ActiveClean: Progressive Data Cleaning For Advanced Analytics

ABSTRACT

Data management systems are increasingly supporting advanced analytics such as Machine Learning. A perennial challenge is the presence of dirty data – in the form of missing, duplicate, incorrect or inconsistent values – which can bias the learning algorithms, leading to incorrect or error-prone predictions. Although, this can be ameliorated by cleaning the input dataset using existing data cleaning techniques, the cost of doing so (e.g., manually finding and cleaning all data) is practically and economically infeasible. In addition, existing learning algorithms are not suited for training on partially cleaned data, which can lead to dramatically incorrect models. In response, we have developed the ActiveClean system, which integrates incremental data cleaning with machine learning algorithms. ActiveClean iteratively updates a machine learning model as data records are individually (or in batches) cleaned. We additionally study numerous database and ML-oriented optimizations to speed the end-to-end process. Evaluation on four real-world datasets suggests that our methodology can learn more accurate models by cleaning significantly fewer records than alternatives such as uniform sampling and active learning, thus bridging the gap between the reality of dirty data sources and the quality of predictive models at the end of the pipeline.

1. INTRODUCTION

The proliferation of learning-based applications such as recommender systems, fraud detection, and automated content tagging, has encouraged the development of advanced statistical analytics libraries integrated with the data management stack [1,5,11,21]. While these libraries significantly reduce the required effort in developing such applications, acquiring “clean" data of sufficient quality for learning is still a bottleneck. Data are susceptible to various forms of corruption such as missing, incorrect, or inconsistent representations, and industrial surveys have established that dirty data are prevalent [38]. Data corruption, when systematic [39] (i.e., correlated with the data), can bias learning algorithms and lead to error-prone predictions [45].

Supervised machine learning is an important class of these new advanced analytics, and it is particularly sensitive to systematic corruption which can lead to confounding correlations between features and labels. 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 the “18-24" age group and music liked by European users. While there is an extensive literature on robust Machine Learning, this work largely focuses on resilience to atypical outliers (i.e., age “150") and not spurious correlations.

Such problems can be ameliorated by a variety of data cleaning techniques (see Rahm and Do [35] for a survey). Recently, several scalable data cleaning frameworks have been proposed including BigDansing [24], SampleClean [4], and Katara [10]. Unfortunately, even with recently proposed optimizations, data cleaning can be time consuming and economically infeasible [44]. Programming data transformations to fix all problems manifest in the data requires substantial developer effort [23]. When scripted transformations do not suffice, crowdsourcing is a popular alternative with recent success in missing value filling and entity resolution [4,10,19,33]. However, crowdsourcing comes at the cost of additional latency, overhead of managing human workers, and significant monetary expense.

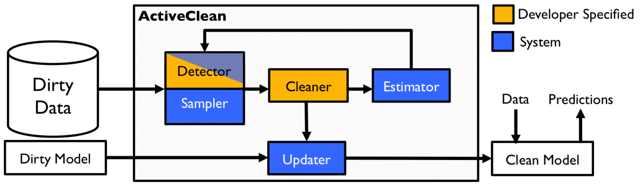


Figure 1: ActiveClean presents an architecture where data cleaning is integrated with model training. The user specifies a data cleaning procedure, and we provide a framework for sampling, model update, and feedback through estimation.

Thus, for many corrupted datasets, the analyst can only practically clean $k \ll N$ records. However, budgeted data cleaning poses a few problematic methodological issues when applied before model training. First, mixing dirty and clean data violates homogeneity assumptions and can lead to dramatically incorrect models in even simple scenarios (Figure 2). To avoid this problem, we could use a random sampling approach (e.g., [44]), where the analyst first samples k records, applies data cleaning to this subset, and then trains a model using only the clean data. While this avoids the earlier problem, model training can require a large amount of training data and k examples may not be enough to support viable predictions. Furthermore, if errors are sparse, any random sampling-based approach will largely select clean data leading to wasted effort. Under strict cleaning budgets, the errors introduced by sampling or mixing may dominate any gains from data cleaning, leading to unreliable or even misleading conclusions about data or model quality.

