PrIU: A provenance-based approach for incrementally updating regression models

Anonymous
Anonymous
Anonymous, Anonymous
Anonymous

ABSTRACT

The ubiquitous use of machine learning algorithms brings new challenges to traditional database problems such as incremental view update. Much effort is being put in better understanding and debugging machine learning models, as well as in identifying and repairing errors in training datasets. Our focus is on how to assist these activities when they have to retrain the machine learning model after removing problematic training samples in cleaning or selecting different subsets of training data for interpretability. This paper presents an efficient provenance-based approach, PrIU, and its optimized version, PrIU-opt, for incrementally updating model parameters without sacrificing prediction accuracy. We prove the correctness and convergence of the incrementally updated model parameters, and validate it experimentally. Experimental results show that up to two orders of magnitude speed-ups can be achieved by PrIU-opt compared to simply retraining the model from scratch, yet obtaining highly similar models.

CCS CONCEPTS

• **Information systems** → **Data cleaning,Incremental maintenance,Data provenance**; • **Mathematics of computing** → *Exploratory data analysis*; • **Theory of computation** → *Convex optimization.*

KEYWORDS

Data provenance, machine learning, deletion propagation

ACM Reference Format:
Anonymous. 2020. PrIU: A provenance-based approach for incrementally updating regression models. In *Proceedings of* . ACM, New York, NY, USA, 16 pages. https://doi.org/xxxxxxxxxx

1 INTRODUCTION

In database terminology, this paper is about *efficient incremental view updates*, specifically about using provenance annotations to propagate the effect of deletions from the input data to the output. However, the views that we consider are *regression models* (linear and binomial/multinomial logistic regression) and the input data consists of the samples used to train these models.

The need for incremental techniques to efficiently update regression models arises in several contexts, for example data cleaning and interpretability. *Data cleaning* has been extensively studied by the database community [7, 10, 15, 45], and is typically an iterative and interactive process, allowing data analysts to alternate between analysis and cleaning tasks, as well as to interact with other parties such as IT staff and data curators [31]. Machine learning techniques are particularly sensitive to dirty data in training datasets, since it can result in erroneous models and counter-intuitive predictions for test datasets [7]. A number of techniques have therefore recently been proposed for detecting and repairing dirty data in machine learning, e.g., [23, 30]. The work presented in this paper can be incorporated into these data cleaning pipelines by assuming that dirty data in the training set has already been detected, and addresses the next step by providing a solution for incrementally updating the machine learning model after the dirty data is removed.

Interpretability is also a major concern in machine learning (see, for example, the general discussions in [13, 36], the extensive human subjects experiments in [44], as well the many references in these papers). The problem is being studied from several different perspectives (see Sec. 2). The data-driven approaches of [13, 33] discover *factors of interpretability* by performing *repeated retraining* of models using multiple different subsets of a training dataset to understand the relationship between samples with certain feature characteristics and the model behavior. Such repeated retraining also occurs in *model debugging* [23, 26, 32] and *deletion diagnostics* [8].

In this respect, our work shares goals with [28], which develops an *influence function* to approximately quantify the influence of a *single* training sample on the model parameters and prediction results; this can also be used for estimating the model parameter change after the removal of one training sample. However, extending the influence function approach to multiple training samples significantly weakens prediction accuracy. *In contrast, our techniques are not only efficient but significantly more accurate.*

Connection to Provenance. Note that the problem of incrementally updating the model after removing a subset of the training samples can be seen as a question of *data provenance* [3, 6, 18]. Data provenance tracks the dependencies between input and output data; in particular, the *provenance semiring framework* [18, 19] has been used for applying incremental updates (specifically deletions) to views.

In the semiring framework, input data is annotated with provenance tokens which are carried through the operators performed on the data (e.g. select, project, join, union). Output data is then annotated with *provenance polynomials* expressed in terms of the provenance tokens. When an input tuple is deleted, the effect on the output can be efficiently calculated by essentially "zeroing out" its token in the provenance polynomial. Recently, the framework has been extended to include basic linear algebra operations: matrix addition and multiplication [50]. In this extension, the provenance polynomials play the role of scalars and multiplication with scalars plays the role of annotating matrices and vectors with provenance.

As an example, suppose that p, q, r, s are provenance tokens that annotate samples in a training dataset. Our methods will show that vectors of interest (such as the vector of model parameters) can be expressed with provenance-annotated expressions such as:

$$\mathbf{w} = (p^2q * \mathbf{u}) + (qr^4 * \mathbf{v}) + (ps * \mathbf{z})$$

Here, $\mathbf{u}, \mathbf{v}, \mathbf{z}$ are numerical vectors signifying contributions to the answer $\mathbf{w}$ and they are annotated (algebraic operation $*$) with p^2q, qr^4, ps which are provenance polynomials to be read as follows: the provenance p^2q represents the use of both data items labeled p and q and, in fact, the first item is used twice. Now suppose the data item annotated r is deleted while those annotated p, q, s are retained. We can express the updated value of $\mathbf{w}$ under this deletion by setting r to the "provenance 0 polynomial", denoted 0_{prov} which signifies absence, and p, q, s to the "provenance 1 polynomial", denoted 1_{prov}, which signifies "neutral" presence, no need to track further. The algebraic properties of provenance polynomials and of their annotation of matrices/vectors ensure what one would expect, e.g, $0_{\text{prov}} \cdot r^4 = 0_{\text{prov}}$ as well as $0_{\text{prov}} * \mathbf{v} = \mathbf{0}$ (the all-zero vector) and $1_{\text{prov}} * \mathbf{z} = \mathbf{z}$. It follows that under this deletion $\mathbf{w} = \mathbf{u} + \mathbf{z}$.

Approach. In this paper, we use the extension of the semiring framework to matrix operations to track the provenance of input samples through the training of logistic regression and linear regression models using gradient descent and its variants. In each iteration of the training phase, a gradient-based "update rule" updates the model parameters, which can be annotated with provenance polynomials. For logistic regression, we can achieve this via piecewise linear interpolation over the non-linear components in the gradient update rule.

In addition to enabling provenance tracking, the linearization of the gradient update rule allows us to separate the contributions of the training samples from the contributions of the model parameters from the previous iteration. As a result, the effect of deleting training samples on the gradient update rule can be obtained by "zeroing out" the provenance tokens corresponding to those samples.

Challenges. Reasoning over provenance to enable incremental updates introduces significant overhead in the gradient descent calculation. To speed up incremental updates over model parameters for dense datasets, we use several optimizations in our implementation, PrIU: First, between iterations during the training phase over the full training dataset, we cache intermediate results (some matrix expression) that capture only the contribution of the training samples. These are annotated with provenance. Then during the model update phase, the propagation of the deletion of a subset of samples comes down to a subtraction of the "zeroed-out" contributions of the removed samples. Second, we apply singular value decomposition (SVD) over the intermediate results to reduce their dimensions. An optimized version of PrIU, PrIU-opt, is also designed for further optimizations over datasets with small feature sets using incremental updates to eigenvalues. (For logistic regression, it is used by terminating provenance tracking early when provenance expressions stabilize. See Section 5 for more details). But the optimizations above cannot work for sparse datasets, for which we use only the linearization of the update rule for logistic regression.

As we shall see, PrIU and PrIU-opt can lead to speed-ups of up to 2 orders of magnitude when compared to a baseline of retraining the model from the updated input data; however, for sparse datasets the speedup is only 10%. While the practical impact of this speed-up may be small for an engineer who only deletes one subset of training samples, especially if retraining takes only a few minutes, the impact is much greater for an engineer who repeatedly removes multiple different subsets of training samples, e.g. when exploring factors of interpretability. In this case, even one order of magnitude speed-up reduces exploration from several hours to a few minutes.

Contributions of this paper include:

(1) A theoretical framework which enables data provenance to be tracked and used for fast incremental model updates when subsets of training samples are removed. The framework extends the approach in [17, 18, 50] to linear regression and (binary and multinomial) logistic regression models.
(2) Analytical results showing the convergence and accuracy of the updated model parameters for logistic regression, which are approximately computed by applying piecewise linear interpolation over the non-linear operations in the model parameter update rules.
(3) Efficient provenance-based algorithms, PrIU and PrIU-opt, which achieve fast model updates after removing subsets of training samples.
(4) Extensive experiments showing the effectiveness and accuracy of PrIU and PrIU-opt in incrementally updating the linear regression and logistic regression models compared to the straightforward approach of retraining from scratch, as well as compared to implementing an extension of the influence function in [28].
(5) Enabling work on interpretability that seeks to understand the effect of removing *subsets* of the training data, rather than just of a single training sample.

The remainder of the paper is organized as follows. In Section 2, we describe related work in incremental model maintenance, data provenance, data cleaning, and machine learning model interpretability. Section 3 reviews the basic concepts of linear regression and logistic regression. The theoretical development of how to use provenance in the update rules of linear regression and logistic regression is presented in Section 4, and its implementation provided in Section 5. Experimental results comparing our approach to other solutions are presented in Section 6. We conclude in Section 7.

To our knowledge, this is the first work to use provenance for the purpose of incrementally updating machine learning model parameters.

2 RELATED WORK

Incremental model maintenance. There have been several proposals for materializing machine learning models for future reuse. [11, 20, 38] target the problem of efficiently updating the model as the training data changes, which focus primarily on linear regression and Naive Bayes models, and use closed-form solutions (rather than iterative algorithms, e.g., gradient-based approaches) of the model parameters to determine incremental updates in light of additions and deletions of training samples while [21] deals with how to merge pre-materialized models to construct new models based on user requests. In addition, [20] also deals with incremental updates of the model parameters based on the Mixture Weight Methods (a variant of gradient descent) for logistic regression. The method, however, puts additional training samples into another batch and averages the pre-computed parameters derived from other batches (over the original data) with the parameters computed over the additional batch. This cannot be used for incremental deletions which is our focus in this paper.

The basic ideas of [11, 20, 38] on how to incrementally update linear regression models are somewhat similar. Due to the existence of the matrix inverse operations in the closed-form solution for linear regression, only the intermediate results built with linear operations are maintained as views. They are updated when insertion or deletion happens in the input training data. After that, matrix inversion is used to compute the final updated model parameters. In contrast, our approach proceeds directly to a gradient descent-based linear regression. As we shall see, our experiments show that our approach is more efficient than the closed-form update.

Data provenance. Data provenance captures where data comes from and how it is processed. Within the database community, various approaches have been proposed to track provenance through queries, e.g. where and why provenance [3], and semiring provenance [2, 18]. Provenance is used to identify the source of errors in computational processes, such as workflows [1] and network diagnostics [51]. It is also used to support efficient incremental updates through database queries and schema mappings [16, 17, 24] and workflow computation [14]. Provenance support for linear algebra operations in the context of machine learning tasks has also been recently studied [50]. This work was mentioned in [4] as a first step in *using data provenance for interpretability* of machine learning models.

