ActiveClean: Interactive Data Cleaning For Statistical Modeling

Sanjay Krishnan, Jiannan Wang[†], Eugene Wu[††], Michael J. Franklin, Ken Goldberg
UC Berkeley, [†]Simon Fraser University, [††]Columbia University
{sanjaykrishnan, franklin, goldberg}@berkeley.edu jnwang@sfu.ca ewu@cs.columbia.edu

ABSTRACT

Analysts often clean dirty data iteratively–cleaning some data, executing the analysis, and then cleaning more data based on the results. We explore the iterative cleaning process in the context of statistical model training, which is an increasingly popular form of data analytics. We propose ActiveClean, which allows for progressive and iterative cleaning in statistical modeling problems while preserving convergence guarantees. ActiveClean supports an important class of models called convex loss models (e.g., linear regression and SVMs), and prioritizes cleaning those records likely to affect the results. We evaluate ActiveClean on five real-world datasets UCI Adult, UCI EEG, MNIST, IMDB, and Dollars For Docs with both real and synthetic errors. The results show that our proposed optimizations can improve model accuracy by up-to 2.5x for the same amount of data cleaned. Furthermore for a fixed cleaning budget and on all real dirty datasets, ActiveClean returns more accurate models than uniform sampling and Active Learning.

1. INTRODUCTION

Statistical models trained on historical data facilitate several important predictive applications such as fraud detection, recommendation systems, and automatic content classification. In a survey of Apache Spark users, over 60% responded that support for advanced statistical analytics was Spark's most important feature [1]. This sentiment is echoed across both industry and academia, and there has been significant interest in improving the efficiency of model training pipelines. Although it is often overlooked, an important step in all model training pipelines is handling dirty or inconsistent data including extracting structure, imputing missing values, and handling incorrect data. Analysts widely report that cleaning dirty data is a major concern [21], and consequently, it is important to understand the efficiency and correctness of such operations in the context of emerging statistical analytics.

While many aspects of the data cleaning problem have been well-studied for SQL analytics, the results can be counter-intuitive in high-dimensional statistical models. For example, studies have shown that many analysts do not approach cleaning as a one-shot pre-processing step, and instead, repeatedly alternate between cleaning and analysis. It is common to use the preliminary analysis on dirty data as a guide to help identify potential errors and design repairs [15,21,22]. Unfortunately, for statistical models, iteratively cleaning some data and re-training on a partially clean dataset can lead to biases in even the simplest models.

Figure 1: (a) Systematic corruption in one variable can lead to a shifted model. The dirty examples are labeled 1-5 and the cleaned examples are labeled 1'-5'. (b) Mixed dirty and clean data results in a less accurate model than no cleaning. (c) Small samples of only clean data can result in similarly issues.

Consider a linear regression model on systematically translated data (Figure 1a). If one only cleans two of the data points, the intermediate result reveals a misleading trend (Figure 1b). This is a consequence of the well-known Simpson's paradox where aggregates over different populations of data can result in spurious relationships [32]. Similarly, statistical models face more dramatic sampling effects than the traditional 1D `sum`, `count`, `avg` SQL aggregates (Figure 1c). The challenges with Simpson's paradox and sampling are problematic because recent advances in SQL data cleaning, such as Sample-and-Clean [33] and Progressive Data Cleaning [6,28,36], actually advocate cleaning subsets of data to avoid the potentially expensive cleaning costs. Clearly, such data cleaning approaches will have to be re-evaluated for the statistical modeling setting, and this paper explores how to adapt such approaches with guarantees of convergence for an important class of modeling problems.

Data cleaning is a broad area that encompasses extraction, deduplication, schema matching, and many other problems in relational data. We focus on two common operations that often require iterative cleaning: removing outliers and attribute transformation. For example, battery-powered sensors can transmit inaccurate measurements when battery levels are low [20]. Similarly, data entered by humans can be susceptible to a variety of inconsistencies (e.g., typos), and unintentional cognitive biases [23]. Since these two types of errors do not affect the schema or leave any obvious signs of corruption (e.g., NULL values), model training may seemingly succeed–albeit with an inaccurate result.

Proceedings of the VLDB Endowment, Vol. 9, No. 12

We propose ActiveClean, a model training framework that allows for iterative data cleaning while preserving provable convergence properties. The analyst initializes ActiveClean with a model, a featurization function, and a pointer to a dirty relational table. In each iteration, ActiveClean suggests a sample of data to clean based on the data's value to the model and the likelihood that it is actually dirty. The analyst can apply value transformations and filtering operations to the sample. ActiveClean will incrementally and safely update the current model (as opposed to complete re-training). We propose several novel optimizations that leverage information from the model to guide data cleaning towards the records most likely to be dirty and most likely to affect the results.

From a statistical perspective, our key insight is to treat the cleaning and training iteration as a form of Stochastic Gradient Descent, an iterative optimization method. We treat the dirty model as an initialization, and incrementally take gradient steps (cleaning a sample of records) towards the global solution (i.e., the clean model). Our algorithm ensures global convergence with a provable rate for an important class of models called *convex*-loss models which include SVMs, Linear Regression, and Logistic Regression. Convexity is a property that ensures that the iterative optimization converges to a true global optimum, and we can apply convergence arguments from convex optimization theory to show that ActiveClean converges.

To summarize our contributions:

- We propose ActiveClean, which allows for progressive data cleaning and statistical model training with guarantees.
- **Correctness** We show how to update a dirty model given newly cleaned data. This update converges monotonically in expectation with a with rate $O(\frac{1}{\sqrt{T}})$.
- **Optimizations** We derive a theoretically optimal sampling distribution that minimizes the update error and an approximation to estimate the theoretical optimum. We further show that ActiveClean can integrate with existing dirty data detection techniques. Our proposed optimizations can improve model accuracy by up-to 2.5x for the same amount of data cleaned.
- **Experiments** The experiments evaluate ActiveClean on five datasets with real and synthetic corruption. In a fraud prediction example, ActiveClean examines nearly 10x less records than alternatives to achieve an 80% true positive rate.

The correctness of many of the components detailed proofs, which we have included in our extended technical report [24].

2. ITERATIVE DATA CLEANING

This section introduces the problem of iterative data cleaning through an example application.

2.1 Use Case: Dollars for Docs

ProPublica collected a dataset of corporate donations to medical researchers to analyze conflicts of interest [4]. For reference, the dataset has the following schema:

```
Contribution(
 pi_specialty text, # PI's medical specialty
 drug_nm text ,  # drug brand name, null if not drug
 device_nm text, # device brand name, null if not a device
 corp text,      # name of pharamceutical donor
 amount float,   # amount donated
 dispute bool,   # whether the research is disputed
 status text     # if the donation is allowed
                 # under research protocol
)
```

The dataset comes with a `status` field that describes whether or not the donation was allowed under the declared research protocol. Unfortunately the dataset is dirty, and there are inconsistent ways to specify "allowed" or "disallowed". The ProPublica team identified the suspicious donations via extensive manual review (documented in [2]). However, new donation datasets are continuously released and would need to be analyzed by hand. Thus, let us consider an alternative ActiveClean-based approach to train a model to predict the true `status` value. This is necessary for several reasons: 1) training a model on the raw data would not work because the `status` field is often inconsistent and incorrect; 2) techniques such as active learning are only designed for acquiring *new* labels and not suitable for finding and fixing *incorrect* labels; and 3) other attributes such as company name may also be inconsistent (e.g., Pfizer Inc., Pfizer Incorporated, Pfizer) and need to be canonicalized before they are useful for training the model.

During our analysis, we found that nearly 40,000 of the 250,000 records had some form of inconsistency. Indeed, these errors were structural rather than random—disallowed donations were more likely to have an incorrect `status` value. In addition, larger pharmaceutical companies were more likely to have inconsistent names, and more likely to make disallowed donations. Without cleaning company names, a model would miss-predict many large pharmaceutical donations. In fact, we found that the true positive rate for predicting disallowed donations when training an SVM model on the raw data was only 66%. In contrast, cleaning the entire dataset improves this rate to 97% (Section 7.2.2), and we show in the experiments that ActiveClean can achieve comparable model accuracy (¡ 1% of the true model accuracy) while expending only 10% of the data cleaning effort. The rest of this section will introduce the key challenges in designing a Machine-Learning-oriented iterative data cleaning framework.

2.2 Iteration in Model Construction

Consider an analyst designing a classifier for this dataset. When she first develops her model on the dirty data, she will find that the detection rate (true positives predicted) is quite low 66%. To investigate why she might examine those records that are incorrectly predicted by the classifier.