In response to these problems, we propose ActiveClean, a framework that integrates existing data cleaning techniques with model training in a way that ensures reliable progressive data cleaning. Suppose an analyst has a large dirty dataset, a data cleaning technique, a budget of cleaning k records, and a model she wishes to train. ActiveClean divides the budget into small batches, applies the data cleaning to each batch, and incrementally updates the model after cleaning. The system is tuned to select the most valuable samples of data to clean based on previous batches of cleaned data. ActiveClean can integrate analyst-specified dirty data detection rules (such as constraints) into the framework to further guide sampling. The key insight is that the iterative data cleaning and update process can be analyzed as a form of Stochastic Optimization [8] allowing us to leverage convex-analysis theory to develop optimized sampling methods while preserving provable guarantees.

As analytics frameworks increasingly support advanced analytics, tools that encourage reliable methodologies are required. The incremental updates from ActiveClean give provable bounds on intermediate results, and these results are also viable models that can be used for prediction. Specifically, our contributions are:

- We propose the ActiveClean framework that tightly integrates data cleaning and model training.
- (Model Update Section 5) We propose a technique that updates a dirty model from a sample of clean data based on gradient descent. We batch together data that we expect to be clean leading to significant improvements in convergence rate.
- (Sampling Section 6) We derive an optimal sampling distribution, which can be approximated by maintaining an estimate of the impact of data cleaning on a record.
- (Estimation Section 7) The estimation technique applies linearization to decouple changes in different features allowing us to integrate knowledge about what is wrong with the data (i.e. analyst-specified detection rules) to better estimate error impact.
- We evaluate ActiveClean on real and synthetic datasets to show that our approach gives a sharper cost-accuracy tradeoff than random sampling and Active Learning.

2. PROBLEM SETUP

2.1 Motivating Example

To motivate this work, we present one of our experimental datasets as a running example (Dollars for Docs dataset see Experiments, Section 8.6.1):

Dollars for Docs [2]. ProPublica has collected a dataset of corporate contributions to doctors for analysis of whether these contributions (negatively) affect research. They reported that some doctors received over $500,000 in travel, meals, and consultation expenses [3]. ProPublica meticulously curated a dataset from the Centers for Medicare and Medicaid Services listing nearly 250000 research contributions and aggregated these contributions by physician, drug, and pharmaceutical. We explore whether this problem could have been made easier with Machine Learning where we try to learn what features of a contribution predict an improprietous donation. This problem is typical of fraud detection scenarios based on observational data seen in finance, insurance, and medicine.

The dataset has the following schema:

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

`pi_speciality` is a textual attribute describing the specialty of the doctor receiving the contribution.
`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).
`amount` is a numerical attribute representing the contribution amount.
`dispute` is a Boolean attribute describing whether the research was disputed.
`status` is a string label describing whether the contribution was allowed under the declared research protocol. There are two possible statuses “Covered" and “Non-Covered".

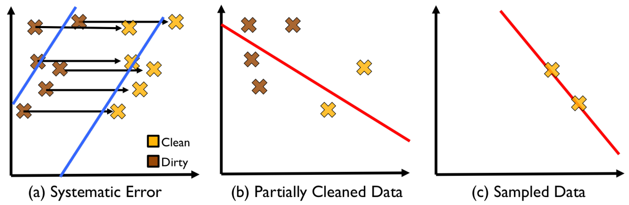
Using this dataset, consider the following analysis scenario.

EXAMPLE 1. *We are interested in predicting impropriety in medical research contributions, by exploring what features of a research contribution predict the* `status` *attribute. We featurize the textual attributes with a bag-of-words model and treat the* `amount` *and* `dispute` *attributes as numerical features. The model is a Support Vector Machine (SVM) that predicts the label* $\{1, 0\}$ *where* 1 *indicates a disallowed contribution.*

However, this dataset is very dirty, and the systematic nature of the corruption can result in a misleading model. On the ProPublica website [2], they list numerous types of corruption that had to be cleaned before publishing the data (see Appendix 12.7). For example, the most significant contributions 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 fraud detection scenario such as this one, the effect of systematic error can be serious. Duplicate entity representations, could reduce the correlation between these entities and impropriety. We found that nearly 40,000 of the 250,000 records had some form of corruption either with entity resolution issues or other inconsistencies.

