Model-based Pricing: Do Not Pay for More than What You Learn!

Lingjiao Chen
University of Wisconsin-Madison
lchen362@wisc.edu

Paraschos Koutris
University of Wisconsin-Madison
paris@cs.wisc.edu

Arun Kumar
University of California San Diego
arunkk@eng.ucsd.edu

ABSTRACT

While a lot of work has focused on improving the efficiency, scalability, and usability of machine learning (ML), little work has studied the *cost of data acquisition* for ML-based analytics. Datasets are already being bought and sold in marketplaces for various tasks, including ML. But current marketplaces force users to buy such data in whole or as fixed subsets without any awareness of the ML tasks they are used for. This leads to sub-optimal choices and missed opportunities for both data sellers and buyers. In this paper, we outline our vision for a formal and practical pricing framework we call *model-based pricing* that aims to resolve such issues. Our key observation is that ML users typically need only as much data as needed to meet their accuracy goals, which leads to novel trade-offs between price, accuracy, and runtimes. We explain how this raises interesting new research questions at the intersection of data management, ML, and micro-economics.

ACM Reference format:
Lingjiao Chen, Paraschos Koutris, and Arun Kumar. 2017. Model-based Pricing: Do Not Pay for More than What You Learn!. In *Proceedings of ACM SIGMOD Workshop on Data Management for End-to-end Machine Learning, Chicago, Illinois, USA, May 2017 (DEEM'17),* 4 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Analytics using machine learning (ML) is now an integral part of science, business intelligence, journalism, and many other domains. Research and industrial efforts have largely focused on how to integrate ML with data systems (e.g., MADlib [8]), how to improve performance or scalability (e.g., GraphLab [18]), cloud ML services (e.g., Microsoft AzureML), or applying database ideas to ML tasks (e.g., Columbus [23]). However, limited research has studied the *cost of acquiring data* for ML-based analytics: users often *buy* data from one or more *data sellers* to train their ML models.

Many companies, including Bloomberg, Twitter, and Lattice Data *sell* rich, marked-up, structured (*relational*) datasets. But such datasets are often very expensive due to the immense effort that went into collecting, integrating, and cleaning them. Today, such datasets are often sold through *data markets*, possibly in the *cloud*, e.g., Azure Marketplace [2] or BDEX [1]. But existing data markets either force users to buy the *whole* dataset or support very simplistic *pricing mechanisms* such as pricing fixed subsets [9] without any awareness of ML tasks. This means that valuable datasets are often not accessible to many users with limited budgets. In turn, this leaves a large market of potential buyers untapped for sellers and marketplaces. As ML-based analytics grow in popularity, *it is a pressing challenge to devise more flexible, ML-aware pricing frameworks that could democratize access to valuable datasets.*

In this paper, we describe our vision for a formal and practical fine-grained pricing framework for ML over relational data in the cloud. Our key observation is that a sample of the available data is often enough to train an ML model to achieve an *accuracy* desired by the user; *the price then should depend on the size of sample used and not the full dataset.* Since the price is based on the model *instance* returned after training over the samples, we call our framework *model-based pricing* (MBP). We start with two motivating examples.

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 demographic information exist online, but they are expensive and exceed Alice's available budget. In this scenario, a system or a data marketplace with MBP would allow Alice to be charged only based on her ML task and desired accuracy. In particular, a subset of the examples and perhaps, an appropriate subset of the features might be sufficient for Alice's purposes.

Example 2. Companies like Twitter, IBM, and Lattice Data process text, images, and other so-called "dark data" sources to build structured knowledge bases, which they then sell. For example, Twitter's GNIP API [3] enables users to pay for aggregate data about an *audience* (a set of Twitter users with the same age, location, and/or gender). Such datasets are bought by advertising companies, online retailers, business analysts, etc., who use it to build ML models about user behavior to develop better market strategies, detect fraud, etc. MBP could offer more flexibility for such buyers, potentially increasing the market size.

Our Contributions. We propose a novel framework, called model-based pricing, to price ML models learned over relational data. Model-based pricing involves three agents: *seller*, *broker*, and *buyer*. The seller provides the dataset that they wish to sell, the buyer specifies the ML model they wish to learn along with specific criteria (e.g., accuracy, budget), and the broker releases an ML *model instance* trained on (subsamples of) the dataset based on the criteria specified by the buyer. The basic premise of MBP is that there is a fundamental *accuracy-price trade-off* in ML analytics: *buyers are typically willing to pay more for a model instance with higher accuracy.* Conversely, *buyers are typically willing to tolerate a lower accuracy for a lower price.* Thus, a key desideratum is that an ML model instance's price has to be significantly lower than the price of the dataset (subsample) it was trained on; otherwise, a buyer can simply purchase the dataset and train a model themselves.

