# ActiveClean: Progressive Data Cleaning For Convex Data Analytics


## ABSTRACT

A perennial challenge in data analytics is presence of dirty data in the form of missing, duplicate, incorrect or inconsistent values. The growing popularity of predictive models leads to additional concerns as these models are highly sensitive to systematic corruption. Although errors can be mitigated through data cleaning, it is often very time consuming. Consequently, expensive cleaning is frequently applied progressively; cleaning only as much as necessary to achieve a desired accuracy. However, existing model training methodologies can return misleading results when trained on a mix of dirty and clean data. The key insight of our framework, ActiveClean, is for convex loss models, data cleaning can be applied during model training (i.e., in the Gradient Descent loop) allowing for progressive cleaning while preserving provable properties. ActiveClean applies a number of optimizations to improve convergence rates such as importance sampling based on value to the model, avoiding data that is expected to be clean, and batching together updates from already cleaned data. Evaluation on four real-world datasets suggests that for a fixed cleaning budget, ActiveClean returns more accurate models than uniform sampling and Active Learning when systematic corruption is sparse.


## 1. INTRODUCTION

Data are susceptible to various forms of corruption such as missing, incorrect, or inconsistent representations [37]. Dirty data can lead to inaccurate analysis, and techniques for processing dirty data are well studied [34]. The growing popularity of predictive models in data analytics [1,6,12,21] leads to additional challenges in managing dirty data. Predictive models rely on learning relationships between features and labels, and systematic corruption [38] (i.e., corruption that disproportionately affects certain data) can mask or even introduce spurious new relationships. Furthermore, the high dimensionality of these models can amplify small problems [43] resulting in error-prone predictions even when trained on mostly clean data.

Consider a music recommender system in which due to a software bug, all users from Europe have an incorrect age attribute defaulted to “18-24". A recommendation model trained on this data may spuriously learn a correlation relationship between age “18-24" and music liked by European users. A bug, which ostensibly affected only the European users’ records, can affect predictions to all users aged “18-24". Systematic corruption prior to featurization is not addressed in the robust Machine Learning literature which focuses on the resilience to outliers (e.g., age “150").

A number of data cleaning frameworks have been recently proposed to address the problem of corrupted data at scale [5,11,24]. However, data analysts report that data cleaning remains one of the most time consuming steps in the analysis process [3]. Data cleaning can require a significant amount of developer effort in writing software or rules to fix the corruption. Crowdsourcing is an increasingly popular alternative with recent success in missing value filling and entity resolution [5,11,19,33]. However, crowdsourcing comes at the cost of additional latency and the overhead of managing human workers.

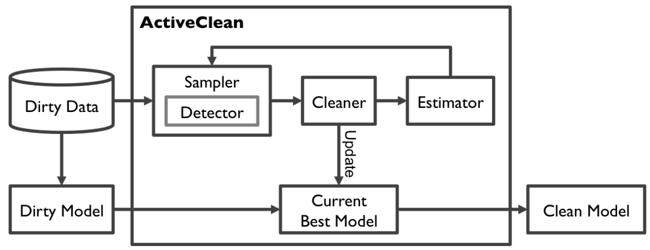


**Figure 1: ActiveClean is an architecture where data cleaning is integrated with model training in a framework with sampling, model update, and feedback through estimation.**

When data cleaning is expensive, it is desirable to apply it **progressively**, where users can inspect early results with only $k \ll N$ records cleaned. Progressive data cleaning enables users to specify a cleaning budget and clean until that budget is reached. Even without a budget, progressive cleaning is useful as it allows users measure the impact of a potentially costly data cleaning operation without applying it to the entire data. However, when applied before predictive modeling, progressive data cleaning poses several methodological problems. Suppose $k$ records are cleaned, but all of the remaining dirty records are retained in the dataset. Training a model on a mixture of dirty and clean data can lead to misleading relationships in even simple scenarios (Figure 2). An alternative is to clean $k$ records and to disregard all of the remaining dirty records (e.g., sampling [42]). While this avoids the mixing problem, accurate model training may require a large amount of training data and $k$ examples may not be enough for a viable model. Finally, both problems are compounded by sparsity, where if corrupted records are uncommon, a random subset of $k$ records may have relatively few examples of corruptions. The errors and inefficiencies introduced by these three problems may dominate any gains from data cleaning, leading to unreliable or misleading conclusions about data or model quality.

We propose ActiveClean, a progressive data cleaning framework, that addresses the three methodological challenges: mixing, sampling, and sparsity. The key insight is that an important class of predictive models, called convex loss models (e.g., linear regression and SVMs), are trained by iteratively drawing random samples of data and updating a model [9]. Rather than cleaning before model training, data cleaning can be directly integrated into the training process, so that ActiveClean can ensure provable guarantees such as convergence and error bounds. In ActiveClean, data are cleaned in small random batches and the model is incrementally updated based on the results. Similar to Active Learning, ActiveClean selects the most valuable records to clean with higher probability, however, it applies a number of optimizations that exploit the data cleaning setting such as avoiding data that is expected to be clean, estimating of the effect of data cleaning for a record, and batching together updates from already cleaned data. This framework is optimized for problems that require expensive data cleaning.

The ActiveClean architecture (Figure 1) consists of a *detector*, *sampler*, *cleaner*, *update process*, and *estimator*. The cleaner is a user provided data cleaning technique (e.g., Entity Resolution), and ActiveClean provides the remaining components to apply the cleaner progressively. To summarize the contributions by component:

- **Detector** (Section 4). The detector can apply rules from data quality constraints or adaptively learn which records are dirty to increase the fraction of dirty records sampled.
- **Sampler** (Section 6). We derive an optimal sampling distribution that minimizes the update variance (i.e., how different would the update be if another sample was drawn) which linearly improves an error bound on the convergence rate.
- **Update** (Section 5). The update procedure applies a weighted stochastic gradient descent step to the current best model. This update is guaranteed to converge, and for batch size $b$ and iterations $T$, converges with rate $O(\frac{1}{\sqrt{bT}})$.
- **Estimator** (Section 7). The estimator applies a Taylor Series linearization to decouple changes in different features to use information from error detection to inform estimation.
- The experiments evaluate these components on four datasets with real and synthetic corruption (Section 8). Results suggests that for a fixed cleaning budget, ActiveClean returns more accurate models than uniform sampling and Active Learning when systematic corruption is sparse.

# 2. PROBLEM SETUP

This section describes an example of data cleaning when training predictive models, and formalizes the class of predictive models (convex loss) explored in this work.

## 2.1 Use Case: Dollars for Docs [2]

ProPublica collected a dataset of corporate donations to doctors to analyze conflicts of interest. They reported that some doctors received over $500,000 in travel, meals, and consultation expenses [4]. ProPublica laboriously curated and cleaned a dataset from the Centers for Medicare and Medicaid Services that listed nearly 250,000 research donations, and aggregated these donations by physician, drug, and pharmaceutical company. We collected the raw unaggregated data and explored whether suspect donation could be predicted with a model. This problem is typical of analysis scenarios based on observational data seen in finance, insurance, medicine, and investigative journalism.

The dataset has the following schema:

```
Contribution(pi_specialty, drug_name, device_name,
corporation, amount, dispute, status)
```

`pi_specialty` is a textual attribute describing the specialty of the doctor receiving the donation.
`drug_name` is the branded name of the drug in the research study (null if not a drug).
`device_name` is the branded name of the device in the study (null if not a device).
`corporation` is the name of the pharmaceutical providing the donation.
`amount` is a numerical attribute representing the donation amount.
`dispute` is a Boolean attribute describing whether the research was disputed.
`status` is a string label describing whether the donation was allowed under the declared research protocol. The goal is to predict disallowed donation.

However, this dataset is very dirty, and the systematic nature of the data corruption can result in an inaccurate model. On the ProPublica website [2], they list numerous types of data problems that had to be cleaned before publishing the data (see Appendix I). For example, the most significant donations were made by large companies whose names were also more often inconsistently represented in the data e.g., "Pfizer Inc.", "Pfizer Incorporated", "Pfizer". In a scenario such as this one, the effect of systematic error can be serious. Duplicate entity representations could artificially reduce the correlation between these entities and suspected contributions. There were nearly 40,000 of the 250,000 records that had either entity resolution issues or other inconsistencies in labeling the allowed or disallowed `status`. With the same Support Vector Machine model, the detection rate of suspect donations was 66% in the dirty data and 97% in the clean data (Section 8.6.1).

## 2.2 Progressive Data Cleaning

Cleaning a dataset of 250,000 records can be very time consuming, and it is important for analysts to be able to evaluate the model before all of the data is cleaned. First, suppose $k$ records are cleaned, but all of the remaining dirty records are retained in the dataset. Figure 2 highlights the dangers of this approach on a very simple dirty dataset and a linear regression model i.e., the best fit line for two variables. One of the variables is systematically corrupted with a translation in the x-axis (Figure 2a). The dirty data is marked in brown and the clean data in green, and their respective best fit lines are in blue. After cleaning only two of the data points (Figure 2b), the resulting best fit line is in the opposite direction of the true model. This is a well-known phenomenon called Simpsons paradox, where mixtures of different populations of data can result in spurious relationships [36]. Training models on a mixture of dirty and clean data can lead to unreliable results, where artificial trends introduced by the mixture can be confused for the effects of data cleaning.

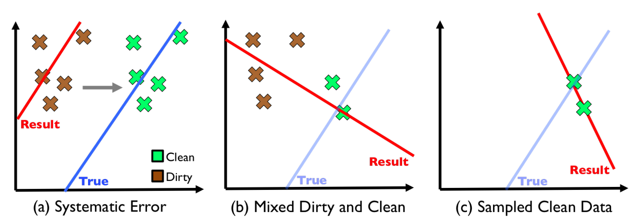


**Figure 2: (a) Systematic corruption in one variable can lead to a shifted model. (b) Mixed dirty and clean data results in a less accurate model than no cleaning. (c) Small samples of only clean data can result in similarly inaccurate models.**

An alternative is to avoid the dirty data altogether instead of mixing the two populations. Suppose $k$ records are randomly sampled from the dataset and cleaned. The model is trained only on the cleaned sample of data. This is similar to SampleClean [42], 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. Figure 2c illustrates that, even in two dimensions, models trained from small samples can be as incorrect as the mixing solution described before.