Cleaning With a Budget: Let us suppose our analyst wants evaluate whether the problems affect her model, and she wants to do this without having to manually validate every record. The natural first solution to is to clean some of the data, and then retrain the model on the partially cleaned data. However, this solution can give unreliable results since we are mixing dirty and clean data. In Figure 2, we illustrate the dangers of this approach on a very simple dirty dataset and model. Suppose, we are trying to find a best fit line for two variables. One of the variables is systematically corrupted with a translation in the x-axis (Figure 2a). We mark the dirty data in brown and the clean data in orange, and show their respective best fit lines in blue. If we clean two of the data points (Figure 2b), and then find the best fit line, the partially cleaned model is dramatically incorrect. This is a well-known phenomenon called Simpsons paradox, where mixtures of different populations of data can result in spurious relationships [37].

An alternative solution is to avoid the dirty data altogether, instead of mixing the two populations. She can take a random sample of data, apply data cleaning, and train a model only on that sample of data. This is similar to SampleClean [44]; 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. The resulting models are not useful (i.e., cannot accurately predict future test data) when the sample size is too small (Figure 2c).



(a) Systematic Error (b) Partially Cleaned Data (c) Sampled Data

Figure 2: (a) Systematic corruption (translation) in one variable where the dirty data is in brown, the clean data is in yellow, and their respective best fit lines are in blue. (b) Applying the same model to partially clean data results in a dramatically incorrect answer. (c) Likewise, small sample sizes can result in similarly incorrect models.

ActiveClean avoids both pitfalls, Simpson's paradox and sample size dependence. In Section 5, we show how we do this with iterative gradient steps (i.e., incrementally moving the line based on the clean data). This takes advantage of the dirty data as well as the clean data, but still have provable properties about the intermediate results. The intuition is that it smoothly and iteratively transitions the model from one population (the dirty data) to another (the clean

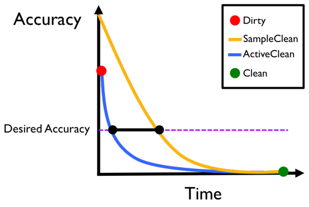


Figure 3: ActiveClean is designed to converge to an accurate model with fewer cleaned records than a uniform sampling approach (SampleClean).

data). In Figure 3, we illustrate our ideal tradeoff space of sampling and data cleaning. At two extremes we have no cleaning (just using the dirty data) and full cleaning.

2.2 Preliminaries

We focus on a class of Supervised Learning problems called convex-loss minimization problems (see Friedman, Hastie, and Tibshirani [18] for an introduction). This class of problems includes all generalized linear models (including linear and logistic regression), and all variants of support vector machines.

Formally, the goal is to learn a vector of model *parameters* θ from training examples. We start with a set of training examples $\{(x_i, y_i)\}_{i=1}^N$ on which we minimize a loss function ϕ (a penalty for getting the prediction wrong) at each point parametrized by θ.

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

For example, in a linear regression ϕ 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 θ 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 $r(\theta)$.

2.2.1 Types of Errors

We use the following metrics to quantify inaccuracy in a model:

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

Testing Error. Let θ be the model trained on the dirty data, and let θ^* be the model trained on the same data if it 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 $T(\theta^*) - T(\theta)$

The *error* in the model and in testing is caused by two underlying problems:

Data Error. Error caused by dirtiness.

Sampling Error. Error introduced by sampling i.e., the difference between a model trained on $p\%$ of the data and 100% of the data.

In this paper, when we use the term *error*, we are referring to **model error**. We will be explicit with other terms when describing data errors (e.g. “data corruption").

