we finish the proof. □

Proof of Proposition 3. Let $\hat{p}^*$ denote the optimal solution of (2) with optimal value C_{SA}, and $\mathbf{x}^*$ the optimal solution of (4) with optimal value C_{MBP}. From Proposition 3, there exists a solution $\hat{q}^*$ of (3) that achieves the same value C_{MBP}. From Lemma 8, $\hat{q}^*$ is also a solution to (2), and hence $C_{MBP} \leq C_{SA}$. Additionally, Lemma 9 implies that $\exists \tilde{q}$ that is a feasible solution of (3) such that for every $x > 0$, $\hat{p}^*(x)/2 \leq \tilde{q}(x) \leq \hat{p}^*(x)$. If C' is the objective value for $\tilde{q}$, we then have that $C' \leq C_{MBP}$. We next show that $C' \geq C_{SA}/2$. First, notice that $\tilde{q}(x) \leq \hat{p}^*(x)$ implies that for every j: $\mathbf{1}_{\tilde{q}(a_j)\leq v_j} \geq \mathbf{1}_{\hat{p}^*(a_j)\leq v_j}$. Now we can write: $C' = g_{\text{BV}}(\tilde{q}(a_1), \ldots, \tilde{q}(a_n)) = \sum_{j=1}^n b_j \tilde{q}(a_j) \cdot \mathbf{1}_{\tilde{q}(a_j)\leq v_j} \geq \sum_{j=1}^n b_j \tilde{q}(a_j) \cdot \mathbf{1}_{\hat{p}^*(a_j)\leq v_j} \geq \frac{1}{2}\sum_{j=1}^n b_j \hat{p}^*(a_j) \cdot \mathbf{1}_{\hat{p}^*(a_j)\leq v_j} = C_{SA}/2$ where the last inequality comes from the fact that $\hat{p}^*(x)/2 \leq \tilde{q}(x)$. This concludes the proof. □

Proof of Theorem 10. We prove this theorem by constructing an algorithm based on dynamic programming that optimally solves (4) with objective function T_{BV}.

Suppose that we are given as input the parameters v_j, b_j that correspond to point a_j for $j = 1, \ldots, n$. Assume that $a_1 \leq a_2 \leq \cdots \leq a_n$ and $v_1 \leq v_2 \leq \cdots \leq v_n$ (*i.e.*, the valuations of the buyers are monotone w.r.t. the error). Let $s(k, \Delta)$ denote the optimum solution for the subproblem with points $j = k, \ldots, n$, with the restriction that for every $j \geq k$ we have $s_j(k, \Delta)/a_j \leq \Delta$. Denote by $OPT(k, \Delta)$ the objective value of this optimum solution. Observe that the optimum solution for the initial problem is $s(1, +\infty)$, with optimum value $OPT(1, +\infty)$. We will provide a recursive formula to compute $OPT(k, \Delta)$ for any k, Δ. Our first observation is that for $k = n$, we can easily compute the optimum solution via $s_n(n, \Delta) = \min\{v_n, \Delta a_n\}, OPT(n, \Delta) = b_n \cdot s_n(n, \Delta)$. This follows from that it is always more profitable to assign a higher price, as long as it is under the valuation v_n. To compute the recursive formula for $s(k, \Delta)$, we need a few more lemmas.