DEEM'17, Chicago, Illinois, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND AND RELATED WORK

Pricing Relational Queries. The problem of pricing relational data has received a lot of attention recently. This has been driven by the growing number of data brokers and data markets in the cloud. The pricing schemes currently supported in data markets are typically simplistic: a prospective 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 [4, 9–11, 17] started the formal study of pricing schemes for assigning prices to relational queries issued over such data, a setting called *query-based pricing*. In query-based pricing, 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 query-based pricing 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 data buyer has an arbitrage opportunity to purchase the desired information for a lesser price. To formalize information disclosure, query-based pricing uses the mathematical tool of *query determinacy* [20, 21]. In the proposed framework [9], the seller specifies a set of fixed prices for views over the data (*price points*), based on which an arbitrage-free price is computed for any query.

At first glance, MBP seems similar to *query-based pricing*. 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 instance. 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 using random sampling. Also, in MBP, we enable the buyer to specify an *accuracy* constraint to control the predictive power of the model instance they buy.

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 $\mathbf{X}$. We assume that $\mathbf{X}$ and Y correspond to the *attributes* of a single relation. In this setting, the labeled examples are typically assumed to 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. For example, the ordinary least squares regression model assumes the data can be represented using d-dimensional hyperplane. An *ML model instance* is a specific instance of that ML model that corresponds to some prediction function $f : \mathcal{D}_{\mathbf{X}} \rightarrow \mathcal{D}_Y$. For example, an instance of the least squares linear regression model a given vector $w \in \mathbb{R}^d$. Given D, a *learning algorithm* computes such a prediction function. 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 *holdout test error* [5]. There are hundreds of ML models [19]; some of the most popular ones are Naive Bayes, other Bayesian Networks, and Generalized Linear Models (GLMs). These models are popular mainly due to their *interpretability*, simplicity, speed, and extensive systems support. Thus, we primarily focus on such models.

3 OVERVIEW OF MODEL-BASED PRICING

We now provide an overview of *model-based pricing*, including the key concepts and assumptions.

In a data marketplace, the *seller* provides the dataset $D(\mathbf{X}, Y)$ that they wish to sell. The *broker* specifies a set $\mathcal{M}$ of supported ML models (e.g., logistic regression and Naive Bayes for classification, ordinary least squares for regression, etc.). Let $\mathcal{H}_m$ denote the hypothesis space of an ML model $m \in \mathcal{M}$ and $h_m \in \mathcal{H}_m$ denote an instance of ML model m. The broker also specifies a *training algorithm* T_m for each supported ML model m. For the sake of simplicity, we also assume that the broker (or the seller) sets aside a holdout test set with the same schema as D to measure the *generalization error* (in short, error) of an ML model. The seller also provides the broker model-specific *pricing functions* for D, which help determine how much lower the price of a model instance trained on a subset of D should be compared to a model instance trained using all of D. We will discuss here two orthogonal scenarios for pricing:

Horizontal pricing : in this scenario, a subset of the examples in the dataset are used to train the model.

Vertical pricing : in this scenario, a subset of the available features (attributes) are used to train the model.

The *buyer* specifies an ML model $m \in \mathcal{M}$ they are interested in learning over (a subset of) D. They buyer also specifies a set of optimization criteria: *objective* and *budget constraints* (along with a random *seed* for reproducibility) to compute a desired model instance h_m. While the specifics of the optimization formulations differ for horizontal and vertical pricing, in general, the buyer can specify an *accuracy* (error) budget or a *price* budget and request the broker to optimize for the other. The broker performs the ML task based on the buyer's preferences and returns a learned model instance h_m to the buyer. We will later discuss on how to extend the framework to also handle cloud resource costs and time constraints: the buyer can then also specify a *time* budget, while the broker also factors in the cost of the computational resources used to handle the buyer's request. Our framework aims to make it easier for buyers to specify their ML tasks over data for sale in the cloud, for the sellers to specify their pricing functions, and for brokers to efficiently perform the model-based pricing computation.

4 PROBLEMS AND CHALLENGES

In this section we discuss the proposed pricing framework, consisting of horizontal pricing, vertical pricing, and pricing in practice.