In this work, we propose a new methodology to avoid Simpson's Paradox and the strong dependence on sample size. Instead of mixing dirty and clean data, ActiveClean uses a model trained on the dirty data as an initialization, and then iteratively updates this model using samples of clean data. The intuition is that this algorithm smoothly transitions the model from one population (the dirty data) to another (the clean data), leading to provable guarantees about intermediate results.

## 2.3 Preliminaries: Convex Data Analytics

To design an algorithm that avoids Simpson's paradox, this work focuses on an initial class of well analyzed predictive analytics problems; ones that can be expressed as the minimization of convex loss functions. Convex loss minimization problems are amenable a variety of incremental optimization methodologies with provable guarantees (see Friedman, Hastie, and Tibshirani [18] for an introduction). Examples includes all generalized linear models (including linear and logistic regression), all variants of support vector machines, and in fact, means and medians are also special cases.

Formally, for labeled training examples $\{(x_i, y_i)\}_{i=1}^{N}$, the problem is to find a vector of *model parameters* $\theta$ by minimizing a loss function $\phi$ over all training examples:

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

Where $\phi$ is a convex function in $\theta$. For example, in a linear regression $\phi$ is:

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

Typically, a *regularization* term $r(\theta)$ is added to this problem. $r(\theta)$ penalizes high or low values of feature weights in $\theta$ to avoid overfitting to noise in the training examples.

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

In this work, without loss of generality, we will include the regularization as part of the loss function i.e., $\phi(x_i, y_i, \theta)$ includes $r(\theta)$.

DEFINITION 1 (CONVEX DATA ANALYTICS). *A convex data analytics problem is specified by a set of features $X$, corresponding set of labels $Y$, and a parametrized loss function $\phi$ that is convex in its parameter $\theta$. The result is a* ***model*** *$\theta$ that minimizes the sum of losses over all features and labels.*

## 2.4 ActiveClean Problem

The core problem addressed by ActiveClean is incremental model update while progressively cleaning data.

PROBLEM 1 (ACTIVECLEAN PROBLEM). *Let $R$ be a dirty relation, $F(r) \mapsto (x, y)$ be a featurization that maps a record $r \in R$ to a feature vector $x$ and label $y$, $\phi$ be a convex regularized loss, and $C(r) \mapsto r_{clean}$ be a cleaning technique that maps a record to its cleaned value. Given these inputs, the ActiveClean problem is to return a* ***reliable*** *estimate $\hat{\theta}$ of the clean model for any limit $k$ on the number of times the data cleaning $C(\cdot)$ can be applied.*

***Reliable*** *precisely means that the expected error in this estimate (i.e., L2 difference w.r.t a model trained on a fully cleaned dataset) is bounded above by a monotonically decreasing function in $k$ and the error of the dirty model.*

Addressing this problem requires analysis of the loss function $\phi$ and how $\phi$ is affected by dirty data. The solution is to integrate data cleaning and model training, where $\phi$ is simultaneously minimized while the data is cleaned. The tight feedback loop between model training and data cleaning poses several new algorithmic challenges and systems challenges. From an algorithmic perspective, the *reliability* of the estimates differentiate ActiveClean from alternative methodologies. We have to re-weight data to avoid biases and the population mixture problems described previously. There are also numerous new opportunities for optimizations such as pragmatically prioritizing data and avoiding data that are expected to be clean.

From a systems perspective, data cleaning and model training happen at very different time scales. When humans are involved, per record latencies for data cleaning are orders of magnitude larger than the CPU time needed for model training. We can compare recent results in data cleaning to a model training framework like CoCoA implemented on Spark [22]. Per record, BigDansing, a highly optimized automated Spark-based data cleaning system is 15.5x slower than CoCoA[1]. Crowd based techniques like CrowdFill [33] and CrowdER [41] are over 100,000x slower per record. Consequently, all of the optimizations in ActiveClean are designed to address data cleaning latency (i.e., more progress with fewer cleaned records) rather than optimizing for numerical computation (i.e., process fewer records).

There are two standard metrics that we will use to measure the performance of ActiveClean:

**Model Error.** Let $\theta$ be the model trained with ActiveClean, and let $\theta^*$ be the model trained on the same data if all of the records were cleaned. Then the model error is defined as $\|\theta - \theta^*\|$.

**Testing Error.** Let $\theta$ be the model trained with ActiveClean, and let $\theta^*$ be the model trained on the same data if all of the records were cleaned. Let $T(\theta)$ be the out-of-sample testing error when the dirty model is applied to the clean data, and $T(\theta^*)$ be the test error when the clean model is applied to the clean data. The testing error is defined as $T(\theta) - T(\theta^*)$

# 3. SYSTEM ARCHITECTURE

This section describes the ActiveClean architecture and the basic algorithmic framework. The individual components will be addressed in the subsequent sections.

## 3.1 Overview

Figure 1, in the introduction, overviews the entire framework. The first step of ActiveClean is *initialization*. In this step, there is a dirty relation $R$, a featurization $F(\cdot)$, a data cleaning technique $C(\cdot)$, and a dirty model $\theta^{(d)}$ trained on the featurized dirty relation. Optionally, ActiveClean integrates with dirty data detection rules $D(\cdot)$ which selects the set of likely corrupted records from $R$. If one is not provided, ActiveClean starts by treating all of the data as dirty and tries to learn a detector as data are cleaned. At initialization, there are two hyperparameters to set, the cleaning budget $k$ and the batch size $b$ (the number of iterations is $T = \frac{k}{b}$). We discuss how to set $b$ and the tradeoffs in setting a larger or smaller $b$ in Section 5.

After initialization, ActiveClean begins the cleaning and model update iterations. The *sampler* selects a sample of dirty data based on the batch size. At this step, ActiveClean can use the detector $D$ to narrow the sample to likely dirty data. Once a sample is selected, the *cleaner* applies $C(\cdot)$ to the dirty sample. ActiveClean is initialized with the dirty model, and this model is iteratively updated as more batches are cleaned.

The next two steps in the architecture are feedback steps where the sampling distribution is updated for the next iteration. The *estimator* uses previously cleaned data to estimate the value of data cleaning on new records. This information is used to guide sampling towards more valuable records. After estimation, the detector $D(\cdot)$ is also updated based on cleaned data. After all of the iterations are complete, the system returns the updated model.

[1] For CoCoA to reach a precision of 1e-3

To summarize the architecture in pseudocode:

1. `Init(dirty_data, cleaned_data, dirty_model, batch, iter)`
2. For each t in $\{1, ..., T\}$
   (a) `dirty_sample = Sampler(dirty_data, sample_prob, detector, batch)`
   (b) `clean_sample = Cleaner(dirty_sample)`
   (c) `current_model = Update(current_model, sample_prob, clean_sample)`
   (d) `cleaned_data = cleaned_data + clean_sample`
   (e) `dirty_data = dirty_data - clean_sample`
   (f) `sample_prob = Estimator(dirty_data, cleaned_data, detector)`
   (g) `detector = DetectorUpdater(detector, cleaned_data)`
3. `Output: current_model`

Here is an example application of ActiveClean:

EXAMPLE 1. *The analyst first trains an SVM model on the dirty data ignoring the effects of the errors resulting in a model $\theta^{(d)}$. 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). She initializes ActiveClean with $\theta^{(d)}$. ActiveClean samples an initial batch of 10 records. She manually cleans those records by merging similar drug names, making corporation names consistent, and fixing incorrect labels. After each batch, the model is updated with the most recent cleaning results $\theta^{(t)}$. The model improves after each iteration. After $t = 10$ of cleaning, the analyst has an accurate model trained with 100 cleaned records but still utilizes the entire dirty data.*

## 3.2 Challenges and Formalization

We highlight the important components and formalize the research questions explored in this paper.

**Detector (Section 4).** The first challenge in ActiveClean is dirty data detection. In this step, the detector select a candidate set of dirty records $R_{dirty} \subseteq R$. There are two techniques to do this: (1) an *a priori* case, and (2) and an adaptive case. In the *a priori* case, the detector knows which data is dirty in advance. In the adaptive case, the detector learns classifier based on previously cleaned data to detect corruption in uncleaned data.

**Sampler (Section 6).** The sampler draws a sample of records $S_{dirty} \subseteq R_{dirty}$. This is a non-uniform sample where each record $r$ has a sampling probability $p(r)$. We will derive the optimal sampling distribution, and show how the theoretical optimal can be approximated by the next estimator.

**Cleaner (User-Specified).** Given the sample of the records $S_{dirty}$, the cleaner applies the user-specified data cleaning $C(\cdot)$. This paper focuses on a record-by-record cleaning model where the function $C$ is applied to a record and produces the clean record:

$$S_{clean} = \{C(r) : \forall r \in S_{dirty}\}$$

This allows for uniform measure of the performance of ActiveClean in terms of model error as a function of sample size. The record-by-record cleaning model is not a fundamental restriction of this approach, and in the extensions (Section A.1), there is a discussion on a compatible "set of records" cleaning model. Consider the case where an analyst finds a dirty record, and is able to fix all records (possibly outside the sample) the with same error throughout the dataset efficiently.

**Update (Section 5).** This step updates the model $\theta^{(t)}$ based on the featurized (with featurization $F(\cdot)$) cleaned sample $F(S_{clean})$ resulting in $\theta^{(t+1)}$. Analyzing the model update procedure as a stochastic gradient descent algorithm will help derive the sampling distribution and estimation.

**Estimator (Section 7):** The estimator approximates the optimal distribution derived in the Sample step. Based on the change in the featurized data $F(S_{clean})$ and $F(S_{dirty})$, it directs the next iteration of sampling to select points that will have changes most valuable to the next model update.

# 4. DETECTION

To maximize the benefit of data cleaning, detection ensures that sampling draws records likely to be dirty.

## 4.1 Goals

The detector returns two important aspects of a record: (1) whether the record is dirty, and (2) if it is dirty, what is wrong with the record. The sampler can use (1) to select a subset of dirty records to sample at each batch. The estimator can use (2) estimate the value of data cleaning based on other records with the same corruption. ActiveClean supports two types of detectors: *a priori* and *adaptive*. In the *a priori* case, there is a way to select the set of dirty records before any cleaning. This case is possible for some types of data errors. In the adaptive case, detection is learned as data is cleaned.

## 4.2 *A Priori* Case

For many types of dirtiness such as missing attribute values and constraint violations, it is possible to efficiently enumerate a set of corrupted records and determine how the records are corrupted.

DEFINITION 2 (A PRIORI DETECTION). *Let $r$ be a record in $R$. An a priori detector is a detector that returns a Boolean of whether the record is dirty and a set of columns $e_r$ that are dirty.*

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

*From the set of columns that are dirty, find the corresponding features that are dirty $f_r$ and labels that are dirty $l_r$.*

Here are example use cases of this definition using data cleaning methodologies proposed in the literature.