Lemma 11. *For every k, $s_k(k, \Delta) \geq \min\{v_k, \Delta a_k\}$.*

Proof. Suppose not; we will then show that we can obtain a solution with a larger objective value. Let $\ell \geq k$ be the largest index such that $s_\ell(k, \Delta) = s_k(k, \Delta)$. Clearly we have that $s_k(k, \Delta) = s_{k+1}(k, \Delta) = \ldots, s_\ell(k, \Delta)$. Since $s_k(k, \Delta) < v_k$ and $v_k \leq v_{k+1} \leq \cdots \leq v_\ell$, we must have that $s_j(k, \Delta) < v_j$ for every $j = k, \ldots, \ell$. Similarly, since $s_k(k, \Delta) < \Delta a_k$ and $a_k \leq a_{k+1} \leq \cdots \leq a_\ell$, we must have that $s_j(k, \Delta) < \Delta a_j$ for every $j = k, \ldots, \ell$. Let $\epsilon = \min_{j=k}^{\ell} \frac{\min\{v_k, \Delta a_k\}}{s_j(k,\Delta)} > 1$. Define $s'(k, \Delta)$ such that for $j = k, \ldots, \ell$ we have $s'_j(k, \Delta) = \epsilon \cdot s(k, \Delta)$, and for $j > \ell$ it remains the same. It is easy to see that the resulting solution is feasible, and also produces a strictly greater revenue, a contradiction. □

Lemma 12. *Let $k < n$. If $a_k \Delta \leq v_k$, then:*

$$s_k(k, \Delta) = \Delta a_k,\ s_j(k, \Delta) = s_j(k+1, \Delta), j > k$$
$$OPT(k, \Delta) = b_k \Delta a_k + OPT(k+1, \Delta)$$

Proof. From Lemma 11, we have that $s_k(k, \Delta) \geq \Delta a_k$. But it must also be that $s_k(k, \Delta) \leq \Delta a_k$, thus the only optimal solution is $s_k(k, \Delta) = \Delta a_k$. Also, since $(\Delta a_k)/a_k = \Delta \geq s_j(k+1, \Delta)/a_{k+1}$, the weakened subadditive constraint is satisfied. Finally, we have $s_{k+1}(k+1, \Delta) \geq \min\{\Delta a_{k+1}, v_{k+1}\} \geq \min\{\Delta a_k, v_k\} = \Delta a_k$, which implies monotonicity. □

Lemma 13. *Let $k < n$. If $a_k \Delta > v_k$, define*

$$s'_k(k, \Delta) = v_k,\ s'_j(k, \Delta) = s_j(k+1, v_k/a_k), j > k$$
$$s''_k(k, \Delta) = s_{k+1}(k+1, \Delta)\frac{a_k}{a_{k+1}},\ s''_j(k, \Delta) = s_j(k+1, \Delta), j > k$$

with optimum values respectively

$$OPT'(k, \Delta) = b_k v_k + OPT(k+1, v_k/a_k)$$
$$OPT''(k, \Delta) = OPT(k+1, \Delta)$$

Then, $OPT(k, \Delta)$ is the maximum between the two options, and $s(k, \Delta)$ is the solution that achieves the maximum.

Proof. From Lemma 11, we have $s_k(k, \Delta) \geq v_k$. The first option examines what will happen if we set $s_k(k, \Delta) = v_k$, where we will obtain a profit of $b_k v_k$ from this price point. If $s_k(k, \Delta) > v_k$, then we get 0 revenue from this point. Also, the higher the price, the more revenue is extracted from the remaining points until we reach Δa_k. It is straightforward to see that the weakened subadditive constraint is satisfied in both cases. We now show the same for monotonicity as well. For the first option, since $v_k \leq v_{k+1}$ and $v_k = (v_k/a_k)a_k \leq (v_k/a_k)a_{k+1}$, we have $v_k \leq \min\{v_{k+1}, (v_k/a_k)a_{k+1}\} \leq s_{k+1}(k+1, v_k/a_k)$. For the second option, monotonicity follows from that $a_k \leq a_{k+1}$. □

We can now use the recursive formulas from the two lemmas above to obtain an efficient dynamic programming algorithm. The key observation is that we only need to consider $(n+1)$ values of Δ, since from the recurrence relations, Δ can only take values from the set $\{v_1/a_1, v_2/a_2, \ldots, v_n/a_n, +\infty\}$. The dynamic programming algorithm will first compute $s(n, \Delta)$, for the $(n+1)$ values of Δ, and then iteratively compute $s(k, \Delta)$ for $k = n-1, n-2, \ldots, 1$. The final solution will be $s(1, +\infty)$. Since we have n iterations, each of which computes $(n+1)$ subproblems, the total runtime is $O(n^2)$. □