Data cleaning. The goal of data cleaning is to detect and fix errors in data, and is a crucial step in preparing data for data analytics/machine learning tasks [12, 32]. However, if erroneous/dirty data is detected after the model has been trained, the machine learning algorithm must be rerun to obtain the updated model parameters. This repetitive training can cause significant delays when large volumes of data are processed. One approach is to start each training phase by setting the initial model parameters to the ones generated by the previous training phase over the dirty data [32]. Our contribution is orthogonal to this approach, and updates the machine learning model parameters directly by reasoning over provenance rather than retraining from scratch.

Interpreting and understanding ML models. Fully understanding the behavior of ML models, especially deep neural network models, is difficult due to their complexity. Moreover, there are different perspectives on what we should understand. For example, one approach separates model components into "shape" functions, one for each feature, in generalized additive models, in particular for linear and logistic regression [5, 37]. Closest to our perspective is the idea of *influence function* [28] (similar problem is also mentioned in [43]), which originates from *deletion diagnostics* in statistics [8]. [28] estimates the effect of removing a *single* training sample on the already obtained model, *without* retraining the model. The influence function uses the Taylor expansion of the derivative of a customized objective function for the model parameter. The calculation (and thus the approximation of model parameter change) is only based on lower-order terms in Taylor expansion.

This can be seen as a method for incremental model update for just one sample deletion. In fact, we have observed that the method could be extended to deleting an arbitrary number of samples, which led us to compare it experimentally to our approach. The results (see Section 6) show that this approach leads to very inaccurate results when multiple training samples are deleted.

3 PRELIMINARIES

We give an overview of linear and logistic regression along with the gradient-based method for learning model parameters. Assume a training dataset $(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ is an $n \times m$ matrix representing the feature matrix while $\mathbf{Y}$ is an $n \times 1$ vector representing the labels, i.e.:

$$\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\right]^T \mathbf{Y} = \left[y_1, y_2, \ldots, y_n\right]^T \tag{1}$$

For both linear and logistic regression we only focus on a common case: $L2-$regularization. The objective functions of linear regression, binary logistic regression and multinomial logistic regression with $L2-$regularization are presented in Equations 2-4 respectively [1]

$$h(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\mathbf{w})^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{2}$$

$$h(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\ln(1 + \exp\{-y_i\mathbf{w}^\top\mathbf{x}_i\}) + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{3}$$

$$\begin{aligned} h(\mathbf{w}) &= \frac{1}{n}\sum_{k=1}^{q}\sum_{y_i=k}(\ln(\sum_{j=1}^{q} e^{\mathbf{w}_j^\top\mathbf{x}_i}) - \mathbf{w}_k^T\mathbf{x}_i) \\ &+ \frac{\lambda}{2}||vec([\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_q])||_2^2 \\ \mathbf{w} &= vec([\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_q]) \end{aligned} \tag{4}$$

where $\mathbf{w}$ is the vector of model parameters and λ is the *regularization rate*. For simplicity, we denote $\mathbf{w} = vec([\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_q])$ for multinomial logistic regression where q represents the number of possible classes. Typical learning methods for computing $\mathbf{w}$ are to apply gradient descent (GD) or its variant, stochastic gradient descent (SGD) or mini-batch stochastic gradient method (mb-SGD) [46] to minimize the objective function $h(\mathbf{w})$ iteratively. GD, SGD and mb-SGD are the same in nature since mb-SGD can be regarded as a generalization of GD and SGD. They are therefore called Gradient-based method (GBM) for short, and hereafter we will only take mb-SGD as an example. Considering the similarities between binary logistic regression and multinomial logistic regression and the complexity of the computation related to the latter one, we will only present the formulas related to binary logistic regression below. All the theorems that hold for binary logistic regression can be also proven to be true for multinomial logistic regression.

At each iteration, mb-SGD updates the $\mathbf{w}^{(t)}$ by using the average gradient of $h(\mathbf{w})$ over a randomly selected mini-batch from the training dataset. Specifically, for linear regression and logistic regression, the rule for updating $\mathbf{w}^{(t)}$ under mb-SGD is presented below (Equations 5 and 6 respectively):

$$\mathbf{w}^{(t+1)} \leftarrow (1 - \eta_t\lambda)\mathbf{w}^{(t)} - \frac{2\eta_t}{B}\sum_{i\in\mathscr{B}^{(t)}}\mathbf{x}_i(\mathbf{x}_i^T\mathbf{w}^{(t)} - y_i) \tag{5}$$

$$\begin{aligned}\mathbf{w}^{(t+1)} &\leftarrow (1 - \eta_t\lambda)\mathbf{w}^{(t)} \\ &+ \frac{\eta_t}{B}\sum_{i\in\mathscr{B}^{(t)}} y_i\mathbf{x}_i(1 - \frac{1}{1 + \exp\{-y_i\mathbf{w}^{(t)T}\mathbf{x}_i\}})\end{aligned} \tag{6}$$

where η_t is called the *learning rate* and $\mathscr{B}^{(t)}$ represents a mini-batch of B training samples. For SGD, $\mathscr{B}^{(t)}$ includes only one sample ($B = 1$), while for GD, $\mathscr{B}^{(t)}$ includes all the training samples ($B = n$).

4 ITERATION MODELS

In this section, we first discuss the annotation with provenance of the gradient-bases update rules in our approach. Next, we discuss for the non-linear operations in logistic regression the linearization that makes our provenance annotation framework usable. Finally, we give a rigorous theoretical analysis of the convergence of the iterative process with provenance-annotated update rules for both linear and logistic regression model and the similarity to the expected results after linearization for logistic regression models.

4.1 Provenance annotations for matrices

In the semiring framework [2, 18, 19] one begins by annotating input data with elements of a set T of *provenance tokens*. These annotations are then propagated through query operators as they combine according to two operations: "+" that records *alternative* use of information, as in relational union or projection, and "·", that records *joint* use of information, as in relational join. With these, the annotations become *provenance polynomials* whose indeterminates are tokens

[1]Here we assume that the two possible labels in binary logistic regression are 1 and -1.

and with coefficients in $\mathbb{N}$. For example, the monomial p^2q is the provenance of a result for which the data item annotated p was used *twice* together with the item annotated q used once. We denote the set of polynomials by $\mathbb{N}[T]$.

In the extension of the framework to matrix algebra [50], annotation formally becomes a *multiplication of vectors with scalars* as in linear algebra. The role of scalars is played by provenance polynomials and the role of vectors, of course, is played by matrices (generalizing their row vectors and the transposes of these).

Matrices annotated with provenance polynomials form a nice algebraic structure that extends matrix multiplication and addition. We denote multiplication with scalars by "$*$" writing $\mathfrak{p} * \mathbf{A}$ for the matrix $\mathbf{A}$ annotated with the provenance polynomial $\mathfrak{p}$. For space reasons we cannot repeat here the technical development in [50], however, we mention a crucial algebraic property of annotated matrix multiplication, which also illustrates combining provenance in joint use:

$$(\mathfrak{p}_1 * \mathbf{A}_1)(\mathfrak{p}_2 * \mathbf{A}_2) = (\mathfrak{p}_1 \cdot \mathfrak{p}_2) * (\mathbf{A}_1\mathbf{A}_2)$$

We apply this framework to tracking input training samples through GBM's in which the update involves only matrix multiplication and addition. Let the training dataset be $(\mathbf{X}, \mathbf{Y})$ where $\mathbf{X}$ is an $n \times m$ feature matrix and $\mathbf{Y}$ is an $n \times 1$ column vector of sample labels. For $i = 1, \dots, n$, we annotate every sample $(\mathbf{x}_i, y_i)$ ($\mathbf{x}_i$ and $[y_i]$ are the i'th rows in $\mathbf{X}$ respectively $\mathbf{Y}$) with a distinct provenance token p_i. Next, we decompose $\mathbf{X}$ and $\mathbf{Y}$ as algebraic expressions in terms of $p_1 * \mathbf{x}_1, \dots, p_n * \mathbf{x}_n, p_1 * [y_1], \dots, p_n * [y_n]$ and some matrices made up of the reals 0 and 1. These "helper" matrices are annotated with the provenance polynomial $1_{\text{prov}} \in \mathbb{N}[T]$ (has only a term of degree zero which is the natural number 1) meaning "always available, no need to track". We illustrate with the provenance-annotated $\mathbf{X}$ when $n = 2$:

$$\mathbf{X} = (1_{\text{prov}} * \begin{bmatrix}1\\0\end{bmatrix})(p_1 * \mathbf{x}_1) + (1_{\text{prov}} * \begin{bmatrix}0\\1\end{bmatrix})(p_2 * x_2) =$$

$$= (p_1 * \begin{bmatrix}\mathbf{x}_1\\0\dots0\end{bmatrix}) + (p_2 * \begin{bmatrix}0\dots0\\\mathbf{x}_2\end{bmatrix})$$

When $\mathbf{X}$ is transposed, a similar decomposition applies in terms of the annotated column vectors $p_i * \mathbf{x}_i^T$. We also note that the algebra of annotated matrices follows the same laws as the usual matrix algebra. Consequently, we can perform in the algebra of provenance-annotated matrices the calculations involved in the gradient-based update rules. For illustration, a calculation involving $\mathbf{X}$ that without provenance takes the form $\sum_{i=1}^n \alpha_i \mathbf{x}_i \mathbf{x}_i^T$ (where α_i are some real numbers) becomes with provenance annotations

$$\sum_{i=1}^{n}(1_{\text{prov}} * [\alpha_i])(p_i * \mathbf{x}_i)(p_i * \mathbf{x}_i^T) = \sum_{i=1}^{n} p_i^2 * (\alpha_i \mathbf{x}_i \mathbf{x}_i^T)$$

.

And here is the provenance-annotated expression for the update rule of linear regression (i.e. Equation 5):

$$\mathcal{W}^{(t+1)} \leftarrow [(1-\eta_t\lambda)(1_k * \mathbf{I}) - \frac{2\eta_t}{\mathcal{P}^{(t)}} \sum_{i \in \mathscr{B}^{(t)}} p_i^2 * \mathbf{x}_i \mathbf{x}_i^T]\mathcal{W}^{(t)} + \frac{2\eta_t}{\mathcal{P}^{(t)}} \sum_{i \in \mathscr{B}^{(t)}} p_i^2 * \mathbf{x}_i y_i \tag{7}$$

where $\mathcal{W}^{(t)}$ represents the provenance-annotated expression for the vector $\mathbf{w}^{(t)}$ of model parameters while $\mathcal{P}^{(t)}$ represents a provenance-annotated expression for the number of samples in the min-batch $\mathscr{B}^{(t)}$, for example, following the approach to aggregation in [2], $\mathcal{P}^{(t)} = \sum_{i \in \mathscr{B}^{(t)}} p_i * 1$.

In the semiring framework there is no division operation so we used fractions with denominator $\mathcal{P}^{(t)}$ in Equation 7 only for notational purposes. As we shall see immediately below, in incremental update $\mathcal{P}^{(t)}$ can be replaced with an integer.