**Constraint-based Repair:** One model for detecting errors involves declaring constraints on the database.

*Detection.* Let $\Sigma$ be a set of constraints on the relation $\mathcal{R}$. In the detection step, the detector selects a subset of records $\mathcal{R}_{dirty} \subseteq \mathcal{R}$ that violate at least one constraint. The set $e_r$ is the set of columns for each record which have a constraint violation.

EXAMPLE 2. *An example of a constraint on the running example dataset is that the `status` of a contribution can be only "allowed" or "disallowed". Any other value for `status` is an error.*

**Entity Resolution:** Another common data cleaning task is Entity Resolution [19,25,41]. Entity Resolution is the problem of standardizing attributes that represent the same real world entity. A common pattern in Entity Resolution is to split up the operation into two steps: blocking and matching. In blocking, attributes that should be the same are coarsely grouped together. In matching, those coarse groups are resolved to a set of distinct entities.

*Detection.* Detection for entity resolution problems is the matching step. Let $S$ be a similarity function that takes two records and returns a value in $[0, 1]$ (1 most similar and 0 least similar). For some threshold $t$, $S$ defines a similarity relationship between two attributes $r(a)$ and $r'(a)$:

$$r(a) \approx r'(a) : S(r(a), r'(a)) \geq t$$

In the detection step, $R_{dirty}$ is the set of records that have at least one other record in the relation that satisfies $r(a) \approx r(a)'$. The set $e_r$ is the set of attributes of $r$ that have entity resolution problems.

EXAMPLE 3. *An example of an Entity Resolution problem is seen in our earlier example about corporation names e.g. "Pfizer Inc.", "Pfizer Incorporated", "Pfizer".. Given a similarity relationship* $WeightedJaccard(r1, r2) > 0.8$*, the detector selects all records that satisfy this condition (their Weighted Jaccard Similarity is greater than 0.8).*

## 4.3 Adaptive Detection

*A priori* detection is not possible in all cases. The detector also supports adaptive detection where detection is learned from previously cleaned data. Note that this "learning" is distinct from the "learning" at the end of the pipeline. The challenge in formulating this problem is that detector needs to describe how the data is dirty (e.g. $e_r$ in the *a priori* case). The detector achieves this by categorizing the corruption into $u$ classes. These classes are corruption categories that do not necessarily align with features, but every record is classified with at most one category. For example, suppose there are records with outliers and missing values, there are three classes of corruption: outliers, missing values, and both.

While the number of corruption classes is combinatorial in the individual types, the only classes that need to enumerated are ones that are manifest in the data. 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, $1, 2, ..., u + 1$ classes (including one for "not dirty"). It is possible that $u$ increases each iteration as more types of dirtiness are discovered. Then, the detection problem reduces to a multiclass classification problem. This problem can be addressed by any multiclass classifier, and we use an all-versus-one SVM in our experiments.

DEFINITION 3 (ADAPTIVE CASE). *Select the set of records for which* $\kappa$ *gives a positive error classification (i.e., one of the* $u$ *error classes). After each sample of data is cleaned, the classifier* $\kappa$ *is retrained. So the result is:*

$$D(r) = (\{1, 0\}, \{1, ..., u + 1\})$$

**Adaptive Detection With OpenRefine:**

EXAMPLE 4. *OpenRefine is a spreadsheet-based tool that allows users to explore and transform data. However, it is limited to cleaning data that can fit in memory on a single computer. Since the cleaning operations are coupled with data exploration, ActiveClean does not know what is dirty in advance (the analyst may discover new errors as she cleans).*

*Suppose the analyst wants to use OpenRefine to clean the running example dataset with ActiveClean. She takes a sample of data from the entire dataset and uses the tool to discover errors. For example, she finds that some drugs are incorrectly classified as both drugs and devices. She then removes the device attribute for all records that have the drug name in question. As she fixes the records, she tags each one with a category tag of which corruption it belongs to.*

# 5. UPDATE

Before discussing sampling, this section discusses how to update a model. The sampling framework will naturally follow from this update procedure. This section assumes that the detector in the previous section has selected a set of candidate records $R_{dirty}$ and the sampler has sampled from this set of candidate records. This section shows that this model update procedure can be interpreted as a Stochastic Gradient Descent (SGD) algorithm, which gives a theoretical framework to analyze convergence and bound the error at each step.

## 5.1 Update Problem

Let $S_{dirty} \subseteq R_{dirty}$ be a random sample of data from the results of the detection in the previous section, and let $p(r)$ be the sampling probability of each record. The cleaner applies the data cleaning technique to the sample resulting in $S_{clean}$ and clean features and labels $(X^{(c)}, Y^{(c)})$. In the model update problem, ActiveClean applied the update to the dirty model $\theta^{(d)}$, based on the clean sample, to return $\theta^{new}$. The goal is that these updates should minimize the error of the updated model and the true model $\theta^{(c)}$ (if entire data is cleaned):

$$error(\theta^{new}) = \|\theta^{new} - \theta^{(c)}\|$$

## 5.2 Geometric Interpretation

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

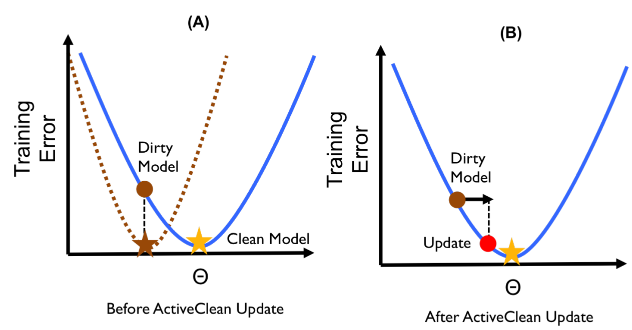


**Figure 3: (A) A model trained on dirty data can be thought of as a sub-optimal point w.r.t to the clean data. (B) The gradient gives us the direction to move the suboptimal model to approach the true optimum.**

The optimal clean model $\theta^{(c)}$ is visualized as a yellow star. The first question is which direction to update $\theta$ (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 $d$-dimensional vector function of the current model $\theta$ and the clean data. Given this direction, ActiveClean needs to update $\theta^{(d)}$ some distance $\gamma$ (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, or corrects 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 this gradient from a sample. The intuition, formalized in Section 5.5, is that if the gradients are on average in the right direction, the algorithm is guaranteed to converge with bounds on the convergence rate.

## 5.3 Average Gradient From a Sample

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 $\theta$, the gradient $g^*(\theta)$ is:

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

Therefore, the gradient 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. Let $S$ be a sample of data, where each $i \in S$ is drawn with probability $p(i)$:

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

Now adding iteration, for every batch of data cleaned $t$, the update to the current best model estimate is:

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

The detection in the previous step adds a small complication, since the sample $S_{dirty}$ is not representative of all of the data. This requires a compensation for this bias by averaging this estimate with the gradient of the complement:

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

Then, for weights $\alpha, \beta$ (discussed in Section 5.5):

$$g(\theta) = \alpha \cdot g_C(\theta) + \beta \cdot g_S(\theta)$$

Finally, adding in the iteration, and at each iteration $t$, the update becomes:

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

## 5.4 Model Update Algorithm

To summarize, the algorithm is initialized with $\theta^{(1)} = \theta^{(d)}$ which is the dirty model. 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.

1. Calculate the gradient over the sample of clean data and call the result $g_S(\theta^{(t)})$
2. Calculate the average gradient over all the data in $R_{clean} = R - R_{dirty}$, and call the result $g_C(\theta^{(t)})$
3. Apply the following update rule:
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \lambda \cdot (\alpha \cdot g_S(\theta^{(t)}) + \beta \cdot g_C(\theta^{(t)}))$$

To summarize the parameters $b$, $\lambda$, $\alpha$, $\beta$.

**Batch Size $b$ :** The batch size $b$ controls the frequency of iteration. Larger batches provide a more accurate estimate of the gradient at each iteration but has less frequent model updates.

**Step Size $\gamma$ :** $\gamma$ controls how far should to travel in the gradient direction.

**Weights $\alpha, \beta$ :** $\alpha, \beta$ are the proportions to combine $g_S(\theta)$ and $g_C(\theta)$.

## 5.5 Analysis with Stochastic Gradient Descent

This update policy can be formalized as a class of very well studied algorithms called Stochastic Gradient Descent. This provides a theoretical framework to understand and analyze the update rule, bound the error, and choose points to clean. Mini-batch stochastic gradient descent (SGD) is an algorithm for finding the optimal value given the convex loss and data. In mini-batch SGD, random subsets of data are selected at each iteration and the average gradient is computed for every batch. The key difference is that in traditional SGD there is no notion of dirty and clean data.

**ActiveClean as Lazy SGD:** This update method is a variant of SGD that lazily materializes the clean value. As data is sampled at each iteration, data is cleaned when needed by the optimization. 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) [10]. Furthermore in this setting, the dirty model can be much more accurate than an arbitrary initialization; leading to highly accurate models without cleaning the entire data.

**Deriving $\alpha$ and $\beta$:** The first problem is to choose $\alpha$ and $\beta$ such that the estimate of the gradient is unbiased. SGD will converge when the estimate is unbiased and the step-size is chosen appropriately. The batch $S_{dirty}$ is drawn only from $R_{dirty}$. Since the sizes of $R_{dirty}$ and its complement are known, it follows that the gradient over the already clean data $g_C$ and the recently cleaned data $g_S$ can be combined as follows:

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

Therefore,

$$\alpha = \frac{\mid R_{clean} \mid}{\mid R \mid}, \beta = \frac{\mid R_{dirty} \mid}{\mid R \mid d}$$

LEMMA 1. *The gradient estimate $g(\theta)$ is unbiased if $g_S$ is an unbiased estimate of:*

$$\frac{1}{\mid R_{dirty} \mid}\sum g_i(\theta)$$

PROOF SKETCH. This result follows directly from the linearity of expectation (See Appendix B). □

**Setting $\gamma$:** There is extensive literature in machine learning for choosing the step size $\gamma$ appropriately. $\gamma$ 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 learning rates or provide different learning scheduling frameworks. 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 $\gamma_t = \frac{\gamma_0}{|S|t}$.