4.1 Horizontal Pricing

Models. In horizontal pricing, only the data examples are priced. Formally, given a dataset (relation) $D(\mathbf{X}, Y)$ with N tuples, all features in $\mathbf{X}$ will be utilized for ML but only a subset of the N examples–determined by the buyer's preferences–will be used. The buyer performs a single transaction with the broker requesting them to subsample D on some specified criteria to train an ML model and return the model instance. We consider two settings from the seller's perspective: *uniform tuple pricing* and *stratified tuple pricing*. In the former, all data examples are treated equally and the price of a model instance depends only on the size of the sample. Formally, the seller specifies a *pricing function* $p : \mathcal{M} \times [N] \rightarrow \mathbb{R}^+$ to price a subset $D' \subseteq D$ at cost $p(m, |D'|)$. In the latter, however,

data examples are treated non-uniformly, i.e., the price depends on both the size and content of the sample. Formally, the seller provides the broker with a collection of datasets $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$ with the same schema: D_i has N_i tuples. Let $D = \cup_{i=1}^{k} D_i$ and $N = \sum_{i=1}^{k} N_i$. The seller also provides k pricing functions of the form $p_i : \mathcal{M} \times [N_i] \rightarrow \mathbb{R}^+$ to price subsets of D_i. Figure 1(B) shows three possible price curves that can be set by a seller given the error curve in Figure 1(A). Note that P is the price of the model instance trained on D, which must be smaller than the price of D itself.

There are two ways to model the user behavior: *(i)* the user has a fixed error tolerance and aims to minimize the price she has to pay to achieve that tolerance, and *(ii)* the user has a fixed price budget and aims to obtain a model instance that is as accurate as possible.

Challenges. As discussed in [9], a desideratum of query-based pricing is that it should be *arbitrage-free*: the buyer should not be able to answer a query Q by somehow "splitting" it into multiple queries such that the total price ends up being cheaper than the price of Q. In model-based pricing, the question becomes the following: *Is it possible for the buyer to obtain a similar (or higher) accuracy at a cheaper price by somehow "combining" multiple model instances trained on multiple smaller subsamples compared to one larger subsample?* We describe next how we can formally define these notions for uniform tuple pricing.

Definition 4.1 (Strong Arbitrage). If there exists $k \in \mathbb{Z}^+$ s.t. for *any* dataset D, there exists $k+1$ subsets of D, viz., $D_1, D_2, \cdots, D_k, D_{k+1}$ and a function $g(\cdot)$ that maps k models to 1 model, such that:

$$\sum_{i=1}^{k} p(m_i, |D_i|) \leq p(m_{k+1}, |D_{k+1}|),$$
$$\epsilon(g(m_1, m_2, \cdots, m_k)) \leq \epsilon(m_{k+1}), \tag{1}$$

then the pricing function $p(\cdot, \cdot)$ is k-strongly arbitrage.

Definition 4.2. Strong arbitrage freeness for uniform tuple pricing: if the pricing function $p(\cdot, \cdot)$ is not k-strongly arbitrage, then we call it k-strongly arbitrage free.

Definition 4.3 (Weak Arbitrage). If there exists $k \in \mathbb{Z}^+$ s.t. there exists *some* dataset D with subsets $D_1, D_2, \cdots, D_k, D_{k+1}$ and a function $g(\cdot)$ that maps k models to 1 model, such that:

$$\sum_{i=1}^{k} p(m_i, |D_i|) \leq p(m_{k+1}, |D_{k+1}|),$$
$$\epsilon(g(m_1, m_2, \cdots, m_k)) \leq \epsilon(m_{k+1}), \tag{2}$$

then the pricing function $p(\cdot, \cdot)$ is k-weakly arbitrage.

Definition 4.4. Weakly arbitrage freeness for uniform tuple pricing: if the pricing function $p(\cdot, \cdot)$ is not k-weakly arbitrage, then we call it k-weakly arbitrage free.

If a pricing function p is strongly arbitrage, then no matter what dataset the seller has, a buyer is always able to combine a few models at a cheaper price to obtain a lower error than buying a single model. That is to say, no matter what the dataset is, there is some subset of the data with an "invalid" price to the seller. A strong arbitrage pricing function is hardly useful to a seller and thus, must be avoided.