It is common for analysts to use the preliminary analysis on dirty data as a guide to help identify potential errors and design repairs [21]. For example, our analyst may discover that there are numerous examples where two records are nearly identical, but one is predicted correctly, and one is incorrect, and their only difference is the `corporation` attribute: Pfizer and Pfizer Incorporated. Upon discovering such inconsistencies, she will merge those two attribute values, re-train the model, and repeat this process.

We define iterative data cleaning to be the process of cleaning subsets of data, evaluating preliminary results, and then cleaning more data as necessary. ActiveClean explores two key questions about this iterative process: (1) *Correctness.* Will this clean-retrain loop converge to the intended result and (2) *Efficiency.* How can we best make use of the existing data and analyst effort.

Correctness: The straight-forward application of data cleaning is to repair the corruption in-place, and re-train the model after each repair. However, this process has a crucial flaw, where a model is trained on a mix of clean and dirty data. It is known that aggregates over mixtures of different populations of data can result in spurious relationships due to the well-known phenomenon called Simpson's paradox [32]. Simpson's paradox is by no means a corner case, and it has affected the validity of a number of high-profile studies [29]. Figure 1 illustrates a simple example where such a process can lead to unreliable results, where artificial trends introduced by the mix-

ture can be confused for the effects of data cleaning. The consequence is that after applying a data cleaning operation on a subset of data the analyst cannot be sure if it makes the model more or less accurate. ActiveClean provides an update algorithm with a monotone convergence guarantee where more cleaning is leads to a more accurate result in expectation.

Efficiency: One could alternatively avoid the mixing problem by taking a small sample of data up-front, perfectly cleaning it, and then training a model. This approach is similar to SampleClean [33], which was proposed to approximate the results of aggregate queries by applying them to a clean sample of data. However, high-dimensional models are highly sensitive to sample size, and it is not uncommon to have models whose training data complexity is exponential in the dimensionality (i.e., the curse of dimensionality). Figure 1c illustrates that, even in two dimensions, models trained from small samples can be as incorrect as the mixing solution described before. Sampling further has a problem of scarcity, where errors that are rare may not show up in the sample. *ActiveClean uses a model trained on the dirty data as an initialization and uses this model as guide to identify future data to clean.*

Comparison to Active Learning: The motivation of ActiveClean similar to that of Active Learning [17,36] as both seek to reduce the number of queries to an analyst or a crowd. However, active learning addresses a fundamentally different problem than ActiveClean. Active Learning poses the problem of iteratively selecting the most informative *unlabeled* examples to label in partially labeled datasets—these examples are labeled by an expert and integrated into the machine learning model. Note that active learning does not handle cases where the dataset is labelled incorrectly but only cases where labels are missing. In contrast, ActiveClean studies the problem of prioritizing modifications to both features and labels in existing examples. In essence, ActiveClean handles incorrect values in *any* part (label or feature) of an example. This property changes the type of analysis and algorithms that can be used. Consequently, our experiments find that general active learning approaches (e.g., uncertainty sampling [31]) converge slower than ActiveClean. If the only form of data error was missing labels, then we would expect active learning to perform comparably or better than ActiveClean.

3. PROBLEM FORMALIZATION

ActiveClean is an iterative framework for cleaning data in support of statistical modeling. Analysts clean batches of data, which are fed back into the system to intelligently re-train the model, and recommend a next batch of data to clean. We will first introduce the terms used in the rest of the paper, describe our assumptions, and define the two problems that we solve in this paper.

3.1 Assumptions

In this paper, we use the term *statistical modeling* to describe a well-studied class of analytics problems; ones that can be expressed as the minimization of convex loss functions. Examples include linear models (including linear and logistic regression), support vector machines, and in fact, means and medians are also special cases. This class is restricted to supervised Machine Learning, and the result of the minimization is a vector of parameters θ. We further assume that there is a one-to-one mapping between records in a relation R and labeled training examples (x_i, y_i).

ActiveClean considers data cleaning operations that are applied record-by-record. That is, the data cleaning can be represented as a user-defined function $C(\cdot)$ that when applied to a record r and can perform two actions: recover a unique clean record $r' = C(r)$ with the same schema or remove the record $\emptyset = C(r)$. ActiveClean is agnostic to how $C(\cdot)$ is implemented, e.g., with software or a manual action by the analyst. We define the clean relation as a relation of all of the records after cleaning:

$$R_{clean} = \cup_i^N C(r_i \in R)$$

Therefore, for every $r' \in R_{clean}$ there exists a unique $r \in R$ in the dirty data. Supported cleaning operations include merging common inconsistencies (e.g., merging "U.S.A" and "United States"), filtering outliers (e.g., removing records with values $> 1e6$), and standardizing attribute semantics (e.g., "1.2 miles" and "1.93 km"). Our technical report discusses a generalization of this basic data cleaning model called the "set of records" cleaning model [24]. In this generalization, the $C(\cdot)$ function is composed of schema preserving map and filter operations applied to the entire dataset. This can model problems such batch merging of inconsistencies with a "find-and-replace". We acknowledge that both definitions of data cleaning are limited as they do *not* cover errors that simultaneously affect multiple records such as record duplication or structure such as schema transformation.

3.2 Notation

The user provides a pointer to a dirty relation R, a cleaner $C(\cdot)$, a featurizer $F(\cdot)$, and a convex loss problem. A total of k records will be cleaned in batches of size b, so there will be $T = \frac{k}{b}$ iterations of the algorithm. We use the following notation to represent relevant quantities:

Dirty Model: $\theta^{(d)}$ is the model trained on R (without cleaning).

Dirty Records: $R_{dirty} \subseteq R$ is the subset of records that are still dirty. As more data are cleaned $R_{dirty} \rightarrow \{\}$.

Clean Records: $R_{clean} \subseteq R$ is the subset of records that are clean, i.e., the complement of R_{dirty}.

Batches: S is a batch of data (possibly selected stochastically but with known probabilities) from the records R_{dirty}. The clean batch is denoted by $S_{clean} = C(S)$.

Clean Model: $\theta^{(c)}$ is the optimal clean model, i.e., the model trained on a fully cleaned relation. Accuracy and convergence are always with respect to $\theta^{(c)}$.

Current Model: $\theta^{(t)}$ is the current best model at iteration $t \in \{1, ..., \frac{k}{b}\}$, and $\theta^{(0)} = \theta^{(d)}$.

3.3 System Architecture

The main insight of ActiveClean is to model the interactive data cleaning problem as Stochastic Gradient Descent (SGD) [9]. SGD is an iterative optimization algorithm that starts with an initial estimate and then takes a sequence of steps "downhill" to minimize an objective function. Similarly, in interactive data cleaning, the human starts with a dirty model and makes a series of cleaning decisions to improve the accuracy of the model. We formalize the link between these two processes, and since SGD is one of the most widely studied forms of optimization, it has well understood theoretical convergence conditions. These theoretical properties give us clear restrictions on different components. Figure 2 illustrates the ActiveClean architecture including the: *Sampler*, *Cleaner*, *Updater*, *Detector*, and *Estimator*.

The first step of ActiveClean is initialization. We first initialize $R_{dirty} = R$ and $R_{clean} = \emptyset$. The system first trains the model on R_{dirty} to find an initial model $\theta^{(d)}$ that the system will subsequently improve iteratively. It turns out that SGD converges for an arbitrary initialization, so $\theta^{(d)}$ need not be very accurate. This can be done by featurizing the dirty records (e.g., using an arbitrary placeholder for missing values), and then training the model.

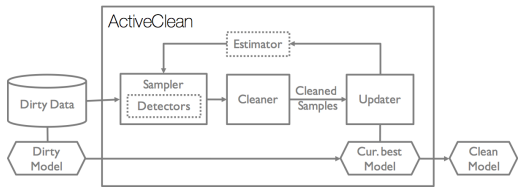


Figure 2: ActiveClean allows users to train predictive models while progressively cleaning data but preserves convergence guarantees. Solid boxes are essential components for correctness and dotted boxes indicate optimizations that can improve model accuracy by up-to 2.5x (Section 7.4.2)

In the next step, the *Sampler* selects a batch of data S from the data that has not been cleaned already. To ensure convergence, the Sampler has to do this in a randomized way, but can assign higher probabilities to some data as long as no data has a zero sampling probability. The *Cleaner* is user-specified and executes $C(\cdot)$ for each sample record and outputs their cleaned versions. The *Updater* uses the cleaned batch to update the model using a Gradient Descent step, thus moving the model closer to the true cleaned model (in expectation). Finally, the system either terminates due to a stopping condition (e.g., $C(\cdot)$ has been called a maximum number of times k, or training error convergence), or passes control to the *sampler* for the next iteration.