**Convergence:** Convergence properties of batch SGD formulations have been well studied [13]. From the preceding analysis, the gradient estimate is unbiased and the step size is appropriately chosen. Following from these two points, convergence is ensured:

PROPOSITION 1. *For an appropriately chosen learning rate $\gamma_t$, ActiveClean will converge if $\mathbb{E}(g_S) = g^*$.*

**Convergence Rate:** The convergence rates of SGD are also well analyzed [9,13,44]. This gives a bound on the error of intermediate models and the expected number of steps before achieving a model within a certain error.

PROPOSITION 2. *For a general convex loss, a batch size b, and $T$ iterations, the convergence rate is bounded by $O(\frac{\sigma^2}{\sqrt{bT}})$. $\sigma^2$ is the variance in the estimate of the gradient at each iteration:*

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

**Setting the batch size:** 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 but make less progress. The overheads introduced by ActiveClean are more evident at smaller batch sizes. There are diminishing returns to increasing the batch size $O(\frac{1}{\sqrt{b}})$. Empirically, in the experiments, batch sizes of 50 converge the fastest. If a data cleaning technique requires a larger batch size than this, i.e., data cleaning is fast enough that the iteration overhead is significant compared to cleaning 50 records, ActiveClean can 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. This dissociates the batching requirements of SGD and the batching requirements of the data cleaning technique.

**Non-convex losses:** If the loss in non-convex, the update procedure will converge towards a local minimum rather than the global minimum (See Appendix A.2).

# 6. SAMPLING

In the previous section, the model update received a sample with probabilities $p(r)$. This section provides a derivation for an optimal sampling problem that directly follows from the analysis of the update rule via SGD. It will turn out that the solution to the optimal sampling problem is not realizable in practice (as it depends on knowing the cleaned value), and this problem will be addressed with an approximation in the next section.

## 6.1 Goals and Challenges

In the Machine Learning and Optimization literature, SGD algorithms are optimized to avoid scanning the entire data. Uniform sampling is cheap so it is the preferred solution. However, data cleaning costs can be many orders of magnitude higher than model training. As a result, uniform sampling may not be the most efficient option. ActiveClean can sacrifice computational overhead by precomputing some results over the entire data for savings during the data cleaning phase. This problem is formulated as an optimal sampling problem to compute the sampling probabilities $p(r)$ that maximize the convergence rate.

## 6.2 Optimal Sampling Problem

Recall that the convergence rate of an SGD algorithm is bounded by $\sigma^2$ 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 sample 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 4 (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:*

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

To construct these sampling probabilities, first consider the following lemma about importance sampling. This lemma describes the optimal distribution over a set of scalars:

LEMMA 2. *Given a set of real numbers $A = \{a_1, ..., a_n\}$, let $\hat{A}$ be a sample with replacement of $A$ of size $k$. If $\mu$ is the mean $\hat{A}$, the sampling distribution that minimizes the variance of $\mu$, i.e., the expected square error, is $p(a_i) \propto a_i$.*

PROOF SKETCH. This proof follows from [31], as it is a straightforward importance sampling result. We include the proof in the appendix (Section D) □

Lemma 2 shows that when estimating a mean of numbers with sampling, the distribution with optimal variance is sampling proportionally to the values. This insight leads to a direct higher-dimensional generalization, where at iteration $t$ the optimal distribution over records in $R_{dirty}$ is probabilities proportional to:

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

However, in this case, it leads to a chicken-and-egg problem. The optimal sampling distribution requires knowing $(x_i^{(c)}, y_i^{(c)})$, however, cleaning is required to know those values. In the next section, the estimator will approximate this distribution by estimating the cleaned value with previously cleaned data. Since the model update can work with *any* distribution, convergence is guaranteed no matter how inaccurate this approximation is. However, a better approximation will lead to an improved convergence rate.

# 7. ESTIMATION

This section makes the sampling result of the previous section practical by approximating the cleaned values.

## 7.1 Goals and Challenges

The optimal sampling distribution is dependent on a value that is not known without data cleaning $\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta^{(t)})$. One way to approximate this distribution is to learn a function $e(\cdot)$ via regression based on previously cleaned data. This is a high-dimensional regression problem which may have to learn a very complicated relationship between dirty and clean data. The biggest challenge with such an estimator is the cold start problem, where if given a small amount of cleaned data, the estimator will be inaccurate. ActiveClean should be optimized to make as much progress as possible in the early iterations so this technique may not work. The estimator takes an alternative approach where it exploits information from the detector to produce an estimate for groups of similarly corrupted records.

## 7.2 Estimation For A Priori Detection

EXAMPLE 5. *Suppose records from running example dataset are corrupted with both entity resolution problems, missing data, and constraint violations. Each training example will have a set of corrupted features (e.g., $\{1, 2, 6\}$, $\{1, 2, 15\}$).*

*Suppose that the cleaner has just cleaned the records $r_1$ and $r_2$ represented as tuples with their corrupted feature set: ($r_1$,$\{1, 2, 3\}$), ($r_2$,$\{1, 2, 6\}$). Then, given a new record ($r_3$,$\{1, 2, 3, 6\}$). The estimator should be able to use the cleaning results from $r_1, r_2$ to estimate the gradient in $r_3$.*

If most of the features are correct, it would seem like the gradient is only incorrect in one or two of its components. 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, the gradient for linear regression is:

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

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 will be incorrect.

### 7.2.1 Error Decoupling

To address this problem, the gradient can be approximated in a way that the effects of dirty features on the gradient are decoupled. Recall, in the *a priori* detection problem, that associated with each $r \in R_{dirty}$ is a set of errors $f_r, l_r$ which is a set that identifies a set of corrupted features and labels. This property can be used to construct a coarse estimate of the clean value. The main idea is to calculate average changes for each feature, then given an uncleaned (but dirty) record, add these average changes to correct the gradient.

To formalize this intuition, instead of computing the actual gradient with respect to the true clean values, compute the conditional expectation given that a set of features and labels $f_r, l_r$ are corrupted:

$$p_i \propto \mathbb{E}(\nabla\phi(x_i^{(c)}, y_i^{(c)}, \theta^{(t)}) \mid f_r, l_r)$$

Corrupted features defined as that:

$$i \notin f_r \implies x^{(c)}[i] - x^{(d)}[i] = 0$$
$$i \notin l_r \implies y^{(c)}[i] - y^{(d)}[i] = 0$$

The needed approximation represents a linearization of the errors, and the resulting approximation will be of the form:

$$p_r \propto \|\nabla\phi(x, y, \theta^{(t)}) + M_x \cdot \Delta_{rx} + M_y \cdot \Delta_{ry}\|$$

where $M_x$, $M_y$ are matrices and $\Delta_{rx}$ and $\Delta_{ry}$ are average change vectors for the corrupted features in $r$. Without this approximation, calculating the expected value conditioned on $f_r, l_r$ would require conditioning on all the combinatorial possibilities.

### 7.2.2 *Deriving $M_x$, $M_y$*

If $d$ is the dirty value and $c$ is the clean value, the Taylor series approximation for a function $f$ is given as follows:

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

Ignoring the higher order terms, the linear term $f'(d) \cdot (d - c)$ is a linear function in each feature and label. Then, taking expected values, it follows that:

$$\approx \nabla\phi(x, y, \theta) + M_x \cdot \mathbb{E}(\Delta x) + M_y \cdot \mathbb{E}(\Delta y)$$

where $M_x = \frac{\partial}{\partial X}\nabla\phi$ and $M_y = \frac{\partial}{\partial Y}\nabla\phi$ (See Appendix E for derivation). Recall that the feature space is $d$ dimensional and label space is $l$ dimensional. Then, $M_x$ is an $d \times d$ matrix, and $M_y$ is a $d \times l$ matrix. Both of these matrices are computed for each record (see Appendix F for an example derivation). $\Delta x$ is a $d$ dimensional vector where each component represents a change in that feature and $\Delta y$ is an $l$ dimensional vector that represents the change in each of the labels.

### 7.2.3 *More Accurate Early Error Estimates*

Linearization over avoids amplifying estimation error. Consider the linear regression gradient:

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

This can be rewritten as a vector in each component:

$$g[i] = \sum_i x[i]^2 - x[i]y + \sum_{j \neq i} \theta[j]x[j]$$

This function is already mostly linear in $x$ except for the one quadratic term. However, this one quadratic term has potential to amplify errors. Consider two expressions:

$$f(x + \epsilon) = (x + \epsilon)^2 = x^2 + 2x\epsilon + \epsilon^2$$
$$f(x + \epsilon) \approx f(x) + f'(x)(\epsilon) = x^2 + 2x\epsilon$$

The only difference between the two estimates is the quadratic $\epsilon^2$, if $\epsilon$ is highly uncertain random variable then the quadratic dominates. If this variance is large, the Taylor estimate avoids amplifying this error. Of course, this is at the tradeoff of some additional bias since the true function is non-linear. We evaluate this linearization in Section 8.5 against alternatives, and find that indeed it provides more accurate estimates for a small number of samples cleaned. When the number of cleaned samples is large the alternative techniques are comparable or even slightly better.

### 7.2.4 *Maintaining Decoupled Averages*

This linearization allows ActiveClean to maintain per feature (or label) average changes and use these changes to center the optimal sampling distribution around the expected clean value. To estimate $\mathbb{E}(\Delta x)$ and $\mathbb{E}(\Delta y)$, consider the following lemma:

LEMMA 3 (SINGLE FEATURE). *For a feature i, we average all $j = \{1, ..., K\}$ records cleaned that have an error for that feature, weighted by their sampling probability:*

$$\bar{\Delta}_{xi} = \frac{1}{NK} \sum_{j=1}^{K} (x^{(d)}[i] - x^{(c)}[i]) \times \frac{1}{p(j)}$$

*Similarly, for a label i:*

$$\bar{\Delta}_{yi} = \frac{1}{NK} \sum_{j=1}^{K} (y^{(d)}[i] - y^{(c)}[i]) \times \frac{1}{p(j)}$$

Each $\bar{\Delta}_{xi}$ and $\bar{\Delta}_{yi}$ represents an average change in a single feature. A single vector can represent the necessary changes to apply to a record $r$:

LEMMA 4 (DELTA VECTOR). *For a record r, the set of corrupted features is $f_r, l_r$. Then, each record r has a d-dimensional vector $\Delta_{rx}$ which is constructed as follows:*

$$\Delta_{rx}[i] = \begin{cases} 0 & i \notin f_r \\ \bar{\Delta}_{xi} & i \in f_r \end{cases}$$

*Each record r also has an l-dimensional vector $\Delta_{ry}$ which is constructed as follows:*

$$\Delta_{rx}[i] = \begin{cases} 0 & i \notin l_r \\ \bar{\Delta}_{yi} & i \in l_r \end{cases}$$

Finally, the result is:

$$p_r \propto \|\nabla\phi(x, y, \theta^{(t)}) + M_x \cdot \Delta_{rx} + M_y \cdot \Delta_{ry}\| \blacksquare$$

## 7.3 Estimation For Adaptive Case

A similar procedure holds in the adaptive setting, however, it requires reformulation of "similarly corrupted". Here, ActiveClean uses $u$ corruption classes provided by the detector. Instead of conditioning on the features that are corrupted, the estimator conditions on the classes. So for each error class, it computes a $\Delta_{ux}$ and $\Delta_{uy}$. These are the average change in the features given that class and the average change in labels given that class.

$$p_{r,u} \propto \|\nabla\phi(x, y, \theta^{(t)}) + M_x \cdot \Delta_{ux} + M_y \cdot \Delta_{uy}\| \blacksquare$$

# 8. EXPERIMENTS

First, the experiments evaluate how various types of corrupted data benefit from data cleaning. Next, the experiments explore different prioritization and model update schemes for progressive data cleaning. Finally, ActiveClean is evaluated end-to-end in a number of real-world data cleaning scenarios.