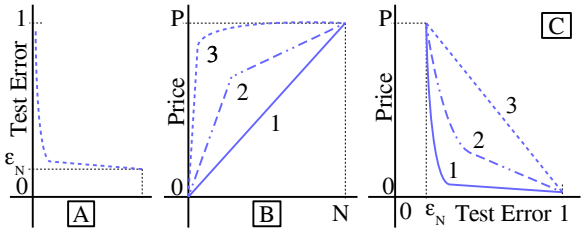

Figure 1: Illustrating pricing functions. (A) An ML classifier's error; (B) Different pricing functions in the model-based pricing setting for ML analytics. (C) the price error curve. The tradeoffs of the pricing scheme become clear. Curve 1 sees the price drop sharply even for modest increases in error, which makes it hard for buyers to fix a precise error budget. In contrast, curve 3 extracts a much higher price even if only small fraction of the dataset is used.

If a pricing function p is weak arbitrage, then for some dataset, a buyer could be able to achieve a lower error than using a single model by combining a few models with a lower price. It is certainly possible that a buyer cannot make profit from some dataset using a weak arbitrage pricing function. Therefore, a weak arbitrage price function can still be valid to a seller for certain datasets. Note that weak arbitrage freeness is desired, but strong arbitrage freeness is required. In other words, the former one implies the latter one.

Given the above definitions, we seek to characterize the class of pricing functions that satisfy the (strong/weak) arbitrage freeness guarantees. This task is challenging because the characterization depends on the particular ML task and the dataset on hand.

4.2 Vertical Pricing

Models. In this setting, we allow the buyer to pick a subset of the features to learn a model. Formally, the seller registers $D(\mathbf{X}, Y)$ with N tuples and $d = |\mathbf{X}|$ features with the broker. In contrast to horizontal pricing, in vertical pricing, we assume that all examples in D will be used for ML but perhaps only a subset of the d features, determined by the user's preferences, will be used. Similar to the uniform versus stratified distinction in horizontal pricing, there are two alternatives pricing settings in vertical pricing too, viz., *uniform feature pricing* and *stratified feature pricing*. The former assumes that all features are treated equally and the price only depends on the number of features used, while the latter assumes different prices for different features, as specified by the seller.

Challenges. Similar to horizontal pricing, we need to define the notion of arbitrage freeness and provide corresponding guarantees. A new challenge is how to incorporate *feature selection* methods with pricing. Since optimal feature selection is an NP-hard problem, most practical feature selection methods use different heuristic techniques [7]. Furthermore, any database dependencies among features also need to be taken into account in the pricing formulation [15]. These issues make it challenging to incorporate feature selection with pricing in a principled way.

4.3 Pricing in Practice

Models. Besides accuracy and price, computational cost is another important resource in practical machine learning tasks, especially those performed in the cloud setting.

Example. Consider the journalist Alice again, who wants to learn a logistic regression model over a demographic dataset. But now she has an additional constraint of having to finish this task within 30 minutes so that she can prepare her report in time. Of course, she also has a price budget and would like the lowest error. This leads to a new optimization problem space for the broker: should they spend more of the money for more resources to run more iterations or should they spend more of the money for a larger sample and run fewer iterations with fewer resources?

This example illustrates the fundamental challenge: the price-accuracy tradeoff of model-based pricing has to be expanded to a *price-accuracy-runtime tradeoff space*. To take into account the computational cost, the broker needs two more tools. First, a *runtime modeling* function should be provided to measure how much time it takes to run a specific ML model using a given number of tuples and features. Second, a *resource pricing* function must be provided to measure how much the resource costs. Note that this is the cost of computational resources, including CPUs or GPUs and memory, not the cost of the data.

Challenges. The first challenge is how to model the resource costs for ML. This is challenging partly because different ML algorithms have different runtime characteristics and hyper-parameters. From a technical perspective, this problem subsumes the problem of pricing cloud resources for ML tasks; thus, any new mechanisms developed as part of this work are likely to have implications beyond just the context of pricing.

We first consider ML models with no hyper-parameters that affect runtime. This primarily means that the training algorithm only needs only *one pass* over the data (e.g., inversion-based least squares regression and Naive Bayes), or the number of iterations is known (e.g., logistic regression with 20 iterations of gradient descent). It also includes single-pass or fixed-pass feature selection techniques. However, even for fixed-pass algorithms, the computations are varied, e.g., Naive Bayes requires counting distinct discrete values, while linear regression requires a matrix multiplication over a numeric space. Moreover, there is a large variety of resource types in the cloud: single-core, single-node multi-core, NUMA, distributed memory, etc. These issues make it challenging to develop generic mechanisms to model the resource costs of ML workloads. We plan to build upon some recent work in this space [6, 13].