A user provided *Detector* can be used to identify records that are more likely to be dirty (e.g., using data quality rules), and thus improves the likelihood that the next sample will contain true dirty records. Furthermore, the *Estimator* uses previously cleaned data to estimate the effect that cleaning a given record will have on the model. These components are optional, but our experiments show that these optimizations can improve model accuracy by up-to 2.5x (Section 7.4.2).

Going back to the ProPublica example in Section 2.1:

EXAMPLE 1. *The analyst chooses to use an SVM model, and manually cleans records by hand (the $C(\cdot)$). ActiveClean initially selects a sample of 50 records (the default) to show the analyst. She identifies records that are dirty, fixes them by normalizing the drug and corporation names with the help of a search engine, and corrects the labels with typographical or incorrect values. The system then uses the cleaned records to update the current best model and selects the next sample of 50. The analyst can stop at any time and use the improved model to predict whether a record is fraudulent or not.*

3.4 Problem Statements

Update Problem: Given a newly cleaned batch of data S_{clean} and the current best model $\theta^{(t)}$, the model update problem is to calculate $\theta^{(t+1)}$. $\theta^{(t+1)}$ will have some error with respect to the true model $\theta^{(c)}$, which we denote as:

$$error(\theta^{(t+1)}) = \|\theta^{(t+1)} - \theta^{(c)}\|$$

The Update Problem is to update the model with a monotone convergence guarantee such that more cleaning implies a more accurate model.

Since the sample is potentially stochastic, it is only meaningful to talk about expected errors. Formally, we require that the expected error is upper bounded by a monotonically decreasing function μ of the amount of cleaned data:

$$\mathbb{E}(error(\theta^{new})) = O(\mu(\mid S_{clean} \mid))$$

Prioritization Problem: The prioritization problem is to select S_{clean} in such a way that the model converges in the fewest iterations possible. Formally, in each batch of data, every r has a probability $p(r)$ of being included. The Prioritization Problem is to select a sampling distribution $p(\cdot)$ to maximize the progress made which each iteration of the update algorithm. We derive the optimal sampling distribution for the updates, and show how the theoretical optimum can be approximated, while still preserving convergence.

4. UPDATING THE MODEL

This section describes an algorithm for reliable model updates. For convex loss minimization, Stochastic Gradient Descent converges to an optimum from any initialization as long each gradient descent step is unbiased. We show how we can leverage this property to prove convergence for interactive data cleaning regardless of the inaccuracy of the initial model–as long as the analyst does not systematically exclude certain data from cleaning. The updater only assumes that it is given a sample of data S_{dirty} from R_{dirty} where $i \in S_{dirty}$ has a known sampling probability $p(i)$.

4.1 Convex Loss Models

Formally, suppose x is a feature vector and y is a label. For labeled training examples $\{(x_i, y_i)\}_{i=1}^N$, the problem is to find a vector of model parameters θ by minimizing a loss function ϕ (a function that measures prediction error) over all training examples:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \phi(x_i, y_i, \theta)$$

where ϕ is a convex function in θ. For example, in a linear regression ϕ is:

$$\phi(x_i, y_i, \theta) = \|\theta^T x_i - y_i\|_2^2$$

Sometimes, a convex *regularization* term $r(\theta)$ is added to the loss:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \phi(x_i, y_i, \theta) + r(\theta)$$

However, we ignore this term without loss of generality, since none of our results require analysis of the regularization. The regularization can be moved into the sum as a part of ϕ for the purposes of this paper.

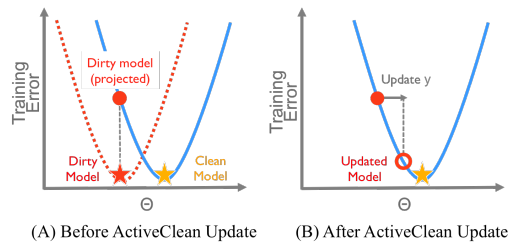
4.2 Geometric Derivation

The update algorithm intuitively follows from the convex geometry of the problem. Consider the problem in one dimension (i.e., the parameter θ is a scalar value), so then the goal is to find the minimum point (θ) of a curve $l(\theta)$. The consequence of dirty data is that the wrong loss function is optimized. Figure 3A illustrates the consequence of the optimization. The red dotted line shows the loss function on the dirty data. Optimizing the loss function finds $\theta^{(d)}$ at the minimum point (red star). However, the true loss function (w.r.t to the clean data) is in blue, thus the optimal value on the dirty data is in fact a suboptimal point on clean curve (red circle).

In the figure, the optimal clean model $\theta^{(c)}$ is visualized as a yellow star. The first question is which direction to update $\theta^{(d)}$ (i.e., left or right). For this class of models, given a suboptimal point, the direction to the global optimum is the gradient of the loss function. The gradient is a p-dimensional vector function of the current model $\theta^{(d)}$ and the clean data. Therefore, ActiveClean needs to update $\theta^{(d)}$ some distance γ (Figure 3B):

$$\theta^{new} \leftarrow \theta^{(d)} - \gamma \cdot \nabla\phi(\theta^{(d)})$$

At the optimal point, the magnitude of the gradient will be zero. So intuitively, this approach iteratively moves the model downhill



(A) Before ActiveClean Update (B) After ActiveClean Update

Figure 3: (A) A model trained on dirty data can be thought of as a sub-optimal point w.r.t to the clean data. (B) The gradient gives us the direction to move the suboptimal model to approach the true optimum.

(transparent red circle) – correcting the dirty model until the desired accuracy is reached. However, the gradient depends on all of the clean data, which is not available, and ActiveClean will have to approximate the gradient from a sample of newly cleaned data.

To derive a sample-based update rule, the most important property is that sums commute with derivatives and gradients. The convex loss class of models are sums of losses, so given the current best model θ, the true gradient $g^*(\theta)$ is:

$$g^*(\theta) = \nabla\phi(\theta) = \frac{1}{N}\sum_{i=1}^{N}\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta)$$

ActiveClean needs to estimate $g^*(\theta)$ from a sample S, which is drawn from the dirty data R_{dirty}. Therefore, the sum has two components which are the gradient from the already clean data g_C and g_S a gradient estimate from a sample of dirty data to be cleaned:

$$g(\theta) = \frac{\mid R_{clean} \mid}{\mid R \mid} \cdot g_C(\theta) + \frac{\mid R_{dirty} \mid}{\mid R \mid} \cdot g_S(\theta) \qquad (1)$$

g_C can be calculated by applying the gradient to all of the already cleaned records:

$$g_C(\theta) = \frac{1}{\mid R_{clean} \mid}\sum_{i \in R_{clean}}\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta)$$

g_S can be estimated from a sample by taking the gradient w.r.t each record, and re-weighting the average by their respective sampling probabilities. Before taking the gradient, the cleaning function $C(\cdot)$ is applied to each sampled record. Therefore, let S be a sample of data, where each $i \in S$ is drawn with probability $p(i)$:

$$g_S(\theta) = \frac{1}{\mid S \mid}\sum_{i \in S}\frac{1}{p(i)}\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta)$$

Then, at each iteration t, the update becomes:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma \cdot g(\theta^{(t)})$$

4.3 Model Update Algorithm

We present an outline for one iteration of the update algorithm. To summarize, the algorithm is initialized with $\theta^{(0)} = \theta^{(d)}$ which is the dirty model. There are three user set parameters: the budget k, batch size b, and the step size γ. In the following section, we provide references from the convex optimization literature that allow the user to appropriately select these values. At each iteration $t = \{1, ..., T\}$, the cleaning is applied to a batch of data b selected from the set of candidate dirty records R_{dirty}. Then, an average gradient is estimated from the cleaned batch and the model is updated. Iterations continue until $k = T \cdot b$ records are cleaned.

The algorithm is as follows:

1. Take a sample of data S from R_{dirty}
2. Calculate the gradient over the sample of newly clean data and call the result $g_S(\theta^{(t)})$
3. Calculate the average gradient over all of the already clean records in $R_{clean} = R - R_{dirty}$, and call the result $g_C(\theta^{(t)})$
4. Apply the following update rule, which is a weighted average of the gradient on the already clean records and newly cleaned records:
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma \cdot \left(\frac{\mid R_{dirty} \mid}{\mid R \mid} \cdot g_S(\theta^{(t)}) + \frac{\mid R_{clean} \mid}{\mid R \mid} \cdot g_C(\theta^{(t)})\right)$$
5. Append the newly cleaned records to set of previously clean records $R_{clean} = R_{clean} \cup S$

This basic algorithm will serve as the scaffolding for the optimizations in the subsequent sections. For example, if we know that a record is likely to be clean, we can move it from R_{dirty} to R_{clean} without having to sample it. Similarly, we can set the sampling probabilities $p(\cdot)$ to favor records that are likely to affect the model.