## 8.1 Experimental Setup and Notation

The main metric for evaluation is a relative measure of the trained model and the model if all of the data is cleaned.

**Relative Model Error.** Let $\theta$ be the model trained on the dirty data, and let $\theta^*$ be the model trained on the same data if it was cleaned. Then the model error is defined as $\frac{\|\theta - \theta^*\|}{\|\theta^*\|}$.

### 8.1.1 *Scenarios*

**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 completely clean data of only 65%.

**Handwriting Recognition (MNIST)** [2]**:** In this dataset, the task is to classify 60,000 images of handwritten images into 10 categories with an one-to-all multiclass SVM classifier. The unique part of this dataset is the featurized data consists of a 784 dimensional vector which includes edge detectors and raw image patches.

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

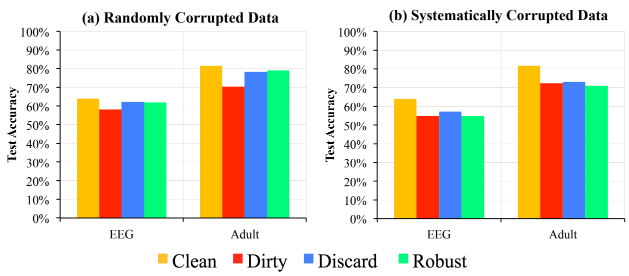


**Figure 4: (a) Robust techniques and discarding data work when corrupted data are random and look atypical. (b) Data cleaning can provide reliable performance in both the systematically corrupted setting and randomly corrupted setting.**

**Dollars For Docs:** The dataset has 240,089 records with 5 textual attributes and one numerical attribute. The dataset is featurized with bag-of-words featurization model for the textual attributes which resulted in a 2021 dimensional feature vector, and a binary SVM is used to classify the status of the medical donations.

### *8.1.2 Compared Algorithms*

Here are the alternative methodologies evaluated in the experiments:

**Robust Logistic Regression [17].** Feng et al. proposed a variant of logistic regression that is robust to outliers. 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. (See details in Appendix H.1)

**Discarding Dirty Data.** As a baseline, dirty data is discarded.

**SampleClean (SC) [42].** SampleClean takes a sample of data, applies data cleaning, and then trains a model to completion on the sample.

**Active Learning (AL) [20].** An Active Learning algorithm that integrates with stochastic optimization (See details in Appendix H.2).

**ActiveClean Oracle (AC+O):** In ActiveClean Oracle, instead of an estimation step, the true clean value is used to evaluate the theoretical ideal performance of ActiveClean.

## 8.2 Does Data Cleaning Matter?

The first experiment evaluates the benefits of data cleaning on two of the example datasets (EEG and Adult). This is done without sampling 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). It is important to note that examples can have multiple corrupted features.

Figure 4 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. 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. In the remaining experiments, unless otherwise noted, the experiments use systematic corruption.

*Summary: A 5% systematic corruption can introduce a 10% reduction in test accuracy even when using a robust method.*

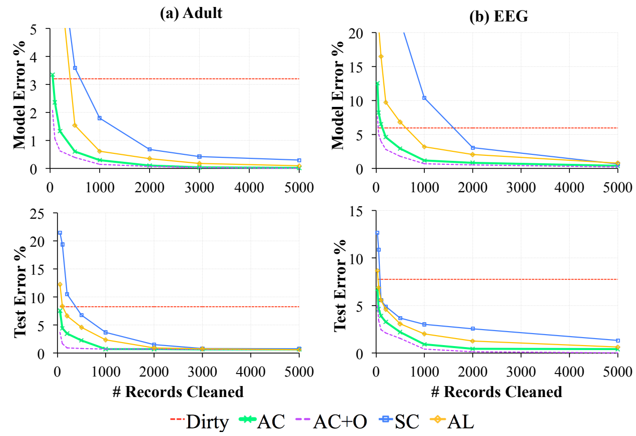


**Figure 5: The relative model error as a function of the number of examples cleaned. ActiveClean converges with a smaller sample size to the true result in comparison to Active Learning and SampleClean.**

## 8.3 ActiveClean: *A Priori* Detection

The next set of experiments evaluate different approaches to cleaning a sample of data compared to ActiveClean using *a priori* detection. *A priori* detection assumes that all of the corrupted records are known in advance but their clean values are unknown.

### *8.3.1 Active Learning and SampleClean*

The next experiment evaluates the samples-to-error tradeoff between four alternative algorithms: ActiveClean (AC), SampleClean, Active Learning, and ActiveClean +Oracle (AC+O). Figure 5 shows the model error and test accuracy as a function of the number of cleaned records. In terms of model error, ActiveClean gives its largest benefits for small sample sizes. For 500 cleaned records of the Adult dataset, ActiveClean has 6.1x less error than SampleClean and 2.1x less error than Active Learning. For 500 cleaned records of the EEG dataset, ActiveClean has 9.6x less error than SampleClean and 2.4x less error than Active Learning. Both Active Learning and ActiveClean benefit from the initialization with the dirty model as they do not retrain their models from scratch, and ActiveClean improves on this performance with detection and error estimation. Active Learning has no notion of dirty and clean data, and therefore prioritizes with respect to the dirty data. These gains in model error also correlate well to improvements in test error (defined as the test accuracy difference w.r.t cleaning all data). The test error converges more quickly than model error, emphasizing the benefits of progressive data cleaning, since it is not neccessary to clean all the data to get a model with essentially the same performance as the clean model. For example, to achieve a test error of 1% on the Adult dataset, ActiveClean cleans 500 fewer records than Active Learning.

*Summary: ActiveClean with a priori detection returns results that are more than 6x more accurate than SampleClean and 2x more accurate than Active Learning for cleaning 500 records.*

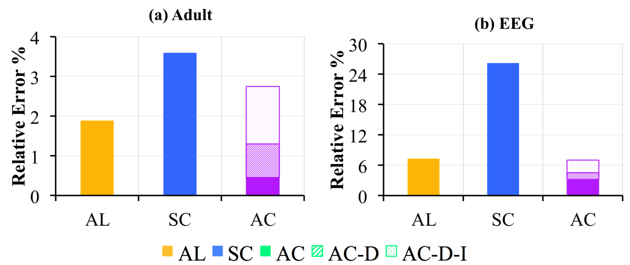


**Figure 6: -D denotes no detection, and -D-I denotes no detection and no importance sampling. Both optimizations significantly help ActiveClean outperform SampleClean and Active Learning.**

### 8.3.2 *Source of Improvements*

The next experiment compares the performance of ActiveClean with and without various optimizations at 500 records cleaned point. This is a vertical slice of the plots in the previous experiments. 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 importance sampling is denoted as (AC-D-I). Figure 6 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 importance sampling, ActiveClean is slightly worse than Active Learning on the Adult dataset but is comparable on the EEG dataset.

*Summary: Both a priori detection and non-uniform sampling significantly contribute to the gains over Active Learning.*

### 8.3.3 *Mixing Dirty and Clean Data*

Training a model on mixed data is an unreliable methodology lacking the same guarantees as Active Learning or SampleClean even in the simplest of cases. For thoroughness, the next experiments include the model error as a function of records cleaned in comparison to ActiveClean. Figure 7 plots the same curves as the previous experiment comparing ActiveClean, Active Learning, and two mixed data algorithms. PC randomly samples data, clean, and writes-back the cleaned data. PC+D randomly samples data from using the dirty data detector, cleans, and writes-back the cleaned data. For these errors PC and PC+D give reasonable results (not always guaranteed), but ActiveClean converges faster. This is because ActiveClean tunes the weighting when averaging dirty and clean data into the gradient.

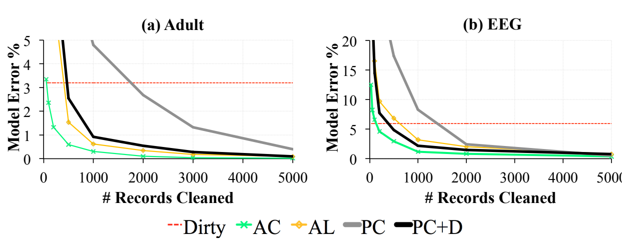


**Figure 7: The relative model error as a function of the number of examples cleaned. ActiveClean converges with a smaller sample size to the true result in comparison to partial cleaning (PC,PC+D).**

*Summary: ActiveClean converges faster than mixing dirty and clean data since it reweights data based on the fraction that is dirty and clean. Partial cleaning is not guaranteed to give sensible results.*

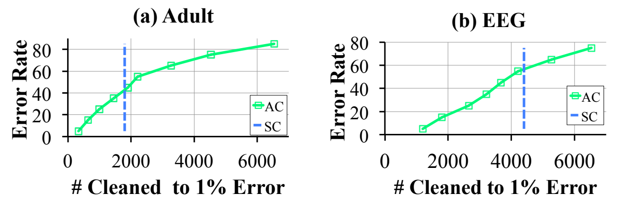


**Figure 8: ActiveClean performs well until the corruption is so severe that the dirty model is not a good initialization. The error of SampleClean does not depend on the corruption rate so it is a vertical line.**

### 8.3.4 *Corruption Rate*

The next experiment explores how much of the performance is due to the initialization with the dirty model (i.e., SampleClean trains a model from “scratch"). Figure 8 varies the systematic corruption rate and plots the number of records cleaned to achieve 1% relative error for SampleClean and ActiveClean. SampleClean does not use the dirty data and thus its error is essentially governed by the Central Limit Theorem. SampleClean 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.