In addition to fixed-pass training algorithms, some ML models enable users to specify accuracy-based *convergence* criteria, e.g., applying a threshold on the fractional decrease in the value of the loss for logistic regression. This makes the task of resource-aware model-based pricing even more challenging for two reasons. First, it is not clear how to obtain estimates of the number of iterations for a given convergence criterion for different sizes of subsamples; this requires developing a predictive model of how the convergence criterion affects the number of iterations. Second, it is not clear how to obtain error estimates for different numbers of iterations of the optimization algorithm used. To the best of our knowledge, the classical error bounds in learning theory do not capture the effects of iterative numerical optimization algorithms.

The final step in our vision is to integrate horizontal, vertical, and resource pricing into a single framework. The key challenge is that runtime will now become a first-class citizen in our optimization formulations. In other words, we have to consider the price-accuracy-runtime tradeoff holistically, wherein the buyer can constrain any combination of these quantities.

5 CONCLUSION AND FUTURE WORK

In this paper, we lay out our vision for a formal and practical framework for pricing datasets and resources for ML in the cloud. This is only the first step towards a comprehensive framework. There are several other exciting directions to expand our scope.

First, more complex ML models such as Bayesian networks, artificial neural networks, decision trees, SVMs, and statistical relational models are also frequently used. Non-relational data (images, text, video, etc.) might require complex feature extraction, possibly implicitly within an ML model (as in deep learning [16]). Handling such complex models is a key avenue to extend our framework. Second, we assumed that buyers know which ML model they want. But in practice, users often perform *model selection* and explore different ML models [5, 19] and refine their choices iteratively [14]. Systems such as AutoWeka [22] and MLbase [12] help automate parts of such model selection tasks. Incorporating such manual, iterative, or automated model selection and refinement along with pricing is another key avenue to extend our framework. Third, in some cases, datasets offered for sale come with privacy constraints because they were extracted from private users. Integrating model-based pricing with data privacy is also a core future challenge.

REFERENCES

[1] Big Data Exchange. www.bigdataexchange.com.
[2] Microsoft Azure Data Market. www.datamarket.azure.com.
[3] Twitter GNIP Audience API. gnip.com/insights/audience.
[4] M. Balazinska, B. Howe, and D. Suciu. Data Markets in the Cloud: An Opportunity for the Database Community. *PVLDB*, 4(12):1482–1485, 2011.
[5] J. H. Friedman et al. *The Elements of Statistical Learning: Data mining, Inference, and Prediction.* Springer-Verlag, 2001.
[6] A. Ghoting et al. SystemML: Declarative Machine Learning on MapReduce. In *ICDE*, 2011.
[7] I. Guyon et al. *Feature Extraction: Foundations and Applications.* New York: Springer-Verlag, 2001.
[8] J. M. Hellerstein et al. The MADlib Analytics Library or MAD Skills, the SQL. In *VLDB*, 2012.
[9] P. Koutris et al. Query-based Data Pricing. In *PODS*, 2012.
[10] P. Koutris et al. QueryMarket Demonstration: Pricing for Online Data Markets. *PVLDB*, 5(12):1962–1965, 2012.
[11] P. Koutris et al. In *SIGMOD 2013*, 2013.
[12] T. Kraska et al. MLbase: A Distributed Machine-learning System. In *CIDR*, 2013.
[13] A. Kumar et al. Learning Generalized Linear Models Over Normalized Data. In *SIGMOD*, 2015.
[14] A. Kumar et al. Model Selection Management Systems: The Next Frontier of Advanced Analytics. *ACM SIGMOD Rec.*, 44(4):17–22, Dec. 2015.
[15] A. Kumar et al. To Join or Not to Join? Thinking Twice about Joins before Feature Selection. In *SIGMOD*, 2016.
[16] Y. LeCun et al. Deep Learning. *Nature*, 10.1038/nature14539, 2015.
[17] C. Li and G. Miklau. Pricing Aggregate Queries in a Data Marketplace. In *WebDB*, 2012.
[18] Y. Low et al. GraphLab: A New Framework For Parallel Machine Learning. In *UAI*, 2010.
[19] T. M. Mitchell. *Machine Learning.* McGraw Hill, 1997.
[20] A. Nash et al. Determinacy and Rewriting of Conjunctive Queries Using Views: A Progress Report. In *ICDT*, 2007.
[21] A. Nash et al. Views and Queries: Determinacy and Rewriting. *ACM Trans. Database Syst.*, 35(3), 2010.
[22] C. Thornton et al. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *KDD*, 2013.
[23] C. Zhang, A. Kumar, and C. Ré. Materialization Optimizations for Feature Selection Workloads. In *SIGMOD*, 2014.