4.4 Analysis with Stochastic Gradient Descent

The update algorithm can be formalized as a class of very well studied algorithms called Stochastic Gradient Descent (SGD; more precisely the mini-batch variant of SGD). In SGD, random subsets of data are selected at each iteration and the average gradient is computed for every batch. The basic condition for convergence is that the gradient steps need to be on average correct. We provided the intuition that this is the case in Equation 1, and this can be more rigorously formalized as an unbiased estimate of the true gradient (see T.R [24]). Then model is guaranteed to converge essentially with a rate proportional to the inaccuracy of the sample-based estimate.

One key difference with the tpyical application of SGD is that ActiveClean takes a *full* gradient step on the already clean data (i.e., not sampled) and averages it with a stochastic gradient step on the dirty data (i.e., sampled). This is because that data is already clean and we assume that the time-consuming step is the data cleaning and not the numerical operations. The update algorithm can be thought of as a variant of SGD that lazily materializes the clean value. As data is sampled at each iteration, data is cleaned when needed. It is well-known that even for an arbitrary initialization, SGD makes significant progress in less than one epoch (a pass through the entire dataset) [9]. However, the dirty model can be much more accurate than an arbitrary initialization leading to even faster convergence.

Setting the step size γ: There is extensive literature in machine learning for choosing the step size γ appropriately. γ can be set either to be a constant or decayed over time. Many machine learning frameworks (e.g., MLLib, Sci-kit Learn, Vowpal Wabbit) automatically set this value. In the experiments, we use a technique called inverse scaling where there is a parameter $\gamma_0 = 0.1$, and at each iteration it decays to $\gamma_t = \frac{\gamma_0}{|S|t}$.

Setting the batch size b: The batch size should be set by the user to have the desired properties. Larger batches will take longer to clean and will make more progress towards the clean model but will have less frequent model updates. On the other hand, smaller batches are cleaned faster and have more frequent model updates. In the experiments, we use a batch size of 50 which converges fast but allows for frequent model updates. If a data cleaning technique requires a larger batch size than 50, ActiveClean can abstract this size away from the update algorithm and apply the updates in

smaller batches. For example, the batch size set by the user might be $b = 1000$, but the model updates after every 50 records are cleaned. We can disassociate the batching requirements of SGD and the batching requirements of the data cleaning technique.

Convergence Conditions and Properties The convergence rates of SGD are also well analyzed [8,11,37]. The analysis gives a bound on the error of intermediate models and the expected number of steps before achieving a model within a certain error. For a general convex loss, a batch size b, and T iterations, the convergence rate is bounded by $O(\frac{\sigma^2}{\sqrt{bT}})$. σ^2 is a measure of how well the sample gradient estimates the gradient over the entire dataset, and $\sqrt{T}$ shows that each iteration has diminishing returns. σ^2 is the variance in the estimate of the gradient at each iteration:

$$\sigma^2 = \mathbb{E}(\|g - g^*\|^2)$$

where g^* is the gradient computed over the full data if it were fully cleaned. This property of SGD allows us to bound the model error with a monotonically decreasing function of the number of records cleaned, thus satisfying the reliability condition in the problem statement.

Gradient descent techniques also can be applied to non-convex losses and they are widely used in graphical model inference and deep learning. In this case, however, instead of converging to a global optimum, they converge to a locally optimal value that dependends on the initialization. In the non-convex setting, ActiveClean will converge to the closest locally optimal value to the dirty model which is how we initialize ActiveClean. Because of this, it is harder to reason about the objective quality of the results and to define accuracy. Different initializations may lead to different local optima, and thus, introduce a complex dependence on the initialization with the dirty model.

EXAMPLE 2. *Recall that the analyst has a dirty SVM model on the dirty data $\theta^{(d)}$. She decides that she has a budget of cleaning 100 records, and decides to clean the 100 records in batches of 10 (set based on how fast she can clean the data, and how often she wants to see an updated result). All of the data is initially treated as dirty with $R_{dirty} = R$ and $R_{clean} = \emptyset$. The gradient of a basic SVM is given by the following function:*

$$\nabla\phi(x, y, \theta) = \begin{cases} -y \cdot \boldsymbol{x} & \text{if } y\boldsymbol{x} \cdot \theta < 1 \\ 0 & \text{if } y\boldsymbol{x} \cdot \theta \geq 1 \end{cases}$$

For each iteration t, a sample of 10 records S is drawn from R_{dirty}. ActiveClean then applies the cleaning function to the sample. Using these values, ActiveClean estimates the gradient on the newly cleaned data:

$$\frac{1}{10}\sum_{i \in S}\frac{1}{p(i)}\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta)$$

ActiveClean also applies the gradient to the already clean data (initially non-existent):

$$\frac{1}{\mid R_{clean} \mid}\sum_{i \in R_{clean}}\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta)$$

Then, it calculates the update rule:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma \cdot (\frac{\mid R_{dirty} \mid}{\mid R \mid} \cdot g_S(\theta^{(t)}) + \frac{\mid R_{clean} \mid}{\mid R \mid} \cdot g_C(\theta^{(t)}))$$

Finally, $R_{dirty} \leftarrow R_{dirty} - S$, $R_{clean} \leftarrow R_{clean} + S$, and continue to the next iteration.

5. DIRTY DATA DETECTION

If corrupted records are relatively rare, sampling might be very inefficient. The analyst may have to sample many batches of data before finding a corrupted record. In this section, we describe how we can couple ActiveClean with prior knowledge about which data are likely to be dirty. In the data cleaning literature, error detection and error repair are treated as two distinct problems [10,14,30]. Error detection is often considered to be substantially easier than error repair since one can declare a set of integrity rules on a database (e.g., an attribute must not be NULL), and select rows that violate those rules. On the other hand, repair is harder and often requires human involvement (e.g., imputing a value for the NULL attribute).

5.1 Detection Problem

First, we describe the required properties of the dirty data detector. The detector returns two important aspects of a record: (1) whether the record is dirty, and (2) if it is dirty, on which attributes there are errors. The sampler can use (1) to select a subset of dirty records to sample at each batch and the estimator can use (2) to estimate the value of data cleaning based on other records with the same corruption.

DEFINITION 1 (DETECTOR). *Let r be a record in R. A detector is a function that returns a Boolean of whether the record is dirty and a set of attributes e_r that are dirty.*

$$D(r) = (\{0, 1\}, e_r)$$

From the set of attributes that are dirty, we can find the corresponding features that are dirty f_r and labels that are dirty l_r since we assume a one-to-one mapping between records and training examples. We will consider two types of detectors: exact rule-based detectors that detect integrity constraint or functional dependency violations, and approximate adaptive detectors that learn which data are likely to be dirty.

5.2 Rule-Based Detector

Data quality rules are widely studied as a technique for detecting data errors. In most rule-based frameworks, an analyst declares a set of rules Σ and checks whether a relation R satisfies those rules. The rules rules can be declared in advance before applying ActiveClean, or constructed from the first batch of sampled data. ActiveClean is compatible with many commonly used classes of rules for error detection including integrity constraints (ICs), conditional functional dependencies (CFDs), and matching dependencies (MDs). The only requirement on the rules is that there is an algorithm to enumerate the set of records that violate at least one rule.

Let R_{viol} and R_{sat} be the subset of records in R_{ditry} that violate at least one rule and satisfy all rules respectively. The rule-based detector modifies the update workflow in the following way:

1. $R_{clean} = R_{clean} \cup R_{sat}$
2. $R_{dirty} = R_{viol}$
3. Apply the algorithm in Section 4.3.

EXAMPLE 3 (RULE-BASED DETECTION). *An example of a rule on the running example dataset is that the* `status` *of a contribution can be only "allowed" or "disallowed". Any other value for* `status` *is considered violation.*

5.3 Adaptive Detection

Rule-based detection is not possible in all cases, especially in cases where the analyst selectively modifies data. This is why we propose an alternative called the adaptive detector. Essentially, we reduce the problem to training a classifier on previously cleaned data. Note that this "learning" is distinct from the "learning" in the user-specified statistical model. One challenge is that the detector needs to describe how the data is dirty. The detector achieves this by categorizing the corruption into u classes, and using a multi-class classifier. These classes are corruption categories that do not necessarily align with features, but every record is classified with at most one category.