*Summary: SampleClean is beneficial in comparison to ActiveClean when corruption rates exceed 45%.*

## 8.4 ActiveClean: Adaptive Detection

This experiment explores how the results of the previous experiment change when using an adaptive detector instead of the *a priori* detector. Recall, in the systematic corruption, 3 of the most informative features were corrupted, thus we group these problems into 9 classes. We use an all-versus-one SVM to learn the categorization.

### 8.4.1 *Basic Performance*

Figure 9 overlays the convergence plots in the previous experiments with a curve (denoted by AC+C) that represents ActiveClean using a classifier instead of the *a priori* detection. Initially ActiveClean is comparable to Active Learning; however, as the classifier becomes more effective the detection improves the performance. Over both datasets, at the 500 records point on the curve, adaptive ActiveClean has a 30% higher model error compared to *a priori* ActiveClean. At 1000 records point on the curve, adaptive ActiveClean has about 10% higher error.

*Summary: For 500 records cleaned, adaptive ActiveClean has a 30% higher model error compared to a priori ActiveClean, but still outperforms Active Learning and SampleClean.*

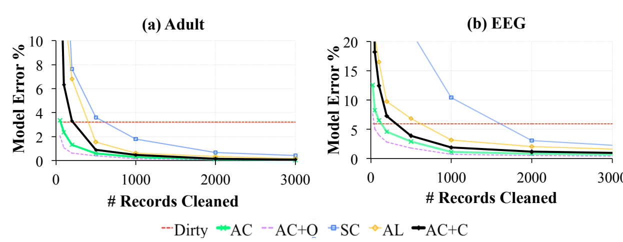


**Figure 9: Even with a classifier ActiveClean converges faster than Active Learning and SampleClean.**

### 8.4.2 *Classifiable Errors*

The adaptive case depends on being able to predict corrupted records. 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. The next experiment explores making the systematic corruption 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 at one vertical slice of the previous plot (cleaning 1000 records). 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. This is because the classifier is imperfect and misclassifies some data points incorrectly as cleaned.

*Summary: When errors are increasingly random (50% random) and cannot be accurately classified, adaptive detection provides no benefit over no detection.*

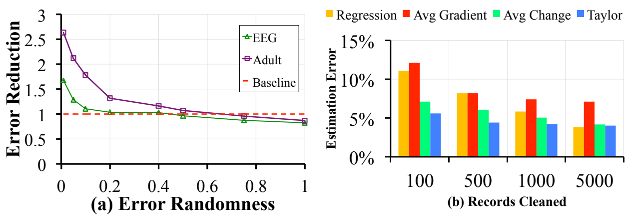


**Figure 10: (a) Data corruptions that are less random are easier to classify, and lead to more significant reductions in relative model error. (b) The Taylor series approximation gives more accurate estimates when the amount of cleaned data is small.**

## 8.5 Estimation

The next 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 cleaned 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, confirming the analysis in Section 7.2.3. Linear regression and the average feature change technique do eventually perform comparably but only after cleaning much more data.

*Summary: Linearized gradient estimates are more accurate when estimated from small samples.*

## 8.6 Real World Scenarios

The next set of experiments evaluate ActiveClean in two real world scenarios, one demonstrating the *a priori* case and the other for the adaptive detection case.

### 8.6.1 A Priori: Constraint Cleaning

The first scenario explores the Dollars for Docs dataset published by ProPublica described throughout the paper. To run this experiment, the entire dataset was cleaned up front, and simulated sampling from the dirty data and cleaning by looking up the value in the cleaned data (see Appendix I for constraints, errors, and cleaning methodology). Figure 11a shows that ActiveClean converges faster than Active Learning and SampleClean. To achieve a 4% relative error (i.e., a 75% error reduction from the dirty model), ActiveClean cleans 40000 fewer records than Active Learning. Also, for 10000 records cleaned, ActiveClean has nearly an order of magnitude smaller error than SampleClean.

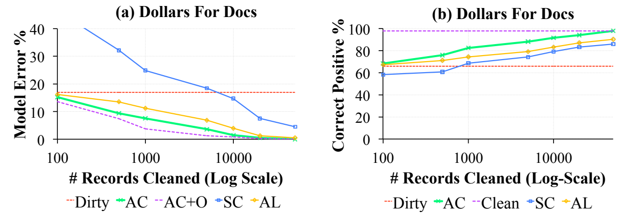


**Figure 11: (a) The relative model error as a function of the number of cleaned records. (b) The true positive rate as a function of the number of cleaned records.**

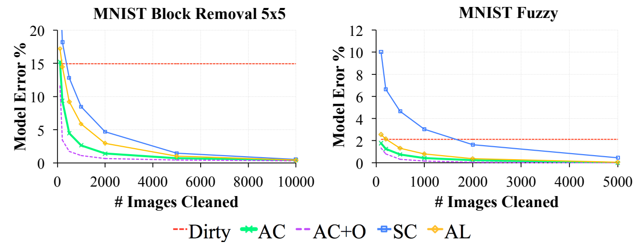


**Figure 12: In a real adaptive detection scenario with the MNIST dataset, ActiveClean outperforms Active Learning and SampleClean.**

Figure 11b shows the detection rate (fraction of disallowed research contributions identified) of the classifier as a function of the number of records cleaned. On the dirty data, we can only correctly classify 66% of the suspected examples (88% overall accuracy due to a class imbalance). On the cleaned data, this classifier is nearly perfect with a 97% true positive rate (98% overall accuracy). ActiveClean converges to the cleaned accuracy faster than the alternatives with a classifier of 92% true positive rate for only 10000 records cleaned.

*Summary: To achieve an 80% detection rate, ActiveClean cleans nearly 10x less records than Active Learning.*

### 8.6.2 Adaptive: Replacing Corrupted Data

The next experiment explores the MNIST handwritten digit recognition dataset with a MATLAB image processing pipeline. In this scenario, the analyst must inspect a potentially corrupted image and replace it with a higher quality one. The MNIST dataset consists of 64x64 grayscale images. There are two types of simulated corruptions: (1) 5x5 block removal where a random 5x5 block is removed from the image by setting its pixel values to 0, and (2) Fuzzy where a 4x4 moving average patch is applied over the entire image. These corruptions are applied to a random 5% of the images, and mimic the random (Fuzzy) vs. systematic corruption (5x5 removal) studied in the previous experiments. The adaptive detector uses a 10 class classifier (one for each digit) to detect the corruption.

Figure 12 shows that ActiveClean makes more progress towards the clean model with a smaller number of examples cleaned. To achieve a 2% error for the block removal, ActiveClean can inspect 2200 fewer images than Active Learning and 2750 fewer images than SampleClean. For the fuzzy images, both Active Learning and ActiveClean reach 2% error after cleaning fewer than 100 images, while SampleClean requires 1750.

*Summary: In the MNIST dataset, ActiveClean significantly reduces (more than 2x) the number of images to clean to train a model with 2% error.*

# 9. RELATED WORK

**Data Cleaning and Databases:** There are also several recent results in data cleaning that we would like to highlight. Progressive data cleaning methodologies have been proposed, however, these techniques tend to be application agnostic [29]. Altowim et al. proposed a framework for progressive entity resolution [7]. Volkovs et al. explored a related topic of maintaining data cleaning rules that change over time [39]. Recently, in works such as SampleClean [42], the application (i.e., queries) are used to inform data cleaning methodology. When the workload is made up of aggregate queries, cleaning samples of data may suffice. Similarly, Bergman et al. explore the problem of query-oriented data cleaning [8]. Given a query they clean data relevant to that query. Bergman et al. does not explore the Machine Learning applications studied in this work. Deshpande et al. studied data acquisition in sensor networks [14]. They explored value of information based prioritization of data acquisition for estimating aggregate queries of sensor readings. Similarly, Jeffery et al. [23] explored similar prioritization based on value of information. We see this work as pushing prioritization further down the pipeline to the end analytics. Finally, incremental optimization methods like SGD have a connection to incremental materialized view maintenance as the argument for incremental maintenance over recomputation is similar (i.e., relatively sparse updates). Krishnan et al. explored how samples of materialized views can be maintained similar to how models are updated with a sample of clean data in this work [27]

**Stochastic Optimization and Active Learning:** Zhao and Tong recently proposed using importance sampling in conjunction with stochastic gradient descent [44]. The ideas applied in ActiveClean are well rooted in the Machine Learning and Optimization literature, and we apply these ideas to the data cleaning problem. This line of work builds on prior results in linear algebra that show that some matrix columns are more informative than others [15], and Active Learning which shows that some labels are more informative that others [35]. Active Learning largely studies the problem of label acquisition [35], and recently the links between Active Learning and Stochastic optimization have been studied [20]. We use the work in Guillory et al. to evaluate a state-of-the-art Active Learning technique against ActiveClean.

**Transfer Learning and Bias Mitigation:** ActiveClean has a strong link to a field called Transfer Learning and Domain Adaptation [32]. 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. In robotics, Mahler et al. explored a calibration problem in which data was systematically corrupted [28] and proposed a rule-based technique for cleaning data. Other problems in bias mitigation (e.g., Krishnan et al. [26]) 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:** Another relevant line of work is the work in private machine learning [16,40]. Learning is performed on a noisy variant of the data which mitigates privacy concerns, embracing the error rather than correcting for it. ActiveClean is also related to work in adversarial learning [30], 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 [43], 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").

# 10. DISCUSSION AND FUTURE WORK

The experimental results suggest the following conclusions about ActiveClean: (1) when the data corruption rate is relatively small (e.g., 5%), ActiveClean cleans fewer records than Active Learning or SampleClean to achieve the same model accuracy, (2) all of the optimizations in ActiveClean (importance sampling, detection, and estimation) lead to significantly more accurate models at small sample sizes, (3) only when corruption rates are very severe (e.g. 50%) , SampleClean outperforms ActiveClean, and (4) two real-world scenarios demonstrate similar accuracy improvements where ActiveClean returns significantly more accurate models than SampleClean or Active Learning for the same number of records cleaned.