2.3 ActiveClean Problem

The basic question that we explore in this work is how to update a model trained on dirty data with data cleaning. Formally,

PROBLEM 1 (ACTIVECLEAN PROBLEM). *Let R be a dirty relation and each row $r \in R$ is turned into a feature vector and label tuple $F(r) \mapsto (x, y)$. We are given a convex regularized loss model parametrized by θ trained on the set of features and labels $\{(x, y)\}$. The user specifies a data repair step which cleans a record $C(r) \mapsto r_{clean}$. Optionally, the user can provide a set of dirty data detection conditions that select a set of candidate dirty data R_{dirty}. With a budget of applying cleaning i.e., $C(\cdot)$, only k times, return an estimate $\hat{\theta}$ of the clean model.*

Our goal is to use information from the model to inform data cleaning on samples, and use the information from clean samples to update the model. The tight feedback loop between model training and data cleaning poses several new algorithmic challenges. If we are not careful about how we update the model, we can end up with the Simpsons paradox problem described before. Data cleaning and model training happen at very different time scales, we have to carefully budget our effort to ensure that optimizations address rate-determining steps in the workflow.

This framework is optimized around problems with expensive data cleaning. As the complexity of dirtiness increases, so does the amount of computation and human-effort needed to resolve them. The consequence is that automated repair is either not scalable or relies on greedy repairs that can be unreliable. Increasingly, human effort is seen as valuable in data cleaning [19,33,43,44]. When humans are involved, per record latencies for data repair 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 [43] are over 100,000x slower per record. Here is an example application of ActiveClean with our running example:

EXAMPLE 2. *The analyst first trains her SVM model on the dirty data ignoring the effects of the errors returning 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. SYSTEM ARCHITECTURE

We describe the ActiveClean architecture and the basic algorithmic framework. We will address the individual components in the subsequent sections.

3.1 Overview

We revisit the architecture diagram (Figure 1) in more detail. The first step of the system is *initialization*. In this step, the user provides a relation R, a data cleaning technique $C(\cdot)$, and a dirty model $\theta^{(d)}$ trained on the dirty dataset. Optionally, the user can provide a dirty data detector $D(\cdot)$ which selects the set of likely corrupted records from R. If one is not provided, we start by treating all of the data is dirty and try to learn a detector as we clean more data. We are given a *featurization* function $F(\cdot)$ that maps

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

each record in R to a feature vector and a label. 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 .

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 select only dirty data. Once a sample is selected, we apply the user-specified data cleaning operation to clean the sample (the *cleaner*). Then, after the sample is cleaned, the dirty model is updated by the *updater*.

The next two steps in the architecture are feedback steps where we update the sampling for the next iteration. The *estimator* estimates the change on the model if a dirty record is cleaned based on prior cleaned data. This information can be used to guide sampling towards more valuable records. After estimation, we also update our detector $D(\cdot)$ based on the data that we have cleaned. After all of the iterations are complete, the system returns the updated model.

To summarize the architecture in pseudocode:

```
1. Init(dirty_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 = Updater(current_model,
        sample_prob, clean_sample)
    (d) sample_prob = Estimator(dirty_data,
        cleaned_data, detector)
    (e) detector = DetectorUpdater(detector)
3. Output: current_model
```

3.2 Challenges and Formalization

We highlight the important components and formalize the research questions that we explore in this paper.

Featurization. There is a featurization function F which maps every record in $\mathcal{R}$ to a d dimensional feature vector and a l dimensional label tuple:

$$F(r \in R) \rightarrow (\mathbb{R}^l, \mathbb{R}^d)$$

The result of the featurization are the data matrices X and Y.

$$F(R) \rightarrow (X, Y)$$

Detector (Section 4). The first challenge in ActiveClean is dirty data detection. In this step, we select a candidate set of dirty records $R_{dirty} \subseteq R$. We will discuss two techniques to do this: (1) an a priori case, and (2) and an adaptive case. In the a priori case, we know which data is dirty in advance. In the adaptive case, we train a classifier based on data that we have already cleaned to select the dirty data.