As with the other applications of the semiring framework, deletion propagation is done by "zeroing-out" the deleted samples. That is, if sample i is deleted we set the corresponding provenance token $p_i = 0_{\text{prov}} \in \mathbb{N}[T]$ (has only a term of degree zero which is the natural number 0). The challenge, as detailed in the following section is how to do this efficiently throughout the gradient descent.

For the samples that remain we obtain (after we stop the iterations) a provenance-annotated expression that can be put in the form $\mathcal{W} = \sum \mathfrak{m}_k * \mathbf{u}_k$ where $\mathfrak{m}_k$ is a *monomial* in the provenance tokens and each $\mathbf{u}_k$ is a vector of contributions to the model parameters. To get the updated vector of model parameters we set each remaining provenance token to 1_{prov} obtaining $\mathbf{w}^{\text{upd}} = \sum \mathbf{u}_k$. And, as promised, we notice that when all the provenance tokens are set to 0_{prov} or 1_{prov} the provenance expression $\mathcal{P}^{(t)}$ comes down to an integer. Denoting this integer by $B_U^{(t)}$ and denoting the subset of training samples that are deleted by $(\Delta\mathbf{X}, \Delta\mathbf{Y})$, the provenance-annotated update rule for $\mathcal{W}^{(t+1)}$ becomes:

$$\mathcal{W}_U^{(t+1)} \leftarrow [(1-\eta_t\lambda)(1_{\text{prov}} * \mathbf{I}) - \frac{2\eta_t}{B_U^{(t)}} \sum_{\substack{i \in \mathscr{B}^{(t)} \\ ,\mathbf{x}_i \notin \Delta\mathbf{X}}} p_i^2 * \mathbf{x}_i \mathbf{x}_i^T]\mathcal{W}_U^{(t)} + \frac{2\eta_t}{B_U^{(t)}} \sum_{\substack{i \in \mathscr{B}^{(t)} \\ ,\mathbf{x}_i \notin \Delta\mathbf{X} \\ ,y_i \notin \Delta\mathbf{Y}}} p_i^2 * \mathbf{x}_i y_i \tag{8}$$

4.2 Linearization for logistic regression

The model in [50] supports tracking provenance through matrix addition and multiplication. In order to apply it to GBM for logistic expression, we linearize, using *piecewise linear interpolation*, the non-linear operations in the corresponding update rules, i.e. Equation 6.

In Equation 6, the non-linear operations can be abstracted as $f(x) = 1 - \frac{1}{1+e^{-x}}$, where the value of the product $y_i \mathbf{w}^{(t)T} \mathbf{x}_i$ is assigned to the variable x in Equation 6. Then $f(x)$ can be approximated by applying 1-D piecewise linear interpolation [29]. So for each $\mathbf{x}_i$ and $\mathbf{w}^{(t)}$, $f(y_i\mathbf{w}^{(t)T}\mathbf{x}_i)$ can be approximated by $s(y_i\mathbf{w}^{(t)T}\mathbf{x}_i) = a^{i,(t)}y_i\mathbf{w}^{(t)T}\mathbf{x}_i + b^{i,(t)}$, where $a^{i,(t)}$ and $b^{i,(t)}$ are the linear coefficients produced by the linearizations, which depends on which sub-interval (defined by piecewise linear interpolation) the value of $y_i\mathbf{w}^{(t)T}\mathbf{x}_i$ locates and thus should be varied between different $\mathbf{x}_i$ and different $\mathbf{w}^{(t)T}$ (see the associated superscript).

Throughout the paper, we will consider the case in which the variable x in $f(x)$ is defined within an interval $[-a, a]$ ($a = 20$) that is equally partitioned into 10^6 sub-intervals; for x outside $[-a, a]$, we assume that $s(x)$ is a constant since when $|x| > a$, the value of $f(x)$ is very close to its bound (0 or 1). We will show that the length of each sub-interval influences the approximation rate.

In terms of multinomial logistic regression, the non-linear operations in its update rule is the softmax function, which is a vector-valued function and thus requires piecewise linear interpolation in multiple dimensions, which can be achieved by using the interpolation method proposed in [49].

After the interpolation step over the update rules for binary logistic regression, Equation 6 can be approximated as:

$$\mathbf{w}_L^{(t+1)} \approx [(1-\eta_t\lambda)\mathbf{I} + \frac{\eta_t}{B}\sum_{i\in\mathscr{B}^{(t)}} a^{i,(t)}\mathbf{x}_i\mathbf{x}_i^T]\mathbf{w}_L^{(t)} + \frac{\eta_t}{B}\sum_{i\in\mathscr{B}^{(t)}} b^{i,(t)}y_i\mathbf{x}_i \tag{9}$$

in which $\mathbf{w}_L^{(t)}$ represents the model parameter after linearization at tth iteration. By annotating each training sample $\mathbf{x}_i$ with provenance token p_i and by taking the similar derivation of Equation 8, after the removal of the subset of training samples the provenance expression becomes:

$$\begin{aligned}\mathcal{W}_{LU}^{(t+1)} \leftarrow & [(1-\eta_t\lambda)(1_{\text{prov}} * \mathbf{I}) \\ & + \frac{\eta_t}{B_U^{(t)}}\sum_{i\in\mathscr{B}^{(t)},\mathbf{x}_i\notin\Delta\mathbf{X}} p_i^2 * (a^{i,(t)}\mathbf{x}_i\mathbf{x}_i^T)]\mathcal{W}_{LU}^{(t)} \\ & + \frac{\eta_t}{B_U^{(t)}}\sum_{i\in\mathscr{B}^{(t)},\mathbf{x}_i\notin\Delta\mathbf{X},y_i\notin\Delta\mathbf{Y}} p_i^2 * (b^{i,(t)}y_i\mathbf{x}_i)\end{aligned} \tag{10}$$

By setting all the p_i in Equation 10 as 1_{prov}, we can get the update rule for the updated model parameter $\mathbf{w}_{LU}^{(t)}$, i.e.:

$$\begin{aligned}\mathbf{w}_{LU}^{(t+1)} \approx & [(1-\eta_t\lambda)\mathbf{I} \\ & + \frac{\eta_t}{B_U^{(t)}}\sum_{\substack{i\in\mathscr{B}^{(t)}\\ \mathbf{x}_i\notin\Delta\mathbf{X}}} a^{i,(t)}\mathbf{x}_i\mathbf{x}_i^T]\mathbf{w}_{LU}^{(t)} + \frac{\eta_t}{B_U^{(t)}}\sum_{\substack{i\in\mathscr{B}^{(t)}\\ ,\mathbf{x}_i\notin\Delta\mathbf{X}\\ ,y_i\notin\Delta\mathbf{Y}}} b^{i,(t)}y_i\mathbf{x}_i\end{aligned} \tag{11}$$

4.3 Convergence analysis for provenance-annotated iterations

One concern in using GBM is whether the model parameters ultimately converge. This has been extensively studied in the machine learning community [27, 34, 47, 48]. In [27], convergence conditions have been provided for various gradient methods (GD, SGD, Greedy Coordinate Descent and so on) over objective functions without constraints under the Polyak-Lojasiewicz Inequality (PL Inequality)[41]. Those convergence conditions can exactly fit linear regression and logistic regression with L2-regularization, which satisfy the PL Inequality [27].

A similar concern occurs when GBM is coupled with provenance, i.e. whether the provenance expression $\mathcal{W}_U^{(t)}$ in Equation 8 and $\mathcal{W}_{LU}^{(t)}$ in Equation 10 converge in the case when the original model parameter $\mathbf{w}^{(t)}$ converges. We propose the following definition for the convergence of provenance-annotated expressions.

Definition 1. **Convergence of provenance-annotated expressions.** The expression $\mathcal{W}^{(t)} = \sum_i \mathfrak{p}_i^{(t)} * \mathbf{u}_i^{(t)}$ converges when $t \longrightarrow \infty$ iff every matrix $\mathbf{u}_i^{(t)}$ converges when $t \longrightarrow \infty$.

As mentioned before, we hope that the convergence of $\mathcal{W}_U^{(t)}$ and $\mathcal{W}_{LU}^{(t)}$ can be achieved when $\mathbf{w}^{(t)}$ can converge. The convergence conditions of $\mathbf{w}^{(t)}$ are presented below:

Lemma 1. ***Convergence conditions for general mb-SGD.*** *[27] Given an objective function $h(\boldsymbol{w})$, L2−regularization term $\lambda||\boldsymbol{w}||_2^2$, once the learning rate η_t satisfies: 1) $\eta_t < \frac{1}{2\lambda}$; 2) η_t is a constant across all the iterations (denoted by η), then $\boldsymbol{w}^{(t)}$ converges and we can obtain a linear convergence rate up to a solution level that is proportional to η, i.e.:*

$$E(h(\boldsymbol{w}^{(t)}) - h^*) \le (1-2\lambda\eta)^t(h(\boldsymbol{w}^{(0)}) - h^*) + O(\eta) \tag{12}$$

where h^* represents the optimal value of $h(\boldsymbol{w})$.

Unfortunately, our theoretical analysis shows that there is no convergence guarantee for $\mathcal{W}_U^{(t)}$ and $\mathcal{W}_{LU}^{(t)}$ under the convergence conditions from Lemma 1, i.e.:

Theorem 2. *$\mathcal{W}_U^{(t)}$ in Equation 8 and $\mathcal{W}_{LU}^{(t)}$ in Equation 10 need not converge under the conditions in Lemma 1.* [2]

However, $\mathcal{W}_U^{(t)}$ in Equation 8 and $\mathcal{W}_{LU}^{(t)}$ in Equation 10 converge under the conditions in Lemma 1 with one more assumption about the provenance expression, i.e.:

Theorem 3. *The expectation of $\mathcal{W}_U^{(t)}$ in Equation 8 and of $\mathcal{W}_{LU}^{(t)}$ in Equation 10, converge when $t \longrightarrow \infty$ if we also assume that provenance polynomial multiplication is* idempotent.

[2]Due to space limitations the proofs of the theorems are omitted. They will appear in the full version of the paper.

Intuitively speaking, the assumption of multiplication idempotence for provenance polynomials means that we do not track *multiple joint uses of the same data sample*, which is not problematic for deletion propagation.

4.4 Accuracy analysis for linearized logistic regression

The next question is whether the approximated model parameters after linearization of Equation 6 (i.e. $\mathbf{w}_L^{(t)}$ in Equation 9) is close enough to the real model parameters from Equation 6. By following the approximation property of piecewise linear interpolation, we can prove that the distance between $\mathbf{w}^{(t)}$ and $\mathbf{w}_L^{(t)}$ is very small.