When using adaptive detection, the repair step has to clean the data and report to which of the u classes the corrupted record belongs. When an example (x, y) is cleaned, the repair step labels it with one of the clean, $1, 2, ..., u$. It is possible that u increases each iteration as more types of dirtiness are discovered. In many real world datasets, data errors have locality, where similar records tend to be similarly corrupted. There are usually a small number of error classes even if a large number of records are corrupted. This problem can be addressed by any classifier, and we use an all-versus-one Logistic Regression in our experiments.

The adaptive detector modifies the update workflow in the following way:

1. Let R_{clean} be the previously cleaned data, and let U_{clean} be a set of labels for each record indicating the error class and if they are dirty or "not dirty".
2. Train a classifier to predict the label $\mathsf{Train}(R_{clean}, U_{clean})$
3. Apply the classifier to the dirty data $\mathsf{Predict}(R_{dirty})$
4. For all records predicted to be clean, remove from R_{dirty} and append to R_{clean}.
5. Apply the algorithm in Section 4.3.

The precision and recall of this classifier should be tuned to favor classifying a record as dirty to avoid falsely moving a dirty record into R_{clean}. In our experiments, we set this value to 0.90 probability of the "clean" class.

6. SELECTING RECORDS TO CLEAN

The algorithm proposed in Section 4.3 will convege for any sampling distribution where $p(\cdot) > 0$ for all records, albeit different distributions will have different convergence rates. The sampling algorithm is designed to include records in each batch that are most valuable to the analyst's model with a higher probability.

6.1 Optimal Sampling Problem

Recall that the convergence rate of an SGD algorithm is bounded by σ^2 which is the variance of the gradient. Intuitively, the variance measures how accurately the gradient is estimated from a uniform sample. Other sampling distributions, while preserving the same expected value, may have a lower variance. Thus, the optimal sampling problem is defined as a search over sampling distributions to find the minimum variance sampling distribution.

DEFINITION 2 (OPTIMAL SAMPLING PROBLEM). *Given a set of candidate dirty data R_{dirty}, $\forall r \in R_{dirty}$ find sampling probabilities $p(r)$ such that over all samples S of size k it minimizes the variance:*

$$\arg\min_{p} \mathbb{E}(\|g_S - g^*\|^2)$$

It can be shown [37] that the optimal distribution over records in R_{dirty} is proportional to: $p_i \propto \|\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta^{(t)})\|$ Intuitively, this sampling distribution prioritizes records with higher gradients, i.e., make a larger impact during optimization. The challenge is that this particular optimal distribution depends on knowing the clean value of a records, which is a chicken-and-egg problem: the optimal sampling distribution requires knowing $(x_i^{(c)}, y_i^{(c)})$; however, we are sampling the values so that they can be cleaned.

One natural solution is to calculate this gradient with respect to the dirty values–implicitly assuming that the corruption is not that severe:

$$p_i \propto \|\nabla\phi(x_i^{(d)}, y_i^{(d)}, \theta^{(t)})\|$$

This solution is highly related to the Expected Gradient Length heuristic that has been proposed before in Active Learning [31]. However, there is additional structure to the data cleaning problem. As the analyst cleans more data, we can build a model for how cleaned data relates to dirty data. By using the detector from the previous section to estimate the impact of data cleaning, we show that we can estimate the cleaned values. We find that this optimization can improve the convergence rate by a factor of 2 in some datasets.

6.2 The Estimator

We call this component, the estimator, which connects the detector and the sampler. The goal of the estimator is to estimate $\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta^{(t)})$ (the clean gradient) using $\nabla\phi(x_i^{(d)}, y_i^{(d)}, \theta^{(t)})$ (the dirty gradient) and $D(r)$ (the detection results). As a strawman approach, one could use all of the previously cleaned data (tuples of dirty and clean records), and use another learning algorithm to relate the two. The main challenges with this approach are: (1) the problem of scarcity where errors may affect a small number of records and a small number of attributes and (2) the problem of modeling where if we have an accurate parametric model relating clean to dirty data then why clean the data in the first place. To address these challenges, we propose a light-weight estimator uses a linear approximation of the gradient and only relies on the average change in each feature value. Empirically, we find that this estimator provides more accurate early estimates and is easier to tune than the strawman approach until a large amount of data are cleaned (Section 7.4.6).

6.3 Linearization Algorithm

The basic idea is for every feature i to maintain an average change δ_i before and after cleaning–estimated from the previously cleaned data. This avoids having to design a complex internal model to relate the dirty and clean data, and is based on just estimating sample means. This would work if the gradient was linear in the features, and we could write this estimate as:

$$\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta^{(t)}) \approx \nabla\phi(x_i^{(d)}, y_i^{(d)}, \theta^{(t)}) + A \cdot [\delta_1, ..., \delta_p]^\intercal$$

The problem is that the gradient $\nabla\phi(\cdot)$ can be a very non-linear function of the features that couple features together. For example, even in the simple case of linear regression, the gradient is a non-linear function in x:

$$\nabla\phi(x, y, \theta) = (\theta^T x - y)x$$

It is not possible to isolate the effect of a change of one feature on the gradient. Even if one of the features is corrupted, all of the gradient components can be incorrect.

The way that we address this problem is to linearize the gradient, where the matrix A is found by computing a first-order Taylor

Series expansion of the gradient. If d is the dirty value and c is the clean value, the Taylor series approximation for a function f is given as follows:

$$f(c) = f(d) + f'(d) \cdot (d - c) + ...$$

In our case, the function f is the gradient $\nabla\phi$, and the linear term $f'(d) \cdot (d - c)$ is a linear function in each feature and label:

$$\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta^{(t)}) \approx \nabla\phi(x_i^{(d)}, y_i^{(d)}, \theta^{(t)})$$
$$+\frac{\partial}{\partial X}\nabla\phi(x_i^{(d)}, y_i^{(d)}, \theta^{(t)}) \cdot (x^{(d)} - x^{(c)})$$
$$+\frac{\partial}{\partial Y}\phi(x_i^{(d)}, y_i^{(d)}, \theta^{(t)}) \cdot (y^{(d)} - y^{(c)})$$

This can be simplified with two model-dependent matrices M_x (relating feature changes to the gradient) and M_y (relating label changes to the gradient) [1]:

$$\approx \phi(x_i^{(d)}, y_i^{(d)}, \theta^{(t)}) + M_x \cdot \Delta x + M_y \cdot \Delta y$$

where $\Delta_x = [\delta_1, ..., \delta_p]^\intercal$ is the average change in each feature and $\Delta_y = [\delta_1, ..., \delta_l]^\intercal$ is the average change in each label. It follows that the resulting sampling distribution is:

$$p(r) \propto \|\nabla\phi(x_i^{(d)}, y_i^{(d)}, \theta^{(t)}) + M_x \cdot \Delta_x + M_y \cdot \Delta_y\|$$

7. EXPERIMENTS

We start by presenting two end-to-end scenarios based on a dataset of movies from IMDB and a dataset of medical donations from ProPublica. Then, we evaluate each of the components of ActiveClean on standard Machine Learning benchmarks (Tax Record Classification, EEG anomaly detection) with synthetic errors.

7.1 Setup

We compare ActiveClean to alternative solutions along two primary axes: the sampling procedure to pick the next set of records to clean, and the model update procedure to incorporate the cleaned sample. All of the compared approaches clean the same amount of data, and we evaluate the accuracy of the approaches as a function of the amount of data examined; defined as the number of evaluations of the user-specified $C()$ (cleaner) on a record whether or not the record is actually dirty.

Naive-Mix (NM): In each iteration, Naive-Mix draws a random sample, merges the cleaned sample back into the dataset, and retrains over the entire dataset.

Naive-Sampling (NS): In contrast to Naive-Mix, Naive-Sampling only re-trains over the set of records that have been cleaned so far.

Active Learning (AL): The samples are picked using Active Learning (more precisely, Uncertainty Sampling [31]), and the model is re-trained over the set of records cleaned so far.

Oracle (O): Oracle has complete access to the ground truth, and in each iteration, selects samples that maximize the expected convergence rate. It uses ActiveClean's model update procedure.

Metrics: We evaluate these approaches on two metrics: *model error*, which is the distance between the trained model and true model if all data were cleaned $\|\theta - \theta^{(c)}\|$, and *test error*, which is the prediction accuracy of the model on a held out set of clean data.