Sampler (Section 6). We take a sample of the 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 (Analyst-Specified). We take the sample of the records S_{dirty}, and apply the analyst-specified data cleaning $C(\cdot)$. In this paper, we focus 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 us to measure the performance of ActiveClean in terms of model error as a function of sample size. The record-by-record cleaning model is not a fundemental restriction of our approach, and in our extensions (Section 12.1.1) , we discuss an also 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) with same error throughout the dataset efficiently.

Updater (Section 5). We update the model $\theta^{(t)}$ based on the newly cleaned data $F(S_{clean})$ resulting in $\theta^{(t+1)}$. Analyzing the model update procedure as a stochastic gradient descent algorithm will help us derive the sampling distribution and estimation.

Estimator (Section 7): The estimator approximates the optimal distribution derived in the Sample step. Based on the change between $F(S_{clean})$ and $F(S_{dirty})$, we direct 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, when we sample data to clean, we want to ensure that the data that we are sampling is likely to be dirty.

4.1 Goals

The detection component needs to give us two important pieces of information about a record: (1) whether the record is dirty, and (2) if it is dirty, what is wrong with the record. From (1), we can select a subset of dirty records to sample at each batch. (2) is important so we can estimate how valuable cleaning will be for this record. There are two cases that we explore in ActiveClean: *a priori* and *adaptive*. In the a priori case, we recognize that for many data cleaning methodologies, we can efficiently select the set of dirty records without repair. In the adaptive case, we relax this assumption, and explore how we can learn which records are dirty and clean with a classifier.

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 enumerate what is wrong with these records.

DEFINITION 1 (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, we can find the corresponding features that are dirty f_r and labels that are dirty l_r.

We highlight example use cases of this definition using data cleaning methodologies proposed in the liteature.

Constraint-based Repair: One model for handling errors in database declaring a set of constraints on a database. Data are cleaned iteratively until the constraints are satisfied [16,24,46].

Detection. Let Σ be a set of constraints on the relation $\mathcal{R}$. In the detection step, we select 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 3. *An example of a constraint for our running example dataset is that the* `status` *of a contribution can be only "covered" or "non-covered". Any other value for* `status` *is an error.*

Entity Resolution: Another common data cleaning task is Entity Resolution [19,25,43]. 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. This 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 satisfy $r(a) \approx r(a)'$. The set e_r is the attributes of r that have entity resolution problems.

EXAMPLE 4. *An example of an Entity Resolution problem is seen in our earlier example about corporation names e.g. "Pfizer Inc.", "Pfizer Incorporated", "Pfizer".. We can define a similarity relationship $WeightedJaccard(r1, r2) > 0.8$ and select all records that satisfy this condition (their Weighted Jaccard Similarity is greater than 0.8).*

4.3 Adaptive Detection

Next, we explore how we can handle the case where the detection is not known in advance. Based on what we have already cleaned, we can learn a detection classifier (note that this "learning" is distinct from the "learning" at the end of the pipeline). The challenge in formulating this problem is that we not only need to know whether or not a record is dirty, but also how it is dirty (e.g. e_r in the a priori case). Instead of assuming that we know which features are corrupted, let us say that we know that there are u classes of data corruption. These classes are corruption categories that do not necessarily align with features, but every records is classified with at most one category. For example, suppose we have outliers and missing values, there are three classes of corruption: outliers, missing values, and both. As the analyst cleans data, she tags dirty data with one of the u classes. Then, the detection problem reduces to a multiclass classification problem. To address this problem, we can use any multiclass classifier, and we use an all-versus-one SVM in our experiments. Since this classifier is internal to our system, it does not have to be a convex model (i.e., it can be a Decision Tree or Random Forest).

When an example (x, y) is cleaned, the repair step also has to provide a label to which of the clean, $1, 2, ..., u$ classes it belongs. It is possible that u increases each iteration as more types of dirtiness are discovered. Thus, after cleaning k records, we have a dataset of k records labeled with $u + 1$ classes (including one for "not dirty").