Theorem 4. *$||E(\mathbf{w}^{(t)} - \mathbf{w}_L^{(t)})||_2$ is bounded by $O(\Delta x)$ where Δx is an arbitrarily small value representing the length of the longest sub-interval used in piecewise linear interpolations.*

Furthermore, in terms of the updated model parameters for logistic regression, we also need to guarantee that the updated parameters $\mathbf{w}_{LU}^{(t)}$ are close to the real updated model parameters without linearization (denoted by $\mathbf{w}_{RU}$), i.e.:

$$\mathbf{w}_{RU}^{(t+1)} \leftarrow (1-\eta_t\lambda)\mathbf{w}_{RU}^{(t)} + \frac{\eta_t}{B_U^{(t)}} \sum_{\substack{i \in \mathscr{B}^{(t)} \\ \mathbf{x}_i \notin \Delta\mathbf{X} \\ ,y_i \notin \Delta\mathbf{Y}}} y_i \mathbf{x}_i f(y_i \mathbf{w}_{RU}^{(t)} \mathbf{x}_i) \tag{13}$$

Recall that $f(x) = 1 - \frac{1}{1+e^{-x}}$. Note that the linear coefficients $a^{i,(t)}$ and $b^{i,(t)}$ in Equation 11 are actually derived in the training phase where all samples exist (rather than in the model update phase), which implies that a larger difference between $\mathbf{w}_{LU}^{(t)}$ and $\mathbf{w}_{RU}^{(t)}$ should be expected. Surprisingly, we can prove that the distance between $\mathbf{w}_{LU}^{(t)}$ and $\mathbf{w}_{RU}^{(t)}$ is still small enough.

Theorem 5. *$||E(\mathbf{w}_{LU}^{(t)} - \mathbf{w}_{RU}^{(t)})||_2$ is bounded by $O(\Delta x) + O(\eta_t)$, where η_t is the learning rate of the GBM.*

5 IMPLEMENTATION

We now discuss how the ideas in the previous section are implemented in PrIU and PrIU-opt, for both linear and logistic regression. Along the way, time and space complexity analyses, as well as theorems that justify our approximation strategies used in PrIU and PrIU-opt, are provided.

5.1 PrIU: Linear regression

In Equation 8, by setting all the p_i as 1_{prov}, the expression $\sum_{i \in \mathscr{B}^{(t)}, \mathbf{x}_i \notin \Delta\mathbf{X}} p_i^2 * \mathbf{x}_i\mathbf{x}_i^T$ becomes $\sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i\mathbf{x}_i^T - \sum_{i \in \mathscr{B}^{(t)}, \mathbf{x}_i \in \Delta\mathbf{X}} \mathbf{x}_i\mathbf{x}_i^T$, in which the first term, $\sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i\mathbf{x}_i^T$, can be regarded as provenance information and thus cached as an intermediate result for each mini-batch during the training phase for the original model parameter $\mathbf{w}^{(t)}$. Thus we only need to compute the latter term during the incremental update phase. $\sum_{i \in \mathscr{B}^{(t)}, \mathbf{x}_i \notin \Delta\mathbf{X}, y_i \notin \Delta\mathbf{Y}} \mathbf{x}_i y_i$ can be computed in a similar way. In the end, Equation 8 can then be rewritten as follows for the purpose of incremental updates:

$$\mathbf{w}_U^{(t+1)} \leftarrow [(1-\eta_t\lambda)\mathbf{I} - \frac{2\eta_t}{B_U^{(t)}} \sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i\mathbf{x}_i^T - \sum_{\substack{i \in \mathscr{B}^{(t)} \\ \mathbf{x}_i \in \Delta\mathbf{X}}} \mathbf{x}_i\mathbf{x}_i^T]\mathbf{w}_U^{(t)} + \frac{2\eta_t}{B_U^{(t)}}(\sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i y_i - \sum_{\substack{i \in \mathscr{B}^{(t)} \\ \mathbf{x}_i \in \Delta\mathbf{X}, y_i \in \Delta\mathbf{Y}}} \mathbf{x}_i y_i) \tag{14}$$

Note that $\sum_{i \in \mathscr{B}^{(t)}, \mathbf{x}_i \in \Delta\mathbf{X}} \mathbf{x}_i\mathbf{x}_i^T$ can be rewritten into matrix form, i.e. $\Delta\mathbf{X}_{\mathscr{B}^{(t)}}^T \Delta\mathbf{X}_{\mathscr{B}^{(t)}}$ where $\Delta\mathbf{X}_{\mathscr{B}^{(t)}}$ is a matrix consisting of the removed samples in the mini-batch $\mathscr{B}^{(t)}$. The associativity property of matrix multiplication can also be used to avoid expensive matrix-matrix multiplications (i.e. $\Delta\mathbf{X}_{\mathscr{B}^{(t)}}^T \Delta\mathbf{X}_{\mathscr{B}^{(t)}}$) by conducting more efficient matrix-vector multiplications instead (e.g. computing $\Delta\mathbf{X}_{\mathscr{B}^{(t)}} \mathbf{w}_U^{(t+1)}$ first and then multiplying by $\Delta\mathbf{X}_{\mathscr{B}^{(t)}}^T$).

Suppose that ΔB samples are removed from each mini-batch on average, then the time complexity of updating the model parameters in each iteration using Equation 14 will be $O(\Delta Bm + m^2)$ (recall that the dimension of $\mathbf{X}$ is $n \times m$). In contrast, the time complexity for retraining from scratch (i.e. not caching $\sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i\mathbf{x}_i^T$ in Equation 14) will be $O((B - \Delta B)m)$. Of course, performance predictions based on asymptotic complexity give only very rough guidance, and we conduct experiments for realistic assessments. Still, the bounds above suggest, for example, that for small ΔB and $m \ll B$ incremental deletions with PrIU work better than retraining (and our experiments verify this, see Section 6).

Typically, however, a smaller mini-batch size B is used. To deal with the case in which $m > B$, we notice that the rank of the intermediate result, $\sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i\mathbf{x}_i^T$ should be no more than B, thus smaller than m when B is smaller than m. This motivates us to reduce the dimension of the intermediate results using SVD, i.e. $\sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i\mathbf{x}_i^T = \mathbf{U}^{(t)}\mathbf{S}^{(t)}\mathbf{V}^{T,(t)}$, where $\mathbf{S}^{(t)}$ is a diagonal matrix whose diagonal elements represent the singular values, while $\mathbf{U}^{(t)}$ and $\mathbf{V}^{(t)}$ are the left and right singular vectors. Suppose after SVD, we only keep the r largest singular values and the corresponding singular vectors where $r \ll B$, then $\sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i\mathbf{x}_i^T$ is approximated by $\mathbf{U}_{1..r}^{(t)}\mathbf{S}_{1..r}^{(t)}\mathbf{V}_{1..r}^{(t)\,T}$ ($\mathbf{U}_{1..r}^{(t)}, \mathbf{V}_{1..r}^{(t)}$ represents the submatrix composed of the first r columns and $\mathbf{S}_{1..r}^{(t)}$ is a diagonal matrix composed of the first r eigenvalues in $\mathbf{S}^{(t)}$). Thus Equation 14 can be rewritten as:

$$\mathbf{w}_U^{(t+1)} \leftarrow [(1-\eta_t\lambda)\mathbf{I} - \frac{2\eta_t}{B_U^{(t)}}(\mathbf{U}_{1..r}^{(t)}\mathbf{S}_{1..r}^{(t)}\mathbf{V}_{1..r}^{T,(t)} - \Delta\mathbf{X}_{\mathscr{B}^{(t)}}^T \Delta\mathbf{X}_{\mathscr{B}^{(t)}})]\mathbf{w}_U^{(t)} + \frac{2\eta_t}{B_U^{(t)}}(\sum_{i \in \mathscr{B}^{(t)}} \mathbf{x}_i y_i - \sum_{\substack{i \in \mathscr{B}^{(t)} \\ ,\mathbf{x}_i \in \Delta\mathbf{X} \\ ,y_i \in \Delta\mathbf{Y}}} \mathbf{x}_i y_i) \tag{15}$$

Here we can cache the results of $\mathbf{U}_{1..r}^{(t)}\mathbf{S}_{1..r}^{(t)}$ (denoted by $\mathbf{P}_{1..r}^{(t)}$) and $\mathbf{V}_{1..r}^{(t)}$ for efficient updates, both of which have dimensions $m \times r$.

Time complexity. The time complexity to update the model parameters using this approach is $O(rm + \Delta Bm)$ for each iteration since the computation time is dominated by the matrix-vector computation, e.g. the multiplications of $\mathbf{V}_{1..r}^{T,(t)}$ and $\mathbf{w}_{U}^{(t)}$. This is more efficient than retraining from scratch, which has time complexity $O((B - \Delta B)m)$. So the total complexity for PrIU is $O(\tau rm + \tau \Delta B)$, where τ is the number of iterations in the training phase.

Space complexity. Using this approximation, at each iteration we only need to cache $\mathbf{P}_{1..r}^{(t)}$ and $\mathbf{V}_{1..r}^{(t)}$, which require space $O(rm)$. So the total space complexity will be $O(\tau rm)$ for τ iterations.

Theorem 6. ***Approximation ratio*** *Under the convergence conditions for $\boldsymbol{w}^{(t)}$, $||\boldsymbol{w}^{(t)}||$ should be bounded by some constant C. Suppose $\frac{||\boldsymbol{U}_{1..r}^{(t)}\boldsymbol{S}_{1..r}^{(t)}\boldsymbol{V}_{1..r}^{T,(t)}||_2}{||\boldsymbol{U}^{(t)}\boldsymbol{S}^{(t)}\boldsymbol{V}^{T,(t)}||_2} \geq 1 - \epsilon$ where ϵ is a small value, then the change of model parameters caused by the approximation will be bounded by $O(\epsilon)$.*

This shows that with proper choice of r in the SVD approximation, the updated model parameters computed by PrIU or PrIU-opt should be still very close to the expected result. So in our implementations, r is chosen based on ϵ (say 0.01) such that the inequality in Theorem 6 is satisfied.