7.2 Real Scenarios

We now describe two data analyst classification scenarios based on popular industry use cases reported in the Apache Spark user survey [1] — these experiments are run with real data and real corruption. The first is a content tagging problem to categorize movies from plot descriptions; the second is a fraud detection problem to determine whether a medical donation is suspicious. Both of the datasets are plagued by systematic errors that affect the classification accuracy by $10 - 35\%$. One indication of the systematic nature of the data error is that errors disproportionately affect one class rather than another. We simulate the analyst's cleaning procedure by looking up the cleaned value in the dataset that we cleaned beforehand. See [24] for constraints, errors, and cleaning methodology, as well as an additional regression scenario.

7.2.1 Movie Prediction

The first scenario uses two published datasets about movies, from IMDB [2] and Yahoo [3]. Each movie has a title, a short 1-2 paragraph plot description, and a list of categories, and the goal is to train a model to predict whether a movie is a "Horror" or "Comedy" from the description and title. The text data was featurized using a TFIDF model.

The IMDB dataset has 486,298 movies and is very dirty. The category list of many movies has redundant and possibly conflicting tags (e.g., "Kids" and "Horror" for the same movie). Also, the plot and title text may have errors from the scraping procedure (e.g., "brĈuin, the"). This dataset also shows experimental evidence for systematic bias. Horror movies were more likely to be erroneously tagged, and consequently, a classifier trained on the dirty data favored "Comedy" predictions. In contrast, the smaller Yahoo dataset (106,959 movies) is much cleaner, and nearly all of the movies are found in the IMDB dataset.

To clean the category lists, this smaller dataset to cross-referencing records when possible and import categories from Yahoo. This was done using a simple entity resolution to match movie titles between the two dataset. When there was a sufficiently close textual match (in terms of title string similarity), we imported the Yahoo dataset's category list to the IMDB dataset. For the movies that did not match, we appended the records to the IMDB dataset. Finally, we filtered the dataset for movies whose category lists included "Horror" or "Comedy". To clean the parsing errors, we identify common parsing artifacts in each sampled batch and write a script to fix all records with that problem (e.g., remove all instances of "Ĉ").

Figure 4a plots the model error and test error versus the number of records examined by each algorithm. For very small batches of data (e.g., 50 out of 400,000), some of the techniques are actually less accurate than the dirty model. This is because the error due to the particular initial sampled batch of dominates, and we find that differences between the techniques are not statistically significant in that regime. However, as more data is cleaned, we see a clear seperation between the techniques The purple bottom curve is the Oracle approach, and we find that out of the practical approaches ActiveClean reduces the model error the fastest, and is the only method that converges to the Oracle performance. Figure 4b directly shows how ActiveClean rapidly reduces the test error — after 2000 records, ActiveClean's test error is within 0.02 of the fully cleaned model. ActiveClean provides superior convergence for two main reasons. First, the update algorithm can incorporate both the raw data as well as the smaller set of cleaned records. This makes ActiveClean less sensitive to sampling error. Next, ActiveClean also selects records that are more likely to by dirty (Section 5) and will will most improve the model (Section 6). This is illustrated in ActiveClean's faster convergence curve in the first 2000 cleaned records.

[1] A number of example linearizations are listed in [24]

[2] ftp://ftp.fu-berlin.de/pub/misc/movies/database/

[3] http://webscope.sandbox.yahoo.com/catalog.php?datatype=r

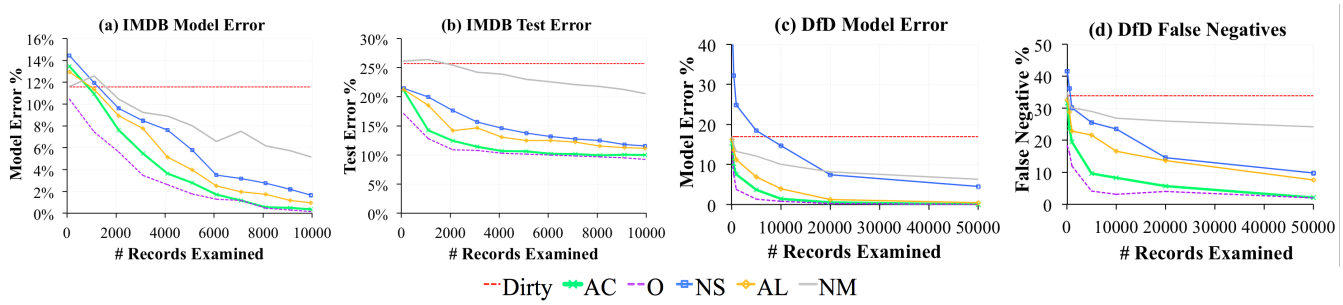


Figure 4: (a) We plot the relative model error as a function of the number of records cleaned for each of the alternatives on the IMDB content tagging task. ActiveClean results in the fastest convergence. (b) Faster convergence translates into improved test accuracy. On a holdout set of 20%, ActiveClean is more accurate than the alternatives. (c) We evaluate ActiveClean on the Dollars For Docs fraud detection task and find similar results, where ActiveClean converges faster than the alternatives. (d) ActiveClean improves the false negative rate (detection error) of the classifier resulting in an improved fraud detector.

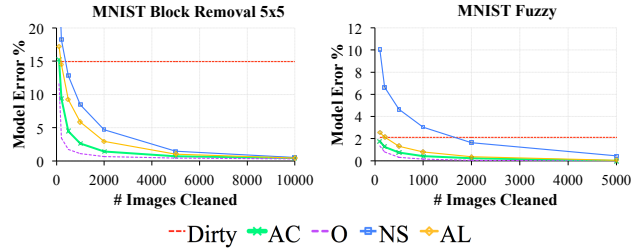


Figure 5: In an image processing pipeline on the MNIST dataset with simulated errors, ActiveClean outperforms Active Learning and Naive-Sample.

7.2.2 Dollars For Docs (DfD)

The second scenario explores ProPublica's Dollars for Docs dataset described in Section 2.1. We featurize the 5 text attributes using a bag-of-words representation, and our goal is to predict the status of medical donations using an SVM ("allowed" or "disallowed"). Figure 4c plots the model error as a function of the number of records examined for each of the techniques. As in the IMDB dataset, we find that ActiveClean converges faster than Naive-Mix, Naive-Sampling, and Active Learning. In fact, ActiveClean can achieve comparable model accuracy (¡ 1% of the true model accuracy) while expending a fraction of the data cleaning effort (10% of the records cleaned).

In terms of test error, Figure 4d reports the detection error or false negative % (percentage of disallowed research contributions that the model misses) on a held-out evaluation set. This measures the real-world utility of the classifier learned with ActiveClean. The dirty and fully cleaned models have respectively a 34% and 3% detection error. Due to the systematic bias in the data errors (explained in Section 2.1), ActiveClean is able to identify the dirty records and reduce the detection error to 8% after examining 10,000 records.

Overall, we found that systematic errors are indeed present in real-world datasets, and that ActiveClean can effectively identify and exploit this bias to more quickly converge to the true clean model as compared to the naive or active learning approaches. In fact, we found that the commonly used retraining approach (Naive-Mix) was almost completely ineffective as compared to other model update techniques.

7.3 Simulated Machine Learning Pipeline

This experiment is representative of modern machine learning pipelines such as AMPLab's Keystone ML [3] and Google's Tensor Flow [5]. The task is to classify 60,000 images of handwritten digits from the MNIST dataset into 10 categories with a one-to-all multiclass SVM classifier [4]. In contrast to the prior scenarios, which directly extracted feature vectors from the raw dataset, image feature extraction involves a pipeline of transformation steps including edge detection, projection, and raw image patch extraction [3,5]. The cleaning function $C()$ involves replacing a potentially corrupted image with a non-corrupted version. We find that pipelines tend to propagate small amounts of corruption, and in fact, and even randomly generated errors can morph into systematic biases.

There are two types of simulated corruptions that mimic standard corruptions in image processing (occlusion and low-resolution): `5x5 Removal` deletes a random 5x5 pixel block by setting the pixel values to 0, and `Fuzzy` blurs the entire image using a 4x4 moving average patch. We apply these corruptions to a random 5% of the images.

Figure 5 shows that ActiveClean makes more progress towards the clean model with a smaller number of examples cleaned. To achieve a 2% error for 5x5 Removal, ActiveClean can inspect 2200 fewer images than Active Learning and 2750 fewer images than Naive-Sampling. For the fuzzy images, both Active Learning and ActiveClean reach 2% error after examining < 100 images, while Naive-Sampling requires 1750. Even though these corruptions are generated independently of the data, the `5x5 Removal` propagates through the pipeline as a systematic error. The image features are constructed with edge detectors, which are highly sensitive to this type of corruption. Digits that naturally have fewer edges than others are disproportionately affected since the removal process adds spurious edges. On the other hand, the `Fuzzy` corruption propagates through the pipeline are similar to random errors (as opposed to systematic).