There are also a few additional points for discussion. ActiveClean provides guarantees for training error on models trained with progressive data cleaning, however, there are no such guarantees on test error. This work focuses on the problem where an analyst has a large amount of dirty data and would like explore data cleaning and predictive models on this dataset. By providing the analyst more accurate model estimates, the value of different data cleaning techniques can be judged without having to clean the entire dataset. However, the exploratory analysis problem is distinct from the model deployment problem (i.e., serving predictions to users from the model), which we hope to explore in more detail in future work. It implicitly assumes that when the model is deployed, it will be applied in a setting where the test data is also clean. Training on clean data, and testing on dirty data, defeats the purpose of data cleaning and can lead to unreliable predictions.

As the experiments clearly show, ActiveClean is not strictly *better* than Active Learning or SampleClean. ActiveClean is optimized for a specific design point of sparse errors and small sample sizes, and the empirical results suggest it returns more accurate models in this setting. As sample sizes and error rates increase, the benefits of ActiveClean are reduced. Another consideration for future work is automatically selecting alternative techniques when ActiveClean is expected to perform poorly.

Beyond these limitations, there are several exciting new avenues for future work. The data cleaning models explored in this work can be extended to handle non-uniform costs, where different errors have a different cleaning cost. Next, the empirical success of Deep Learning has led to increasing industry and research adoption of non-convex losses in many tasks that were traditionally served by convex models. In future work, we hope to explore how we can integrate with such frameworks.

# 11. CONCLUSION

The growing popularity of predictive models in data analytics adds additional challenges in managing dirty data. Progressive data cleaning in this setting is susceptible to errors due to mixing dirty and clean data, sensitivity to sample size, and the sparsity of errors. The key insight of ActiveClean is that an important class of predictive models, called convex loss models (e.g., linear regression and SVMs), can be simultaneously trained and cleaned. Consequently, there are provable guarantees on the convergence and error bounds of ActiveClean. 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 similar accuracy for significantly less records cleaned.

## 12. REFERENCES


[1] Berkeley data analytics stack. `https://amplab.cs.berkeley.edu/software/`.

[2] Dollars for docs. `http://projects.propublica.org/open-payments/`.

[3] For big-data scientists, 'janitor work' is key hurdle to insights. http://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html.

[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] Sampleclean. http://sampleclean.org/, 2015.

[6] A. Alexandrov, R. Bergmann, S. Ewen, J. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke. The stratosphere platform for big data analytics. *VLDB J.*, 23(6):939–964, 2014.

[7] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progressive approach to relational entity resolution. *Proceedings of the VLDB Endowment*, 7(11), 2014.

[8] M. Bergman, T. Milo, S. Novgorodov, and W. C. Tan. Query-oriented data cleaning with oracles. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1199–1214, 2015.

[9] D. P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010:1–38.

[10] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[11] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1247–1261, 2015.

[12] A. Crotty, A. Galakatos, and T. Kraska. Tupleware: Distributed machine learning on small clusters. *IEEE Data Eng. Bull.*, 37(3):63–76, 2014.

[13] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13(1):165–202, 2012.

[14] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 588–599, 2004.

[15] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *The Journal of Machine Learning Research*, 13(1):3475–3506, 2012.

[16] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 429–438. IEEE, 2013.

[17] J. Feng, H. Xu, S. Mannor, and S. Yan. Robust logistic regression and classification. In *Advances in Neural Information Processing Systems*, pages 253–261, 2014.

[18] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[19] 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.

[20] A. Guillory, E. Chastain, and J. Bilmes. Active learning as non-convex optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 201–208, 2009.

[21] J. M. Hellerstein, C. Ré, F. Schoppmann, D. Z. Wang, E. Fratkin, A. Gorajek, K. S. Ng, C. Welton, X. Feng, K. Li, and A. Kumar. The madlib analytics library or MAD skills, the SQL. *PVLDB*, 5(12):1700–1711, 2012.

[22] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

[23] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative support for sensor data cleaning. In *Pervasive*, pages 83–100, 2006.

[24] Z. Khayyat, I. F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, P. Papotti, J.-A. Quiané-Ruiz, N. Tang, and S. Yin. Bigdansing: A system for big data cleansing. 2015.

[25] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1):484–493, 2010.

[26] S. Krishnan, J. Patel, M. J. Franklin, and K. Goldberg. A methodology for learning, analyzing, and mitigating social influence bias in recommender systems. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 137–144, 2014.

[27] S. Krishnan, J. Wang, M. J. Franklin, K. Goldberg, and T. Kraska. Stale view cleaning: Getting fresh answers from stale materialized views. *Proceedings of the VLDB Endowment*, 8(12), 2015.

[28] J. Mahler, S. Krishnan, M. Laskey, S. Sen, A. Murali, B. Kehoe, S. Patil, J. Wang, M. Franklin, P. Abbeel, and K. Y. Goldberg. Learning accurate kinematic control of cable-driven surgical robots using data cleaning and gaussian process regression. In *2014 IEEE International Conference on Automation Science and Engineering, CASE 2014, New Taipei, Taiwan, August 18-22, 2014*, pages 532–539, 2014.

[29] C. Mayfield, J. Neville, and S. Prabhakar. ERACER: a database approach for statistical inference and data cleaning. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 75–86, 2010.

[30] B. Nelson, B. I. Rubinstein, L. Huang, A. D. Joseph, S. J. Lee, S. Rao, and J. Tygar. Query strategies for evading convex-inducing classifiers. *The Journal of Machine Learning Research*, 13(1):1293–1332, 2012.

[31] A. B. Owen. *Monte Carlo theory, methods and examples*. 2013.

[32] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[33] H. Park and J. Widom. Crowdfill: collecting structured data from the crowd. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 577–588, 2014.

[34] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[35] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[36] E. H. Simpson. The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 238–241, 1951.

[37] N. Swartz. Gartner warns firms of 'dirty data'. *Information Management Journal*, 41(3), 2007.

[38] J. R. Taylor. An introduction to error analysis: The study of uncertainties in physical measurements, 327 pp. *Univ. Sci. Books, Mill Valley, Calif*, 1982.

[39] M. Volkovs, F. Chiang, J. Szlichta, and R. J. Miller. Continuous data cleaning. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 244–255. IEEE, 2014.

[40] M. J. Wainwright, M. I. Jordan, and J. C. Duchi. Privacy aware learning. In *Advances in Neural Information Processing Systems*, pages 1430–1438, 2012.

[41] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[42] 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 Conference*, pages 469–480, 2014.

[43] H. Xiao, T. DE, B. Biggio, D. UNICA, G. Brown, G. Fumera, C. Eckert, I. TUM, and F. Roli. Is feature selection secure against training data poisoning? 2015.

[44] P. Zhao and T. Zhang. Stochastic optimization with importance sampling. *arXiv preprint arXiv:1401.2753*, 2014.

## APPENDIX

# A. EXTENSIONS

## A.1 Set-of-Records Cleaning Model

In paper, we formalized the analyst-specified data cleaning as follows. We take the sample of the records $S_{dirty}$, and apply data cleaning $C(\cdot)$. $C$ is applied to a record and produces the clean record:

$$S_{clean} = \{C(r) : \forall r \in S_{dirty}\}$$

The record-by-record cleaning model is a formalization of the costs of data cleaning where each record has the same cost to clean and this cost does not change throughout the entire cleaning session. There are, however, some cases when cleaning the first record of a certain type of corruption is expensive but all subsequent records are cheaper.

EXAMPLE 6. *In most spell checking systems, when a misspelling is identified, the system gives an option to fix all instances of that misspelling.*

EXAMPLE 7. *In entity resolution problems, when an inconsistent entity is identified all other records with the same inconsistency can be efficiently fixed.*

This model of data cleaning can fit into our framework and we formalize it as the "Set-of-Records" model as opposed to the "Record-by-Record" model. In this model, the cleaning function $C(\cdot)$ is not restricted to updating only the records in the sample. $C(\cdot)$ takes the entire dirty sample as an argument (that is the cleaning is a function of the sample), the dirty data, and updates the entire dirty data:

$$R'_{dirty} = C(S_{dirty}, R_{dirty})$$

we require that for every record $s \in S_{dirty}$, that record is completely cleaned after applying $C(\cdot)$, giving us $S_{clean}$. Records outside of $S_{dirty}$ may be cleaned on a subset of dirty attributes by $C(\cdot)$. After each iteration, we re-run the detector, and move any $r \in R'_{dirty}$ that are clean to $R_{clean}$. Such a model allows us to capture data cleaning operations such as in Example 6 and Example 7.

## A.2 Non-convex losses

We acknowledge that there is an increasing popularity of non-convex losses in the Neural Network and Deep Learning literature. However, even for these losses, gradient descent techniques still apply. Instead of converging to a global optimum they converge to a locally optimal value. Likewise, ActiveClean will converge to the closest locally optimal value to the dirty model. Because of this, it is harder to reason about the results. Different initializations will lead to different local optima, and thus, introduces a complex dependence on the initialization with the dirty model. This problem is not fundemental to ActiveClean and any gradient technique suffers this challenge for general non-convex losses, and we hope to explore this more in the future.

# B. PROOF OF LEMMA 1

LEMMA 5. *The gradient estimate $g(\theta)$ is unbiased if $g_S$ is an unbiased estimate of:*

$$\frac{1}{\mid R_{dirty} \mid} \sum g_i(\theta)$$

PROOF SKETCH.

$$\mathbb{E}(\frac{1}{\mid R_{dirty} \mid} \sum g_i(\theta)) = \frac{1}{\mid R_{dirty} \mid} \cdot \mathbb{E}(\sum g_i(\theta)))$$

By symmetry,

$$\mathbb{E}(\frac{1}{\mid R_{dirty} \mid} \sum g_i(\theta)) = g(\theta)$$

$$\mathbb{E}(\frac{1}{\mid R_{dirty} \mid} \sum g_i(\theta)) = \frac{\mid R_{dirty} \mid \cdot g_S + \mid R_{clean} \mid \cdot g_C}{\mid R \mid}$$

□

# C. CONVEX VS. STRONGLY CONVEX

The error bound discussed in Proposition 2 can be tightened for a class of models called strongly convex (see [9] for a defintion).