5.2 PrIU-opt: Optimizations for linear regression

When the feature space is small, additional optimizations can be used for linear regression. Note that according to [27], the model parameters derived by both SGD and mb-SGD will end up with statistically the same results as GD. This means that the update rule in Equation 5 and Equation 14 could be approximated by its alternative using GD, i.e.:

$$\mathbf{w}^{(t+1)} \leftarrow ((1 - \eta_t\lambda)\mathbf{I} - \frac{2\eta_t}{n}\mathbf{X}^T\mathbf{X})\mathbf{w}^{(t)} + \frac{2\eta_t}{n}\mathbf{X}^T\mathbf{Y} \tag{16}$$

$$\begin{aligned}\mathbf{w}_{U}^{(t+1)} \leftarrow ((1 - \eta_t\lambda)\mathbf{I} - \frac{2\eta_t}{n - \Delta n}(\mathbf{X}^T\mathbf{X} - \Delta\mathbf{X}^T\Delta\mathbf{X}))\mathbf{w}_{U}^{(t)} \\ + \frac{2\eta_t}{n - \Delta n}(\mathbf{X}^T\mathbf{Y} - \Delta\mathbf{X}^T\Delta\mathbf{Y})\end{aligned} \tag{17}$$

in which $(\Delta\mathbf{X}, \Delta\mathbf{Y})$ represent the removed samples while Δn represents the number of those samples. Let $\mathbf{M}$ and $\mathbf{N}$ denote $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{Y}$ respectively. Then eigenvalue decomposition can be applied over $\mathbf{M}$, i.e. $\mathbf{M} = \mathbf{Q}\ diag\ (\{c_i\}_{i=1}^n)\ \mathbf{Q}^{-1}$ (where c_i represents the eigenvalues of $\mathbf{M}$). This is then plugged into Equation 16 and computed recursively, which results in the following formula:

$$\begin{aligned}\mathbf{w}^{(t+1)} = \mathbf{Q}\ diag\ (\{\Pi_{j=1}^t(1 - \eta_j\lambda - \frac{2\eta_j}{n}c_i)\}_{i=1}^n)\ \mathbf{Q}^{-1}\mathbf{w}^{(0)} \\ + \mathbf{Q}diag(\sum_{l=1}^{t-1}\eta_l\{\Pi_{j=l+1}^t(1 - \eta_j\lambda - \frac{2\eta_j}{n}c_i)\}_{i=1}^n)\ \mathbf{Q}^{-1}\frac{2\mathbf{N}}{n}\end{aligned} \tag{18}$$

This indicates that once the eigenvalues and eigenvectors of each $\mathbf{M}$ are given, we can derive $\mathbf{w}^{(t)}$ by simply computing the product, $\Pi_{j=1}^t(1 - \eta_j\lambda - \frac{2\eta_j}{n}c_i)$, and the sum of the product, $\sum_{l=1}^{t-1}\eta_l\Pi_{j=l+1}^t(1 - \eta_j\lambda - \frac{2\eta_j}{n}c_i)$, on diagonal entries. The overhead of this is only $O(\tau m)$ (recall that τ represents the total iteration number), and thus we avoid the repetitive matrix multiplication operations through the for-loops. Also, observe that $\mathbf{M}' = \mathbf{X}^T\mathbf{X} - \Delta\mathbf{X}^T\Delta\mathbf{X}$ can be regarded as a small change over $\mathbf{M}$ when Δn is small. Thus we can use the results on incremental updates over eigenvalues in [39], i.e. when the difference between the eigenvectors of $\mathbf{M}'$ and that of $\mathbf{M}$ is negligible then the eigenvalues of $\mathbf{M}'$ can be estimated as:

$$\mathbf{Q}^{-1}\mathbf{M}'\mathbf{Q} = diag(\{c_i'\}_{i=1}^n) \tag{19}$$

Here, c_i' represents the approximate i^{th} eigenvalue of $\mathbf{M}'$. It indicates that we can apply eigenvalue decomposition over $\mathbf{M}$ *offline* before the model incremental update phase and use Equation 19 to get the updated eigenvalues.

Time complexity. The time complexity for updating the model parameters is dominated by the computation of c_i', which is followed by the computation over each c_i' as Equation 18 does. These have time complexities $O(\min\{\Delta n, m\}m^2)$ and $O(\tau m)$, respectively. So the total time complexity is $O(\min\{\Delta n, m\}m^2) + O(\tau m)$, which can be more efficient than the closed-form solution (see experiments in Section 6).

Space complexity. The method avoids caching the provenance information at each iteration, and only requires caching Q, Q^{-1} and all the eigenvalues c_i, which takes space $O(m^2)$

Theorem 7. ***(Approximation ratio)*** *The approximation of PrIU-opt over the model parameters is bounded by $O(||\Delta\boldsymbol{X}^T\Delta\boldsymbol{X}||)$*

This shows that with small number of removed samples, the approximation ratio should be very small.

5.3 PrIU: Logistic regression

As the first step of the implementation of PrIU for logistic regression, non-linear operations are linearized using piecewise linear interpolation. Then, based on the analysis in Section 4, given the subset to be removed in dense datasets, $\Delta\mathbf{X}$, Equation 11 can be rewritten as follows:

$$\begin{aligned}\mathbf{w}_{LU}^{(t+1)} \leftarrow [(1 - \eta_t\lambda)\mathbf{I} + \frac{\eta_t}{B_U^{(t)}}(\mathbf{C}^{(t)} - \Delta\mathbf{C}^{(t)})]\mathbf{w}_{LU}^{(t)} \\ + \frac{\eta_t}{B_U^{(t)}}(\mathbf{D}^{(t)} - \Delta\mathbf{D}^{(t)})\end{aligned} \tag{20}$$

where $\mathbf{C}^{(t)}, \mathbf{D}^{(t)}, \Delta\mathbf{C}^{(t)}, \Delta\mathbf{D}^{(t)}$ are:

$$\mathbf{C}^{(t)} = \sum_{i \in \mathscr{B}^{(t)}} a^{i,(t)} \mathbf{x}_i \mathbf{x}_i^T, \mathbf{D}^{(t)} = \sum_{i \in \mathscr{B}^{(t)}} b^{i,(t)} y_i \mathbf{x}_i$$
$$\Delta\mathbf{C}^{(t)} = \sum_{\substack{\mathbf{x}_i \in \Delta\mathbf{X}, \\ i \in \mathscr{B}^{(t)}}} a^{i,(t)} \mathbf{x}_i \mathbf{x}_i^T, \Delta\mathbf{D}^{(t)} = \sum_{\substack{\mathbf{x}_i \in \Delta\mathbf{X}, \\ i \in \mathscr{B}^{(t)}}} b^{i,(t)} y_i \mathbf{x}_i \tag{21}$$

Similar to linear regression, the intermediate results $\mathbf{C}^{(t)}$ and $\mathbf{D}^{(t)}$ are cached and the dimension of $\mathbf{C}^{(t)}$ can be reduced by using SVD before the model update phase, which can happen offline. Suppose after SVD, $\mathbf{C}^{(t)} \approx \mathbf{P}_{1..r}^{(t)} \mathbf{V}_{1..r}^{T,(t)}$, in which $\mathbf{P}_{1..r}^{(t)}$ and $\mathbf{V}_{1..r}^{(t)}$ are two matrices with dimension $m \times r$. In the end, Equation 20 is modified as below for incremental model updates:

$$\mathbf{w}_{LU}^{(t+1)} \leftarrow [(1 - \eta_t \lambda)\mathbf{I} + \frac{\eta_t}{B_U^{(t)}} (\mathbf{P}_{1..r}^{(t)} \mathbf{V}_{1..r}^{T,(t)} - \Delta\mathbf{C}^{(t)})] \mathbf{w}_{LU}^{(t)} + \frac{\eta_t}{B_U^{(t)}} (\mathbf{D}^{(t)} - \Delta\mathbf{D}^{(t)}) \tag{22}$$

Time complexity. To apply Equation 22 in the model update phase, the computation of $\mathbf{P}_{1..r}^{(t)} \mathbf{V}_{1..r}^{T,(t)} \mathbf{w}_{LU}^{(t)}$ and $\Delta\mathbf{C}^{(t)} \mathbf{w}_{LU}^{(t)}$ become the major overhead, which have time complexity $O(rm)$ and $O(\Delta Bm)$, respectively. Suppose there are τ iterations in total, then the total time complexity is $O(\tau(rm + \Delta Bm))$. In comparison, the time complexity of retraining from scratch is $O(\tau((B - \Delta B)m + C_{non}m))$, where C_{non} represents the overhead of the non-linear operations. When $r \ll B$ and $\Delta B \ll B$, we can therefore expect PrIU to be more efficient than retraining from scratch.

Space complexity analysis Through this approximation, we need to cache $\mathbf{P}_{1..r}^{(t)}$ and $\mathbf{V}_{1..r}^{(t)}$ at each iteration, which requires $O(\tau rm)$ space in total. Plus, $O(n\lceil \frac{\tau B}{n} \rceil)$ extra space is necessary to cache the linear coefficients. So the total space complexity will be $O(\tau rm) + O(n\lceil \frac{\tau B}{n} \rceil)$.

Theorem 8. ***(Approximation ratio)*** *Similar to Theorem 6, the deviation caused by the SVD approximation will be bounded by $O(\epsilon)$, given the ratio $\frac{||\mathbf{P}_{1..r}^{(t)} \mathbf{V}_{1..r}^{T,(t)}||_2}{||\mathbf{P}^{(t)} \mathbf{V}^{T,(t)}||_2} \geq 1 - \epsilon$. So using Theorem 5, $||E(\mathbf{w}_{LU}^{(t)} - \mathbf{w}_{RU}^{(t)})||_2$ is bounded by $O(\Delta x) + O(\eta_t) + O(\epsilon)$.*

This indicates that $\mathbf{w}_{LU}^{(t)}$ should be very close to $\mathbf{w}_{RU}^{(t)}$ (similar to the discussion after Theorem 6).

Discussion Notice that for sparse datasets with large feature space, we can utilize the efficient sparse matrix operations by retraining from scratch. Also note that the intermediate result $\mathbf{C}^{(t)}$ will be a sparse matrix for such datasets. However, after SVD, there is no guarantee that $\mathbf{P}^{(t)}$ and $\mathbf{V}^{(t)}$ are sparse matrices. Therefore, for sparse training datasets, we will simply use the linearized update rule, i.e. Equation 11 directly, without considering the strategies outlined in this subsection.

5.4 PrIU-opt: Optimizations for logistic regression

Again, when the feature space is small additional optimizations are possible. In particular, we observe that for each sample i the change in the coefficients $a^{i,(t)}$ and $b^{i,(t)}$ from one iteration to the next becomes smaller and smaller as $\mathbf{w}^{(t)}$ converges. This suggests that we can stop capturing new provenance information at some earlier iteration, call it t_s, and continue with the same provenance until convergence. Suppose that for each sample i we approximate $a^{i,(t)}, b^{i,(t)}$ by $a^{i,*}$ and $b^{i,*}$ after iteration t_s. Therefore the matrices $\mathbf{C}^{(t)}, \mathbf{D}^{(t)}, \Delta\mathbf{C}^{(t)}$ and $\Delta\mathbf{D}^{(t)}$ will be approximated using the coefficients $a^{i,*}$ and $b^{i,*}$ and will remain the same for all iterations $t \geq t_s$ allowing us to avoid their recomputation. In the experiments, we found that a rule of thumb that takes t_s to be 70% of the total number of iterations works well.

This has the same form as for linear regression, motivating us to use the same techniques from PrIU-opt for linear regression, i.e. conducting eigenvalue decomposition over $\mathbf{C}^{(t)}$, followed by incrementally updating the eigenvalues given the changes $\Delta\mathbf{C}^{(t)}$, thus avoiding recomputations after iteration t_s.

Time complexity. Before iteration t_s, the total time complexity is $O(t_s(rm + \Delta Bm))$; after iteration t_s, the time complexity is $O(\min\{\Delta n, m\}m^2) + O((\tau - t_s)m)$ (see the time complexity analysis in Section 5.2). Thus the total time complexity is $O(t_s(rm+\Delta Bm))+O(\min\{\Delta n, m\}m^2)+O((\tau-t_s)m)$.