7.4 Simulated Error Scenarios

In the next set of experiments, we use standard Machine Learning benchmark datasets and corrupt them with varying levels of systematic noise. We use this to evaluate ActiveClean while isolating certain variables. The two datasets that we used were:

[4] `http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset`

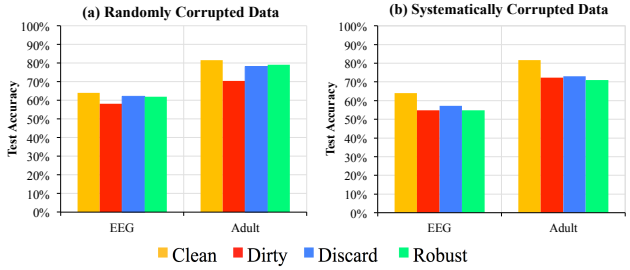


Figure 6: (a) Robust techniques and discarding data work when corrupted data are random and look atypical. (b) Data cleaning can provide reliable performance in both the systematically corrupted setting and randomly corrupted setting.

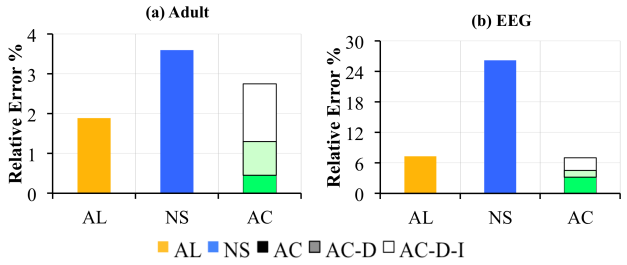


Figure 7: -D denotes no detection, and -D-I denotes no detection and no importance sampling. Both optimizations significantly help ActiveClean outperform Naive-Sampling and Active Learning.

Income Classification (Adult): In this dataset of 45,552 records, the task is to predict the income bracket (binary) from 12 numerical and categorical covariates with an SVM classifier.

Seizure Classification (EEG): In this dataset, the task is to predict the onset of a seizure (binary) from 15 numerical covariates with a thresholded Linear Regression. There are 14980 data points in this dataset. This classification task is inherently hard with an accuracy on the completely clean data of only 65%.

7.4.1 *Data Cleaning v.s. Robust Statistics*

Machine Learning has broadly studied a number of *robust* methods to deal with some types of outliers. In particular, this field studies random high-magnitude outliers and techniques to make statistical model training agnostic to their presence. Feng et al. proposed a variant of logistic regression that is robust to outliers [16]. We chose this algorithm because it is a robust extension of the convex regularized loss model, leading to a better apples-to-apples comparison between the techniques. Our goal is to understand which types of data corruption are amenable to data cleaning and which are better suited for robust statistical techniques. The experiment compares four schemes: (1) full data cleaning, (2) baseline of no cleaning, (3) discarding the dirty data, and (4) robust logistic regression. We corrupted 5% of the training examples in each dataset in two different ways:

Random Corruption: Simulated high-magnitude random outliers. 5% of the examples are selected at random and a random feature is replaced with 3 times the highest feature value.

Systematic Corruption: Simulated innocuous looking (but still incorrect) systematic corruption. The model is trained on the clean data, and the three most important features (highest weighted) are identified. The examples are sorted by each of these features and the top examples are corrupted with the mean value for that feature (5% corruption in all).

Figure 6 shows the test accuracy for models trained on both types of data with the different techniques. The robust method performs well on the random high-magnitude outliers with only a 2.0% reduction in clean test accuracy for EEG and 2.5% reduction for Adult. In the random setting, discarding dirty data also performs relatively well. However, the robust method falters on the systematic corruption with a 9.1% reduction in clean test accuracy for EEG and 10.5% reduction for Adult. Data cleaning is the most reliable option across datasets and corruption types. The problem is that without cleaning, there is no way to know if the corruption is random or systematic and when to trust a robust method. While data cleaning requires more effort, it provides benefits in both settings. The remaining experiments, unless otherwise noted, use systematic corruption.

7.4.2 *Source of Improvements*

This experiment compares the performance of ActiveClean with and without various optimizations for 500 records examined. ActiveClean without detection is denoted as (AC-D) (that is at each iteration we sample from the entire dirty data), and ActiveClean without detection and our prioritized sampling is denoted as (AC-D-I). Figure 7 plots the relative error of the alternatives and ActiveClean with and without the optimizations. Without detection (AC-D), ActiveClean is still more accurate than Active Learning. Removing the sampling, ActiveClean is slightly worse than Active Learning on the Adult dataset but is comparable on the EEG dataset. The advantage of ActiveClean is that it is a composable framework supporting different instantiations of the detection and prioritization modules while still preserving convergence guarantees. With these optimizations, ActiveClean is consistently more efficient than Active Learning.

7.4.3 *Mixing Dirty and Clean Data*

Naive-Mix is an unreliable methodology lacking the same guarantees as Active Learning or Naive-Sampling even in the simplest of cases. We also saw that it is significantly less efficient than ActiveClean, Naive-Sampling, and Active Learning on the real datasets. For thoroughness, these experiments include the model error as a function of records examined in comparison to ActiveClean. We also evaluate this approach with our detection component. Naive-Mix+D randomly samples data using the dirty data detector, applies the user-specified cleaning, and writes-back the cleaned data.

Figure 8 plots the same curves as the previous experiment comparing ActiveClean, Active Learning, and two mixed data algorithms. Intuitively, Naive-Mix makes less progress with each update. Consider the case where 10% of the dataset is corrupted and a small sample of data 1%. ActiveClean and Naive-Sampling extrapolate from the cleaned sample (ActiveClean extrapolates a gradient) while Naive-Mix considers the entire dirty data which might be much larger.

7.4.4 *Corruption Rate*

This experiment explores the tradeoff between Naive-Sampling and ActiveClean. Figure 9 varies the systematic corruption rate and plots the number of records examined to achieve 1% relative error for Naive-Sampling and ActiveClean. Naive-Sampling does not use the dirty data and thus its error is essentially governed by the the sample size and not the magnitude or prevalence of corruption.

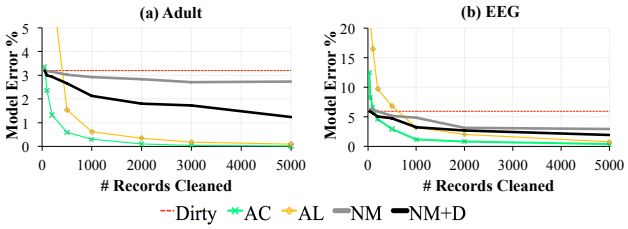


Figure 8: The relative model error as a function of the number of examples cleaned. ActiveClean converges with a smaller sample size to the true result in comparison to partial cleaning (NM, NM+D).

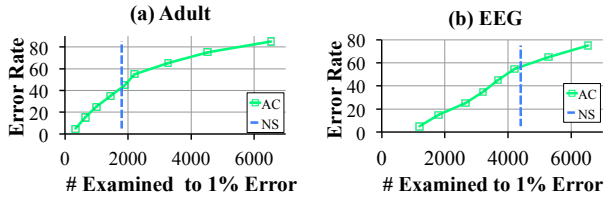


Figure 9: ActiveClean outperforms Naive until the corruption is so severe that the dirty model is initialized very far away from the clean model. The error of Naive-Sampling does not depend on the corruption rate so it is a vertical line.

Naive-Sampling outperforms ActiveClean only when corruptions are very severe (45% in Adult and nearly 60% in EEG). When the initialization with the dirty model is inaccurate, ActiveClean does not perform as well.

7.4.5 *Dirty Data Detection*

Adaptive detection depends on predicting which records are dirty; and this is again related to the systematic nature of data error. For example, random corruption not correlated with any other data features may be hard to learn. As corruption becomes more random, the classifier becomes increasingly erroneous. This experiment explores making our generated systematic corruption incrementally more random. Instead of selecting the highest valued records for the most valuable features, we corrupt random records with probability p. We compare these results to AC-D where we do not have a detector at all (for a fixed number of 1000 records examined). Figure 10a plots the relative error reduction using a classifier. When the corruption is about 50% random then there is a break even point where no detection is better. The classifier is imperfect and misclassifies some data points incorrectly as cleaned.