PROPOSITION 3. *For a strongly convex loss, a batch size $b$, and $T$ iterations, the convergence rate is bounded by $O(\frac{\sigma^2}{bT})$.*

# D. PROOF OF LEMMA 2

The variance of this estimate is given by:

$$Var(\mu) = \mathbb{E}(\mu^2) - \mathbb{E}(\mu)^2$$

Since the estimate is unbiased, we can replace $\mathbb{E}(\mu)$ with the average of $A$:

$$Var(\mu) = \mathbb{E}(\mu^2) - \bar{A}^2$$

Since $\bar{A}$ is deterministic, we can remove that term during minimization. Furthermore, we can write $\mathbb{E}(\mu^2)$ as:

$$\mathbb{E}(\mu^2) = \frac{1}{n^2} \sum_i^n \frac{a_i^2}{p_i}$$

Then, we can solve the following optimization problem (removing the proportionality of $\frac{1}{n^2}$) over the set of weights $P = \{p(a_i)\}$:

$$\min_P \sum_i^N \frac{a_i^2}{p_i}$$

$$\text{subject to: } P > 0, \sum P = 1$$

Applying Lagrange multipliers, an equivalent unconstrained optimization problem is:

$$\min_{P>0,\lambda>0} \sum_i^N \frac{a_i^2}{p_i} + \lambda \cdot (\sum P - 1)$$

If, we take the derivatives with respect to $p_i$ and set them equal to zero:

$$-\frac{a_i^2}{2 \cdot p_i^2} + \lambda = 0$$

If, we take the derivative with respect to $\lambda$ and set it equal to zero:

$$\sum P - 1$$

Solving the system of equations, we get:

$$p_i = \frac{\mid a_i \mid}{\sum_i \mid a_i \mid}$$

# E. TAYLOR APPROXIMATION

We can take the expected value of the Taylor series expansion around the dirty value. If $d$ is the dirty value and $c$ is the clean value, the Taylor series approximation for a function $f$ is given as follows:

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

If we ignore the higher order terms, we see that the linear term $f'(d) \cdot (d - c)$ decouples the features. We only have to know the change in each feature to estimate the change in value. In our case the function $f$ is the gradient $\nabla\phi$. So, the resulting linearization is:

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

$$+ \frac{\partial}{\partial Y}\phi(x, y, \theta) \cdot (y - y^{(c)})$$

When we take the expected value:

$$\mathbb{E}(\nabla\phi(x_{clean}, y_{clean}, \theta)) \approx \nabla\phi(x, y, \theta) + \frac{\partial}{\partial X}\nabla\phi(x, y, \theta) \cdot \mathbb{E}(\Delta x)$$

$$+ \frac{\partial}{\partial Y}\nabla\phi(x, y, \theta) \cdot \mathbb{E}(\Delta y)$$

So the resulting estimation formula takes the following form:

$$\approx \nabla\phi(x, y, \theta) + M_x \cdot \mathbb{E}(\Delta x) + M_y \cdot \mathbb{E}(\Delta y)$$

Recall that we have a $d$ dimensional feature space and $l$ dimensional label space. Then, $M_x = \frac{\partial}{\partial X}\nabla\phi$ is an $d \times d$ matrix, and $M_y = \frac{\partial}{\partial Y}\nabla\phi$ is a $d \times l$ matrix. Both of these matrices are computed with respect to dirty data, and we will present an example. $\Delta x$ is a $d$ dimensional vector where each component represents a change in that feature and $\Delta y$ is an $l$ dimensional vector that represents the change in each of the labels.

# F. EXAMPLE $M_X$, $M_Y$

**Linear Regression:**

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

For a record, $r$, suppose we have a feature vector $x$. If we take the partial derivatives with respect to x, $M_x$ is:

$$M_x[i, i] = 2x[i] + \sum_{i \neq j} \theta[j]x[j] - y$$

$$M_x[i, j] = \theta[j]x[i]$$

Similarly $M_y$ is:

$$M_y[i, 1] = x[i]$$

**Logistic Regression:**

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

where

$$h(z) = \frac{1}{1 + e^{-z}}$$

we can rewrite this as:

$$h_\theta(x) = \frac{1}{1 + e^{\theta^T x}}$$

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

In component form,

$$g = \nabla\phi(x, y, \theta)$$

$$g[i] = h_\theta(x) \cdot x[i] - yx[i]$$

Therefore,

$$M_x[i, i] = h_\theta(x) \cdot (1 - h_\theta(x)) \cdot \theta[i]x[i] + h_\theta(x) - y$$

$$M_x[i, j] = h_\theta(x) \cdot (1 - h_\theta(x)) \cdot \theta[j]x[i] + h_\theta(x)$$

$$M_y[i, 1] = x[i]$$

**SVM:**

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

Therefore,

$$M_x[i, i] = \begin{cases} -y[i] \text{ if } y \cdot \boldsymbol{x} \cdot \theta \leq 1 \\ 0 \text{ if } y\, \boldsymbol{x} \cdot \theta \geq 1 \end{cases}$$

$$M_x[i, j] = 0$$

$$M_y[i, 1] = x[i]$$

# G. AGGREGATE QUERIES AS CONVEX LOSSES

## G.1 AVG and SUM queries

`avg`, `sum` queries are a special case of the convex loss minimization discussed in the paper: If we define the following loss, it is easy to verify the the optimal $\theta$ is the mean $\mu$:

$$\phi = (x_i - \theta)^2$$

with the appropriate scaling it can support `avg`, `sum` queries with and without predicates. Taking the gradient of that loss:

$$\nabla\phi = 2(x_i - \theta)$$

It is also easy to verify that the bound on errors is $O(\frac{\mathbb{E}((x-\mu)^2}{bT})$, which is essentially the CLT. The importance sampling results are inutitive as well. Applying the linearization:

$$M_x = 2$$

The importance sampling prioritizes points that it expects to be far away from the mean.

## G.2 MEDIAN

Similarly, we can analyze the `median` query. If we define the following loss, it is easy to verify the the optimal $\theta$ is the median $m$:

$$\phi = \mid x_i - \theta \mid$$

Taking the gradient of that loss:

$$\nabla\phi = 1 \text{ if } < \text{m, -1 if} > \text{m}$$

Applying the linearization:

$$M_x = 0$$

The intuitive result is that a robust query like a median does not need to consider estimation as the query result is robust to small changes.

# H. EXPERIMENTAL COMPARISON

### H.1 Robust Logistic Regression

We use the algorithm from Feng et al. for robust logistic regression.

1. Input: Contaminated training samples $\{(x_1, y_1), ..., (x_n, y_n)\}$ an upper bound on the number of outliers n, number of inliers n and sample dimension p.
2. Initialization: Set
$$T = 4\sqrt{\log p/n + \log n/n}$$
3. Remove samples $(xi, yi)$ whose magnitude satisfies $\|x_i\| \geq T$.
4. Solve regularized logistic regression problem.

### H.2 Active Learning

There is a well established link between Active Learning and Online Gradient-based Algorithms [20]. To fairly evaluate an Active Learning methodology in these experiments, we run a gradient descent where examples are priortized by expected gradient length (explained in [35]). Ignoring the detection step, there are two differences between this algorithm and the one we propose. First, we calculate the gradient with respect to the an estimate of the clean data, and second we sample rather than using a deterministic ordering. While such Active Learning algorithms have been studied in the Learning Theory community, they have not been adopted in data cleaning or crowdsourcing research. Typical algorithms include uncertainty sampling, where a classifier prioritized data closest to the margin. However, algorithms such as uncertainty sampling focus on the narrow problem of label acquisition in hyperplane classifiers; a problem too narrow for application in this setting.

## I. DOLLARS FOR DOCS SETUP

The dollars for docs dataset has the following schema:

```
Contribution(pi_speciality, drug_name, device_name,
corporation, amount, dispute, status)
```

To flag suspect donations, we used the `status` attribute. When the `status` was "covered" that means it was an allowed contribution under the researcher's declared protocol. When the `status` was "non-covered" that means it was a disallowed contribution under the researcher's declared protocol. The rest of the textual attributes were featurized with a bag-of-words model, and the numerical amount and dispute attributes were treated as numbers.

We cleaned the entire Dollars for Docs dataset upfront to be able to evaluate how different budgeted data cleaning strategies compare to cleaning the full data. To clean the dataset, we loaded the entire data 240089 records into Microsoft Excel. We identified four broad classes of errors:

**Corporations are inconsistently represented:** "Pfizer", "Pfizer Inc.", "Pfizer Incorporated".

**Drugs are inconsistently represented:** "TAXOTERE DOCETAXEL -PROSTATE CANCER" and "TAXOTERE"

**Label of covered and not covered are not consistent:** "No", "Yes","N", "This study is not supported", "None", "Combination"

**Research subject must be a drug OR a medical device and not both:** "BIO FLU QPAN H7N9AS03 Vaccine" and "BIO FLU QPAN H7N9AS03 Device"

To fix these errors, we sorted by each column and merged entities that looked similar and removed inconsistencies as in the status labels. When there were ambiguities, we refered to the drug company's website and whitepapers. When possible, we used batch data transformations, like find and replace (i.e. the Set-of-Records model). In all, 44234 records had some error and full data cleaning required about 2 days of efforts.

Once cleaned, in our experiment, we encoded the 4 problems as data quality constraints. To fix the constraints, we looked up the clean value in the dataset that we cleaned up front.

**Rule 1:** Matching dependency on corporation (Weighted Jaccard Similarity > 0.8).

**Rule 2:** Matching dependency on drug (Weighted Jaccard Similarity > 0.8).

**Rule 3:** Label must either be "covered" or "not covered".

**Rule 4:** Either drug or medical device should be null.

## J. MNIST SETUP

We include visualization of the errors that we generated for the MNIST experiment. We generated these errors in MATLAB by taking the grayscale version of the image (a $64 \times 64$ matrix) and corrupting them by block removal and fuzzying.

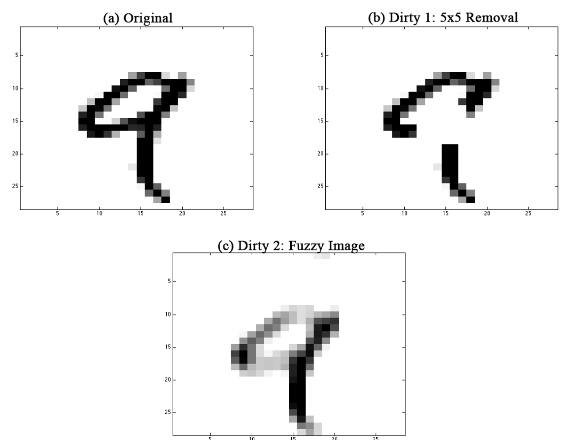


**Figure 13: We experiment with two forms of corruption in the MNIST image datasets: 5x5 block removal and making the images fuzzy. Image (a) shows an uncorrupted "9", image (b) shows one corrupted with block removal, and image (c) shows one that is corrupted with fuzziness.**