Space complexity. After iteration t_s, we only need to keep the eigenvectors of $\mathbf{C}^{(t)}$, which requires $O(m^2)$ space. Including the space overhead for the first t_s iterations, the total space complexity is $O(m^2) + O(t_s rm) + O(n\lceil \frac{t_s B}{n} \rceil)$.

Theorem 9. ***(Approximation ratio)*** *Suppose that after iteration t_s the gradient of the objective function is smaller than δ, then the approximations of PrIU-opt can lead to deviations of the model parameters bounded by $O((\tau - t_s)\delta) + O(||\Delta X^T \Delta X||)$. By combining the analysis in Theorem 5, $||E(\mathbf{w}_{LU}^{(t)} - \mathbf{w}_{RU}^{(t)})||_2$ is bounded by $O(\Delta x) + O(\eta_t) + O((\tau - t_s)\delta) + O(||\Delta X^T \Delta X||)$*

Similar to discussion after Theorem 7, $\mathbf{w}_{LU}^{(t)}$ should be very close to $\mathbf{w}_{RU}^{(t)}$.

Discussion. Our current framework handles linear and logistic models with L2 regularization. Our solutions cannot handle L1 regularization since in this case the gradient of the objective function is not continuous, thus invalidating some of the error bound analysis above. How to handle L1 regularization will be our future work.

6 EXPERIMENTS

6.1 Experimental setup

Platform. We conduct extensive experiments in Python 3.6 and use PyTorch 1.3.0 [40] for the experiments for dense datasets and scipy 1.3.1 [25] for the experiments for sparse datasets. All experiments were conducted on a Linux server with an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz and 64GB of main memory.

Datasets. Six datasets were used in our experiments: (1) the UCI SGEMM GPU dataset[3]; (2) the UCI Covtype dataset [4]; (3) the UCI HIGGS dataset [5]; (4) the RCV1 dataset [6] (5) the Kaggle ECG Heartbeat Categorization Dataset[7]; (6) the CIFAR-10 dataset [8], which are referenced as SGEMM, Cov, HIGGS, RCV1, Heartbeat and cifar10 hereafter.

SGEMM has continuous label values, therefore we use it in experiments with *linear regression* while the rest of them have values that are appropriate for classification. Each dataset is partitioned into *training* (90% of the samples) and *validation* (10% of the samples) datasets, the latter used for measuring the accuracy of models trained from the former.

The characteristics of these datasets are listed in Table 1, which indicates that RCV1 and cifar10 have extremely large feature space (over 30k model parameters) while other datasets have much fewer parameters (Heartbeat has around 1000 while others have less than 500).

6.2 Experiment design

We conduct two sets of experiments, the first of which aims to evaluate the performance of PrIU and PrIU-opt with respect to the deletion of *one subset* of the training samples. We do this over different types of datasets (dense VS sparse, large feature space VS small) with varied configurations (how many samples to be removed, mini-batch size, iteration numbers etc.), and compare against retraining from scratch. The second set of experiments simulate the scenario where users *repetitively* remove different subsets of training samples.

In the first set of experiments we simulate the cleaning scenario. To specify the samples to be removed from the training datasets, we introduce dirty samples, which are a selected subset of samples from the original dataset $\mathcal{T}$ that are modified to incorrect values by rescaling. The resulting dataset is denoted $\mathcal{T}_{\text{dirty}}$, over which the initial model $\mathcal{M}_{\text{init}}$ is constructed. The dirty samples are then removed in the model update phase. The goal is to compare the robustness of PrIU, PrIU-opt and the *influence function* [28] method when dirty data exists. In the experiment we vary the number of erroneous samples generated. The ratio between the erroneous samples and the original training dataset is called the **deletion rate**, and we give it values ranging from 0.0001 (i.e. 0.01%) to 0.2 (i.e. 20%).

In the second set of the experiments, we simulate the scenario in which users debug or interpret models by removing different subsets of samples, necessitating repeated incremental model update operations. We assume that the datasets are very large; to simulate this, we create three synthetic datasets $\mathcal{T}_{\text{cat}}$ by concatenating 4 copies of HIGGS, 20 copies of Cov and 130 copies of Heartbeat such that the total number of training samples is around 40 million, 11 million and 11 million, respectively, which are denoted HIGGS (extended), Cov (extended) and Heartbeat (extended), respectively. In the experiments, ten different subsets are removed and for each of them the deletion rate is about 0.1% of randomly picked samples out of the full training set. The hyperparameters for this set of experiments are listed in Table 2.

Baseline. For both $\mathcal{T}_{\text{dirty}}$ and $\mathcal{T}_{\text{cat}}$, we simulate what users (presumably unaware of errors) would do, and train an initial model $\mathcal{M}_{\text{init}}$ using the following standard method: Manually derive the formula for the gradient of the objective function and then program explicitly the GBM iterations. The erroneous or chosen samples are then removed from $\mathcal{T}_{\text{dirty}}$ or $\mathcal{T}_{\text{cat}}$. For linear regression (except for GBM), we also compare PrIU and PrIU-opt against close-form formula solutions for incremental updates [11, 20, 21, 38], denoted by Closed-form.

Incrementality. To update the model $\mathcal{M}_{\text{init}}$, the straightforward solution is to retrain from the scratch by using the same standard method as before but exclude the removed samples from each mini-batch. We denote this solution by BaseL. In contrast, our approach uses PrIU or PrIU-opt to incrementally update the model. The time taken by BaseL, PrIU or PrIU-opt to produce the updated model is reported in the experiments as the *update time*, and is compared over the two solutions: retraining with BaseL vs. incremental update with PrIU or PrIU-opt.

Note that our PrIU/ PrIU-opt approach uses provenance information collected from the whole training dataset. This phase is *offline* for the PrIU/ PrIU-opt algorithms and is *not* included in their reported running times. In practice, for the first set of experiments (cleaning of erroneous samples) provenance collection is done during the training of $\mathcal{M}_{\text{init}}$ from $\mathcal{T}_{\text{dirty}}$. For the second set of experiments (repeated deletions of subsets for debugging or interpretability) provenance collection is done during an initial training of $\mathcal{M}_{\text{init}}$ from the entire dataset $\mathcal{T}_{\text{cat}}$, which only needs to be done *once* even if many deletions of subsets are performed subsequently.

Since PrIU-opt is the optimized version for datasets with small feature space we only record the update time of PrIU over RCV1 and cifar10, which have very large feature spaces.

Accuracy. We compare the quality of the updated model obtained by BaseL and PrIU/PrIU-opt. The goal is to show that the improvement in update time is not achieved at the

[3]https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance
[4]https://archive.ics.uci.edu/ml/datasets/covertype
[5]https://archive.ics.uci.edu/ml/datasets/HIGGS
[6]simplified version from https://scikit-learn.org/0.18/datasets/rcv1.html
[7]https://www.kaggle.com/shayanfazeli/heartbeat
[8]https://www.cs.toronto.edu/~kriz/cifar.html

expense of accuracy. For experiments with linear regression, we use the *mean squared error (MSE)* over the validation datasets as a measure for accuracy. A lower MSE corresponds to higher accuracy over the validation set. For experiments with binary or multinomial logistic regression, we use the updated model to classify the samples in the validation datasets and report their *validation accuracy*.

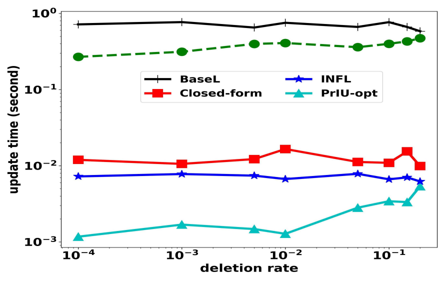
Model comparison. We also compare the updated models *structurally* by comparing the vector of updated model parameters obtained via PrIU/PrIU-opt against the ones by using BaseL. This is done in two different ways: 1) Using *distance*, that is, the *L2-norm* of the difference between the two vectors, for both linear and logistic regression, and 2) Using *similarity*, that is, the *cosine* of the angle between the two vectors. The latter is only done for logistic regression since the angle is only relevant for classification techniques. For both linear regression and logistic regression, we also record the changes of the signs and magnitude of individual coordinate of the updated model parameters by PrIU and PrIU-opt compared to the ones obtained by BaseL.

Comparison with influence function. As indicated in Section 2, the *influence function* method in [28] can be extended to handle the removal of multiple training samples by us (details omitted). We denote the resulting method INFL and compare it against PrIU/PrIU-opt in the experiments. We predicted and verified experimentally that this approach produces models with poor validation accuracy since the derivation of INFL relies on the approximation of the Taylor expansion, which can be inaccurate. We also notice that the Taylor expansion used in INFL involves the computation of the Hessian matrix, which is very expensive for datasets with extremely large feature space. So we did not run INFL over RCV1 and cifar10 in the experiments; the comparison between PrIU/PrIU-opt and INFL over other datasets is enough to show the benefits of our approaches.

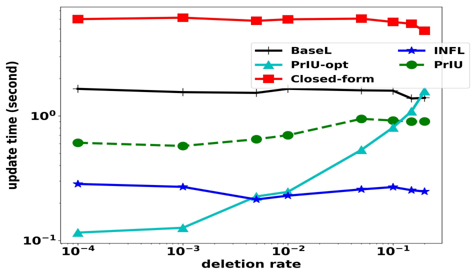
Effect of the hyperparameters and feature space size As discussed in Section 5, the performance of PrIU and PrIU-opt is influenced by the mini-batch size, the number of iterations and the size of the feature space. To explore the effect of the first two parameters for logistic regression, three different combinations of mini-batch size and number of iterations are used over Cov, denoted Cov (small), Cov (large 1) and Cov (large 2) (see Table 2). Since the datasets used for logistic regression have different feature space sizes, the performance difference with respect to feature space size can also be compared. Since there is only one dataset for linear regression, SGEMM, we extend this dataset by adding 1500 random features for each sample to determine the effect of feature space size. The extended version of SGEMM is denoted SGEMM (extended) (see Table 2). Other hyperparameters used in the experiments are shown in Table 2. Note that since erroneous samples exist in the training datasets for the first set of experiments, some values of the learning rate need to be very small to make sure that the convergence can be reached.

In the experiments, we answer the following questions:

- **(Q1)** Do the optimizations used in PrIU-opt compared to PrIU lead to a significant improvement in update time without sacrificing accuracy when the number of features in the training set is small?
- **(Q2)** Do PrIU and PrIU-opt afford significant gains in efficiency compared to BaseL?
- **(Q3)** Are the efficiency gains provided by PrIU and PrIU-opt achieved without sacrificing the accuracy of the updated model?
- **(Q4)** Can we experimentally validate the theoretical analysis in Sections 4.4 and 5, i.e. that the updated model derived through the approximations in PrIU and PrIU-opt is very close to the one obtained by BaseL?
- **(Q5)** Does the influence function approach, INFL, provide a competitive alternative to PrIU and PrIU-opt?
- **(Q6)** Can we experimentally show the effect of the hyperparameters, such as mini-batch size and iteration numbers over the performance gains of PrIU and PrIU-opt?
- **(Q7)** Can we experimentally show the effect of the feature space size (i.e. the number of model parameters, which equals to the feature number times the number of classes for multi-nomial logistic regression)?
- **(Q8)** What is the memory overhead of PrIU and PrIU-opt for caching the provenance information?