DEFINITION 2 (ADAPTIVE CASE). *To select R_{dirty}, we select the set of records for which κ gives a positive error classification (i.e., one of the u error classes). After each sample of data is cleaned, the classifier κ is retrained. So the result is:*

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

We highlight an example of adaptive detection using an interative data cleaning tool such as OpenRefine [40].

Interactive Data Cleaning.

EXAMPLE 5. *OpenRefine is a spreadsheet-based tool that allows users to explore and transform data. However, it is limited to clean data that can fit in memory on a single personal computer. Since the cleaning operations are coupled with data exploration, we do not know what is dirty in advance (the analyst may discover new errors as she cleans).*

Suppose our analyst wants to use OpenRefine to clean our 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 clears the device attribute for all records that have the drug name in question. Every time she makes a batch data transformation (i.e., clearning the device attribute), we can list the set of records that have changed. Each transformation becomes and error class, and the records that have changed records become positive training examples for a classifier to guide future samples.

5. UPDATE

Before we discuss sampling, we will first discuss how to update a model. We will assume that our detection step in the previous section has selected a set of candidate records R_{dirty} and our sampling step has sampled from this set of candidate records. We will show that this model update procedure can be interpreted as a Stochastic Gradient Descent (SGD) algorithm, which gives us a theoretical framework to analyze convergence and bound the error at each step.

5.1 Update Problem

The challenge is that we want a model update technique that takes advantage of the entire dirty data and a small sample of cleaned data. However, we want to avoid the Simpson's paradox problem discussed previously. Let us first describe the abstract problem. Suppose we have a random sample of data $S_{dirty} \subseteq R_{dirty}$ from the results of the detection in the previous section. We also know the sampling probability of each record $p(r)$. We apply our data cleaning technique to the sample and get S_{clean}, and then apply our featurization to get clean features and labels $(X^{(c)}, Y^{(c)})$. In the model update problem, we update the dirty model $\theta^{(d)}$ based on the clean sample and return θ^{new}. Our goal is that these updates should minimize the error of the updated model and the true model $\theta^{(c)}$ (if we cleaned the entire data and trained over the entire data):

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

5.2 Geometric Interpretation

We will present the update algorithm intuitively by describing it in terms of the convex geometry of the problem. Let us consider this problem in one dimension (i.e., the parameter θ is a scalar value), so then the goal is to find the minimum point (θ) of a curve $l(\theta)$. The consequence of dirty data is that we optimize the wrong loss function. We are optimizing the function $l'(\theta)$, when we really want to optimize $l(\theta)$. Figure 4A, illustrates the consequence of this optimization. The brown dotted line shows the loss function on the dirty data. If we optimize this loss function, we find the θ that at the minimum point. However, the true loss function (w.r.t to the clean data) is in blue. This optimal value is a suboptimal point on clean curve.

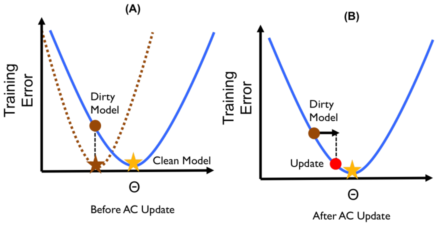


Figure 4: (A) A model trained on dirty data can be thought of as a sub-optimal point w.r.t to the clean data. (B) The gradient gives us the direction in which we need to move the suboptimal model to approach the true optimum.

We want to get to the optimal clean model $\theta^{(c)}$ which is visualized as a yellow star. The first question is which direction do we move θ (i.e., left or right). We do not know whether we need to . For this class of models, given a suboptimal point, we can find the direction to the global optimum with the gradient of the loss function. The gradient is a d-dimensional vector function of the current model θ and the clean data. We need to move some distance γ along this direction (Figure 4b):

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

At the optimal point, the magnitude of the gradient will be zero. So intuitively, this approach allows us to iteratively move the model downhill, or correct the dirty model until the budget is reached.

However, in reality, we do not have the all of the clean data and have to approximate this gradient from a sample. The intuition, which we will formalize in Section 5.5, is that if we are on average in the right direction the algorithm is guaranteed to converge with bounds on the convergence rate. It turns out that for an appropriately chosen γ, this convergence will hold for any batch size that we choose.