7.4.6 *Impact Estimation*

This experiment compares estimation techniques: (1) "linear regression" trains a linear regression model that predicts the clean gradient as a function of the dirty gradient, (2) "average gradient" which does not use the detection to inform how to apply the estimate, (3) "average feature change" uses detection but no linearization, and (4) the Taylor series linear approximation. Figure 10b measures how accurately each estimation technique estimates the gradient as a function of the number of examined records on the EEG dataset. Estimation error is measured using the relative L2 error with the true gradient. The Taylor series approximation proposed gives more accurate for small cleaning sizes.

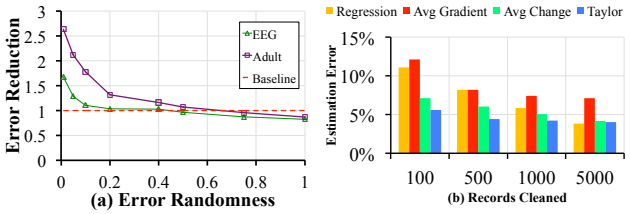


Figure 10: (a) Data corruptions that are less random are easier to classify, and lead to more significant reductions in relative model error. (b) The Taylor series approximation gives more accurate estimates when the amount of cleaned data is small.

8. RELATED WORK

Data Cleaning: There are a number of other works that use machine learning to improve the efficiency and/or reliability of data cleaning [17,35,36]. For example, Yakout et al. train a model that evaluates the likelihood of a proposed replacement value [35]. Another application of machine learning is value imputation, where a missing value is predicted based on those records without missing values. Machine learning is also increasingly applied to make automated repairs more reliable with human validation [36]. Human input is often expensive and impractical to apply to entire large datasets. Machine learning can extrapolate rules from a small set of examples cleaned by a human (or humans) to uncleaned data [17,36]. This approach can be coupled with active learning [25] to learn an accurate model with the fewest possible number of examples. While, in spirit, ActiveClean is similar to these approaches, it addresses a very different problem of data cleaning before user-specified modeling.

SampleClean [33] applies data cleaning to a sample of data, and estimates the results of aggregate queries. Sampling has also been applied to estimate the number of duplicates in a relation [19]. Similarly, Bergman et al. explore the problem of query-oriented data cleaning [7], where given a query, they clean data relevant to that query. Deshpande et al. studied data acquisition in sensor networks [12]. They explored value of information based prioritization of data acquisition for estimating aggregate queries of sensor readings. Similarly, Jeffery et al. [20] explored similar prioritization based on value of information. Existing work does not explore cleaning driven by the downstream machine learning "queries" studied in this work.

Stochastic Optimization and Active Learning: Zhao and Tong recently proposed using importance sampling in conjunction with stochastic gradient descent [37]. This line of work builds on prior results in linear algebra that show that some matrix columns are more informative than others [13], and Active Learning which shows that some labels are more informative that others [31]. Active Learning largely studies the problem of label acquisition [31], and recently the links between Active Learning and Stochastic optimization have been studied [18].

Transfer Learning and Bias Mitigation: ActiveClean has a strong link to a field called Transfer Learning and Domain Adaptation [27]. The basic idea of Transfer Learning is that suppose a model is trained on a dataset D but tested on a dataset D'. Much of the complexity and contribution of ActiveClean comes from efficiently tuning such a process for expensive data cleaning applications – costs not studied in Transfer Learning. Other problems in bias mitigation (e.g., Krishnan et al. [23]) have the same structure, systematically corrupted data that is feeding into a model. In this work,

we try to generalize these principles given a general dirty dataset, convex model, and data cleaning procedure.

Secure Learning: ActiveClean is also related to work in adversarial learning [26], where the goal is to make models robust to adversarial data manipulation. This line of work has extensively studied methodologies for making models private to external queries and robust to malicious labels [34], but the data cleaning problem explores more general corruptions than just malicious labels. One widely applied technique in this field is reject-on-negative impact, which essentially, discards data that reduces the loss function–which will not work when we do not have access to the true loss function (only the "dirty loss").

9. CONCLUSION

The growing popularity of predictive models in data analytics adds additional challenges in managing dirty data. We propose ActiveClean, a model training framework that allows for iterative data cleaning while preserving provable convergence properties. We specifically focus on problems that arise when data error is systematic, i.e., correlated with the hypotheses of interest. The key insight of ActiveClean is that convex loss models (e.g., linear regression and SVMs) can be simultaneously trained and cleaned. ActiveClean also includes numerous optimizations such as: using the information from the model to inform data cleaning on samples, dirty data detection to avoid sampling clean data, and batching updates. The experimental results are promising as they suggest that these optimizations can significantly reduce data cleaning costs when errors are sparse and cleaning budgets are small. Techniques such as Active Learning and SampleClean are not optimized for the sparse low-budget setting, and ActiveClean achieves models of high accuracy for significantly less records cleaned.

This research is supported in part by DHS Award HSHQDC-16-3-00083, NSF CISE Expeditions Award CCF-1139158, DOE Award SN10040 DE-SC0012463, an SFU Presidents Research Start-up Grant (NO. 877335), and DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, IBM, SAP, The Thomas and Stacey Siebel Foundation, Apple Inc., Arimo, Blue Goji, Bosch, Cisco, Cray, Cloudera, Ericsson, Facebook, Fujitsu, HP, Huawei, Intel, Microsoft, Pivotal, Samsung, Schlumberger, Splunk, State Farm and VMware.

10. REFERENCES

[1] Apache spark survey. https://databricks.com/blog/2015/09/24/spark-survey-results-2015-are-now-available.html.

[2] Dollars for docs. https://projects.propublica.org/docdollars/.

[3] Keystone ml. http://keystone-ml.org/.

[4] A pharma payment a day keeps docs' finances okay. https://www.propublica.org/article/a-pharma-payment-a-day-keeps-docs-finances-ok.

[5] Tensor flow. https://www.tensorflow.org/.

[6] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progressive approach to relational entity resolution. In *VLDB*, 2014.

[7] M. Bergman, T. Milo, S. Novgorodov, and W. C. Tan. Query-oriented data cleaning with oracles. In *SIGMOD*, 2015.

[8] D. P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. In *CoRR*, 2015.

[9] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*. 2012.

[10] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons, Inc., 2003.

[11] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. In *JMLR*, 2012.

[12] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2004.

[13] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. In *JMLR*, 2012.

[14] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. 2012.

[15] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[16] J. Feng, H. Xu, S. Mannor, and S. Yan. Robust logistic regression and classification. In *NIPS*, 2014.

[17] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu. Corleone: Hands-off crowdsourcing for entity matching. In *SIGMOD*, 2014.

[18] A. Guillory, E. Chastain, and J. Bilmes. Active learning as non-convex optimization. In *AISTATS*, 2009.

[19] A. Heise, G. Kasneci, and F. Naumann. Estimating the number and sizes of fuzzy-duplicate clusters. In *CIKM*, 2014.

[20] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative support for sensor data cleaning. In *Pervasive Computing*, 2006.

[21] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. In *TVCG*, 2012.

[22] S. Krishnan, D. Haas, M. J. Franklin, and E. Wu. Towards reliable interactive data cleaning: A user survey and recommendations. In *HILDA*, 2016.

[23] S. Krishnan, J. Patel, M. J. Franklin, and K. Goldberg. A methodology for learning, analyzing, and mitigating social influence bias in recommender systems. In *RecSys*, 2014.

[24] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning while learning convex loss models. In *Arxiv*, 2015.

[25] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, and S. Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. In *VLDB*, 2014.

[26] B. Nelson, B. I. P. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. D. Tygar. Query strategies for evading convex-inducing classifiers. In *JMLR*, 2012.

[27] S. J. Pan and Q. Yang. A survey on transfer learning. In *TKDE*. IEEE, 2010.

[28] T. Papenbrock, A. Heise, and F. Naumann. Progressive duplicate detection. In *TKDE*, 2015.

[29] J. Pearl. Causality: models, reasoning and inference. *Economet. Theor*, 19:675–685, 2003.

[30] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. In *IEEE Data Eng. Bull.*, 2000.

[31] B. Settles. Active learning literature survey. In *University of Wisconsin, Madison*, 2010.

[32] E. H. Simpson. The interpretation of interaction in contingency tables. In *Journal of the Royal Statistical Society. Series B (Methodological)*. JSTOR, 1951.

[33] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *SIGMOD*, 2014.

[34] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In *ICML*, 2015.

[35] M. Yakout, L. Berti-Equille, and A. K. Elmagarmid. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In *SIGMOD*, 2013.

[36] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. In *VLDB*, 2011.

[37] P. Zhao and T. Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, 2015.