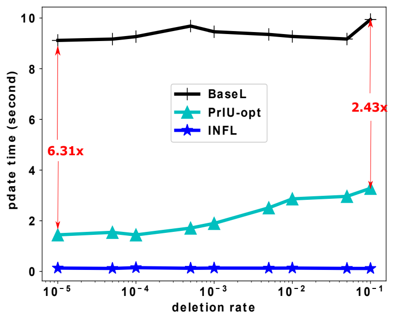


(a) SGEMM (original)

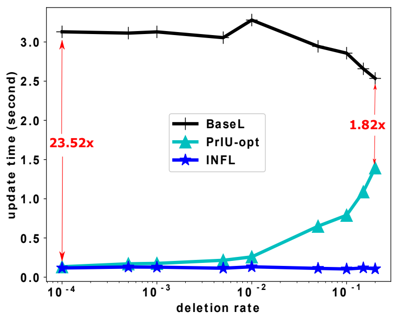


(b) SGEMM (extended)

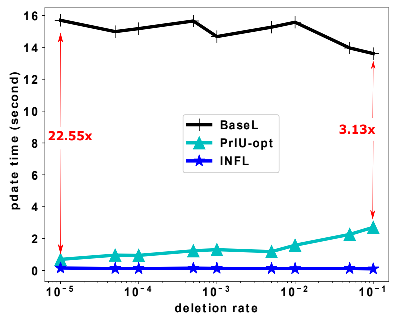
Figure 1: Update time using linear regression



(a) Cov (small)

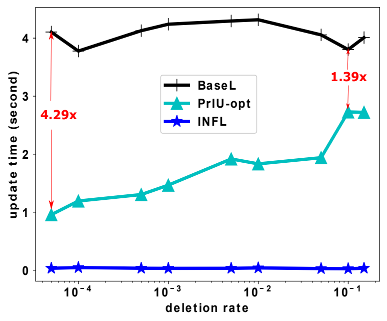


(b) Cov (large 1)

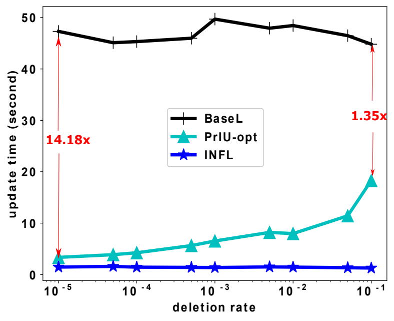


(c) Cov (large 2)

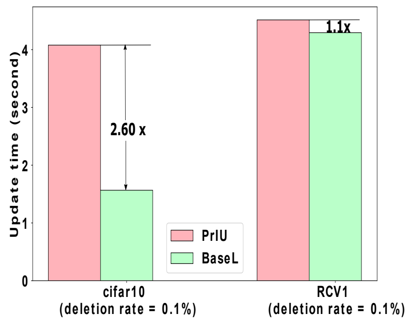
Figure 2: Update time using logistic regression over Cov and the hyperparameters from Table 2



(a) Heartbeat



(b) HIGGS



(c) RCV1 and cifar10

Figure 3: Update time using logistic regression

Table 1: Summary of datasets

name	# features	# classes	# samples
SGEMM	18		241,600
Cov	54	7	581,012
HIGGS	28	2	11,000,000
RCV1	47,236	2	23,149
Heartbeat	188	7	87,553
cifar10	3072	10	50,000

6.3 Experimental results

We report the results of our experiments in this subsection.

(Q1) We compare the update time of PrIU and PrIU-opt for linear regression using SGEMM (extended) in Figure 1b. The results show that the update time of PrIU-opt is significantly better than that of PrIU except when the deletion rate is approaching 20%. We also see from Table 4 that PrIU-opt and BaseL yield models that have exactly the same validation accuracy. Therefore, although PrIU-opt uses additional approximations for optimization, they do not hurt the predictive power of the updated models. This shows that the optimization strategies in Sections 5.2 and 5.4 are worth the design and implementation effort. Consequently, we will only compare PrIU-opt against other approaches except for cifar10 and RCV1 which have extremely large feature spaces.

Table 2: Summary of hyperparameters used in the experiments

name	mini-batch size	# of iterations	other hyper-parameters (η, λ)
SGEMM (original)	200	2000	$(5 \times 10^{-3}, 0.1)$
SGEMM (extended)	200	2000	$(5 \times 10^{-3}, 0.1)$
Cov (small)	200	10000	$(1 \times 10^{-4}, 0.001)$
Cov (large 1)	10000	500	$(1 \times 10e^{-4}, 0.001)$
Cov (large 2)	10000	3000	$(1 \times 10^{-4}, 0.001)$
HIGGS	2000	20000	$(1e^{-5}, 0.01)$
Cov (extended)	1000	40000	$(1 \times 10^{-4}, 0.001)$
HIGGS	2000	20000	$(1e^{-5}, 0.01)$
HIGGS (extended)	2000	60000	$(1 \times 10^{-5}, 0.01)$
Heartbeat	500	5000	$(1 \times 10^{-5}, 0.1)$
Heartbeat (extended)	500	40000	$(1 \times 10^{-5}, 0.1)$
RCV1	500	3000	$(1 \times 10^{-6}, 0.5)$
cifar10	500	1000	$(0.001, 0.1)$

Table 3: Memory consumption summary (GB)

Dataset	BaseL	PrIU	PrIU-opt
Cov (small)	0.71	4.30	4.34
Cov (large 1)	0.87	4.02	3.49
Cov (large 2)	1.34	21.0	17.4
HIGGS	5.09	8.40	8.40
SGEMM (original)	2.43	2.45	2.48
SGEMM (extended)	4.94	6.66	5.74
Heartbeat	0.46	6.01	5.69
RCV1	0.28	0.3	-
cifar10	0.79	26.59	-

(Q2) Figures 1a-1b compare the update time in BaseL and PrIU-opt using linear regression (ignore the INFL lines for the moment), while Figures 2-3 show the same results for logistic regression for single model update operation. Observe that for both linear and logistic regression, when the deletion rate is small (<0.01), PrIU-opt can achieve significant speed-up compared to BaseL: up to two orders of magnitude for linear regression and up to around 23x for logistic regression (for Cov (large 1) and Cov (large 2) with low deletion rate). Even when the feature spaces are extremely large, with deletion rate 0.1%, there is around a 2.6x speed-up for dense datasets (cifar10 in Figure 3c) and only 10% for sparse datasets (RCV1 in Figure 3c), respectively (similar speed-ups were observed for other small deletion rates). The former shows the effectiveness of the optimization strategies in PrIU over dense datasets with a large feature space while the latter is due to the fact that the optimization strategies for dense datasets were not applied over the sparse ones. Notice that for linear regression, PrIU-opt is always faster than Closed-form. Figure 4 shows the results of repetitive model updates; PrIU-opt achieves an order of magnitude speed-up for HIGGS (extended).

(Q3) Table 4 (validation accuracy for PrIU and PrIU-opt column) compares the quality of the models obtained by PrIU/PrIU-opt with that of the models obtained by BaseL. For these results we chose the highest deletion rate in the experiments, i.e. 20%. For all the experiments, the validation accuracy figures (MSE in the case of linear regression) of the updated models obtained by PrIU and PrIU-opt *match exactly* the accuracy of the ones obtained by BaseL. Combined with the answer to **Q2**, we can conclude that *PrIU-opt speeds up the model update time by up to two orders of magnitude without sacrificing any validation accuracy.*

(Q4) We investigate why PrIU-opt has the same validation accuracy as BaseL by measuring the distance and similarity between the updated models computed by PrIU-opt and BaseL. The results are presented in Table 4 (again, ignore the columns for INFL). The results indicate that the updated model parameters computed by PrIU-opt are very close to the ones obtained by BaseL since the cosine similarity is almost 1 (see the “similarity” column) while the L2-dist is very small (see the “distance” column). An even finer-grained analysis, comparing the signs and magnitude of each coordinate in the model parameters updated by PrIU-opt and BaseL shows that there is no sign flipping and only negligible magnitude changes for PrIU-opt compared to BaseL when the deletion rate is small. Even with a large deletion rate of 20% in HIGGS, only 2 out of 58 coordinates flip their signs with small magnitude change.

(Q5) The model update time of INFL is also included in Figures 2 and 3. Note that it can be up to one order of magnitude better than PrIU-opt, which is expected since using INFL to update the model parameters does not require an iterative computation. However, there is a significant drop in validation accuracy of the updated model derived by INFL compared to BaseL and PrIU-opt (see Table 4), which is due to the significantly higher L2-dist (see the “distance” column) and lower cosine similarity (see the “similarity” column) of its updated model compared to the model derived by BaseL. We conclude that PrIU and PrIU-opt produce much better models than INFL yet can still achieve comparable speed-ups.

(Q6) Effect of mini-batch size. The effect of mini-batch size is seen by comparing Cov (large 1) and Cov (small). One observation is that with larger mini-batch size, the maximal speed-up of PrIU-opt is around 23x, while with the smaller mini-batch size it is only about 6x, see Figures 2a and 2b This confirms the analysis in Section 5. In the second set of experiments, we used a small mini-batch size for Cov (1000) and Heartbeat (500), resulting in only 4.62x and 3.2x speed-ups by PrIU-opt, respectively (see Figure 4).

Effect of number of iterations. A comparison of Cov (large 1) and Cov (large 2), which have the same mini-batch size but a different number of iterations, can be found in Figures 2b and 2c. We observe that no matter how many iterations the program runs for, at the same deletion rate PrIU-opt achieves a similar speed-up against BaseL. For example, we have up to around 23x speed-up for small deletion rates and smaller speed-up for higher deletion rates (note the difference in y-axis scale between Figures 2b and 2c). However, increasing the number of iterations increases the amount of provenance information cached for PrIU-opt, thus requiring more memory. As Table 3 indicates, since there are 6x iterations for Cov (large 2) compared to Cov (large 1), roughly 6x memory is needed, confirming the analysis in Section 5. However for Cov, with a large mini-batch size and 500 iterations, convergence is achieved and we do not observe a difference in validation accuracy between Cov (large 1) and Cov (large 2). Note that according to [9, 35, 42], the theoretical optimal number of passes for logistic regression using mb-SGD (one pass equals to the total number of iterations divided by the number of iterations used for going

Table 4: Accuracy and similarity comparison between PrIU-opt and INFL with deletion rate 0.2

Dataset	Validation accuracy		distance		similarity	
	BaseL = PrIU-opt	INFL	PrIU-opt	INFL	PrIU-opt	INFL
Cov (small)	48.76%	36.93%	0.184	1.287	0.992	0.624
Cov (large 1)	48.76%	37.99%	0.0016	1.047	1.0	0.738
Cov (large 2)	48.76%	46.38%	0.0003	1.430	1.0	0.471
HIGGS	52.99%	47.99%	0.0004	0.006	0.979	-0.040
Heartbeat	82.78%	74.34%	0.0016	0.583	1.00	0.143
SGEMM (origin)	0.001	0.002	0.027	0.140	-	-
SGEMM (extended)	0.001	0.002	0.029	0.141	-	-

through the full training set) is quite small. However, for Cov (large 2) the number of passes over the full training set is quite large ($3000/(581012/10000) \approx 60$). Such a high memory usage should therefore not arise in practice.

(Q7) In terms of the update time for experiments over datasets with a comparable mini-batch size but with different feature space sizes (Heartbeat VS HIGGS), we notice that a larger number of model parameters leads to poorer performance by PrIU-opt (compare Figures 3a and 3b). This is also validated through a second set of experiments in which HIGGS (extended) achieves significant speed-up compared to Heartbeat (extended) (see Figure 4). This confirms the analysis in Section 5, where we show how the asymptotic execution time of PrIU and PrIU-opt depends on the number of the model parameters.

(Q8) Table 3 shows that in most cases, both PrIU and PrIU-opt only consume no more than 5x memory compared to BaseL (ignore the number for Cov (large 2) since, as discussed earlier, it is a rare case in practice). However, with a large number of model parameters (like cifar10 and Heartbeat) there is over 10x memory consumption for PrIU and PrIU-opt. How to decrease the memory usage for dense datasets with large feature space is left for future work.

Discussion. Extensive experiments using linear regression and logistic regression over the datasets above show the feasibility of our approach. PrIU and PrIU-opt can achieve up to two orders of magnitude speed-up for incrementally updating model parameters compared to the baseline, especially for large datasets with a small feature space. This is done without sacrificing the correctness of the results (measured by similarity to the updated model parameters

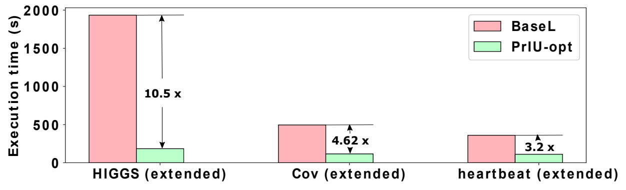


Figure 4: The execution time of repetitively removing 10 different subsets

by BaseL) and the prediction performance. The experiments also show that the optimizations used in PrIU-opt give significant performance gains compared to PrIU with only a small loss of accuracy. We observe that INFL is not a good solution because of the poor quality of models produced when more than one sample is removed.

Limitations. Our experiments also show the limitations of our solutions. They concern the memory footprint when the feature space or the number of iterations are large (anticipated by several analyses in Section 5) and the marginal speed-up for large sparse datasets (discussed at the end of Section 5.3). We shall endeavor to approach these limitations in future work.

7 CONCLUSIONS

In this paper, we build a connection between data provenance and incremental machine learning model updates, which is useful in many machine learning and data science applications such as data cleaning and interpretability. Building on an extension of the provenance semiring framework [19] to include basic linear algebra operations [50], we capture provenance in the training phase of linear regression and (binary and multinomial) logistic regression. This enables model parameters to be efficiently updated after a subset of training samples are removed. We assume that gradient descent and its variants (i.e. stochastic gradient descent and mini-batch stochastic gradient descent) are used, and address non-linear operations in logistic regression using piecewise linear interpolation. We prove that linearization does not harm convergence of the updated parameters and similarity to the expected results. Based on these theoretical results, we construct solutions, PrIU and PrIU-opt, which are optimized to reduce the time and space overhead. The benefits of our solutions are experimentally verified through extensive evaluations over various datasets. Looking forward, we believe that these solutions for simpler machine learning models are likely to extend to *generalized additive models* [22] and they also pave the way toward solutions for more complicated machine learning models such as deep neural networks.

REFERENCES

[1] Yael Amsterdamer, Susan B Davidson, Daniel Deutch, Tova Milo, Julia Stoyanovich, and Val Tannen. 2011. Putting lipstick on pig: Enabling database-style workflow provenance. *Proceedings of the VLDB Endowment* 5, 4 (2011), 346–357.

[2] Yael Amsterdamer, Daniel Deutch, and Val Tannen. 2011. Provenance for aggregate queries. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 153–164.

[3] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. 2001. Why and where: A characterization of data provenance. In *International conference on database theory*. Springer, 316–330.

[4] Peter Buneman and Wang-Chiew Tan. 2018. Data Provenance: What next? *ACM SIGMOD Record* 47, 3 (2018), 5–16.

[5] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 1721–1730.

[6] James Cheney, Laura Chiticariu, and Wang Chiew Tan. 2009. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases* 1, 4 (2009), 379–474.

[7] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2201–2206.

[8] R Dennis Cook. 1977. Detection of influential observation in linear regression. *Technometrics* 19, 1 (1977), 15–18.

[9] John Darzentas. 1984. Problem complexity and method efficiency in optimization. *Journal of the Operational Research Society* 35, 5 (1984), 455–455.

[10] Tamraparni Dasu and Theodore Johnson. 2003. *Exploratory data mining and data cleaning*. Vol. 479. John Wiley & Sons.

[11] Amol Deshpande and Samuel Madden. 2006. MauveDB: supporting model-based user views in database systems. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. ACM, 73–84.

[12] Mohamad Dolatshah, Mathew Teoh, Jiannan Wang, and Jian Pei. 2018. Cleaning crowdsourced labels using oracles for statistical classification. *Proceedings of the VLDB Endowment* 12, 4 (2018), 376–389.

[13] Finale Doshi-Velez and Been Kim. 2017. A roadmap for a rigorous science of interpretability. *arXiv preprint arXiv:1702.08608* 150 (2017).

[14] Tommy Ellkvist, David Koop, Erik W Anderson, Juliana Freire, and Cláudio Silva. 2008. Using provenance to support real-time collaborative design of workflows. In *International Provenance and Annotation Workshop*. Springer, 266–279.

[15] Wenfei Fan and Floris Geerts. 2012. *Foundations of Data Quality Management*. Morgan & Claypool Publishers.

[16] Todd J Green, Grigoris Karvounarakis, Zachary G Ives, and Val Tannen. 2007. Update exchange with mappings and provenance. In *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 675–686.

[17] Todd J Green, Grigoris Karvounarakis, Zachary G Ives, and Val Tannen. 2010. Provenance in ORCHESTRA. (2010).

[18] Todd J Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 31–40.

[19] Todd J. Green and Val Tannen. 2017. The Semiring Framework for Database Provenance. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*. 93–99.

[20] Priyank Gupta, Nick Koudas, Europa Shang, Ryan Johnson, and Calisto Zuzarte. 2015. Processing analytical workloads incrementally. *arXiv preprint arXiv:1509.05066* (2015).

[21] Sona Hasani, Saravanan Thirumuruganathan, Abolfazl Asudeh, Nick Koudas, and Gautam Das. 2018. Efficient construction of approximate ad-hoc ML models through materialization and reuse. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1468–1481.

[22] Trevor Hastie and Robert Tibshirani. 1986. Generalized additive models. *Statist. Sci.* 1, 3 (1986), 297–318.

[23] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-Shot Learning for Error Detection. *arXiv preprint arXiv:1904.02285* (2019).

[24] Zachary G Ives, Todd J Green, Grigoris Karvounarakis, Nicholas E Taylor, Val Tannen, Partha Pratim Talukdar, Marie Jacob, and Fernando Pereira. 2008. The ORCHESTRA collaborative data sharing system. *ACM Sigmod Record* 37, 3 (2008), 26–32.

[25] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. http://www.scipy.org/

[26] Daniel Kang, Deepti Raghavan, Peter Bailis, and Matei Zaharia. 2018. Model Assertions for Debugging Machine Learning. https://www-cs.stanford.edu/~matei/papers/2018/mlsys_model_assertions.pdf Preprint.

[27] Hamed Karimi, Julie Nutini, and Mark Schmidt. 2016. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 795–811.

[28] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1885–1894.

[29] Rainer Kress. 1998. *Interpolation*. Springer New York, New York, NY, 151–188. https://doi.org/10.1007/978-1-4612-0599-9_8

[30] Sanjay Krishnan, Michael J Franklin, Ken Goldberg, and Eugene Wu. 2017. Boostclean: Automated error detection and repair for machine learning. *arXiv preprint arXiv:1711.01299* (2017).

[31] Sanjay Krishnan, Daniel Haas, Michael J Franklin, and Eugene Wu. 2016. Towards reliable interactive data cleaning: a user survey and recommendations. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM, 9.

[32] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. 2016. ActiveClean: interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment* 9, 12 (2016), 948–959.

[33] Sanjay Krishnan and Eugene Wu. 2017. Palm: Machine learning explanations for iterative debugging. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*. ACM, 4.

[34] Raunak Kumar and Mark Schmidt. 2017. Convergence rate of expectation-maximization. In *10th NIPS Workshop on Optimization for Machine Learning*.

[35] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer, 9–48.

[36] Zachary Lipton. 2016. The Mythos of Model Interpretability. *CoRR* abs/1606.03490 (2016).

[37] Yin Lou, Rich Caruana, and Johannes Gehrke. 2012. Intelligible models for classification and regression. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. 150–158.

[38] Milos Nikolic, Mohammed Elseidy, and Christoph Koch. 2014. LINVIEW: incremental view maintenance for complex analytical queries. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*. 253–264.

[39] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S Huang. 2010. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition* 43, 1 (2010), 113–127.

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[41] Boris Teodorovich Polyak. 1963. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 3, 4 (1963), 643–653.

[42] Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization* 30, 4 (1992), 838–855.

[43] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2017. Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1723–1726.

[44] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Vaughan, and Hanna Wallach. 2018. Manipulating and measuring model interpretability. *arXiv preprint arXiv:1802.07810* (2018).

[45] Erhard Rahm and Hong Hai Do. 2000. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.* 23, 4 (2000), 3–13.

[46] Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.

[47] Mark Schmidt. 2014. Convergence rate of stochastic gradient with constant step size. (2014).

[48] Jennifer She and Mark Schmidt. 2017. Linear convergence and support vector identifiation of sequential minimal optimization. In *10th NIPS Workshop on Optimization for Machine Learning*. 5.

[49] Alan Weiser and Sergio E Zarantonello. 1988. A note on piecewise linear and multilinear table interpolation in many dimensions. *Math. Comp.* 50, 181 (1988), 189–196.

[50] Zhepeng Yan, Val Tannen, and Zachary G Ives. 2016. Fine-grained Provenance for Linear Algebra Operators.. In *TaPP*.

[51] Wenchao Zhou, Micah Sherr, Tao Tao, Xiaozhou Li, Boon Thau Loo, and Yun Mao. 2010. Efficient querying and maintenance of network provenance at internet-scale. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 615–626.