5.3 Average Gradient From a Sample

To derive a sample-based update rule, the first property that we should recognize is that sums commute with derivatives and gradients. Our class of models are based on loss minimization, that is they are a sum of losses, so given the current best model θ, the gradient $g^*(\theta)$ is:

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

We estimate this gradient from a sample by averaging over a sample of clean data and reweighting them by their sampling probability. 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 if we add in the iteration, for every batch of data cleaned t, we apply the update to the current best model estimate:

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

The detection in the previous step adds a small complication, since the sample S_{dirty} is not representative of all of the data. We also have to compensate 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 α, β which we will subsequently discuss how to select (Section 5.5):

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

Finally, we add 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

We initialize the algorithm with $\theta^{(1)} = \theta^{(d)}$ which is the dirty model. At each iteration $t = \{1, ..., T\}$, we clean a batch of data b selected from the set of candidate dirty records R_{dirty}. Then, starting with $\theta^{(d)}$, we apply an averaged gradient update to get $\theta^{(t)}$. We iterate until our budget of cleaning $k = T \cdot b$ record is reached.

We present the model update algorithm here:

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 meanings of the parameters b, λ, α, β.

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 analyst gets less frequent model updates.

Step Size γ : γ controls how far should we travel in the gradient direction.

Weights α, β : α, β are the proportions with which we should 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 gives us a theoretical framework to understand and analyze our 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, and we have to formalize our method as a variant of mini-batch SGD to apply the theory.

ActiveClean as Lazy SGD: One way to interpret this update method is a variant of SGD applied to the clean data 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 arbitrary initializations SGD makes significant progress in less than one epoch (a pass through the entire dataset) [9]. Furthermore in our setting, the dirty model can be much more accurate than an arbitrary initialization. This is the property that we exploit to make significant progress by cleaning only a sample of data.

Deriving α and β: We first show how to select α and β such that our estimate of the gradient is unbiased. We will then show that SGD will converge when the estimate is unbiased and the stepsize is chosen appropriately. We construct batch S_{dirty} only from R_{dirty} and apply cleaning to this batch. Since we know the sizes of R_{dirty} and its complement, we can combine the two estimates g_C and g_S together:

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

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 12.2). □

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

Convergence: Convergence properties of batch SGD formulations has been well studied [12]. From the preceding analysis, we know that our gradient estimate is unbiased and we know how to select the step size. Following from these two points, we can ensure convergence:

PROPOSITION 1. *For an appropriately chosen learning rate γ_t, batch stochastic gradient descent will converge if $\mathbb{E}(g_S) = g^*$.*

Convergence Rate: The convergence rates of SGD are also well analyzed [8,12,47]. This allows us to bound the error of intermediate models and understand the expected number of steps before 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* $O(\frac{\sigma^2}{\sqrt{bT}})$. σ^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 analyst 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 our approach are more evident at smaller batch sizes. There are diminishing returns to increasing the batch size $O(\frac{1}{\sqrt{b}})$. Empirically, we find that batch sizes of 50 converge the fastest on our data and we specify this parameter in our experiments. 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, we can always apply the updates in smaller batches. For example, the batch size set by the analyst might be $b = 1000$, but we apply the model updates after every 50 records are cleaned. This way we can dissociate the requirements of SGD and the data cleaning technique.

Non-covex losses: (See Appendix 12.1.2 for a note on such losses.)

6. SAMPLING

In the previous section, we assumed that our model update received a sample with probabilities $p(r)$. In this section, we derive an optimal sampling problem that directly follows from our analysis of our 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), but we can use this to inform the next section where we estimate the cleaned value.

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. We can sacrifice computational overhead by precomputing some results over the entire data for savings during the data cleaning phase. We formulate this problem as an optimal sampling problem where we want to compute the sampling probabilities $p(r)$ that maximize the accuracy of our updates.

6.2 Optimal Sampling Problem

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

DEFINITION 3 (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:*

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

In other words, we want to most accurately (in the mean squared error sense) estimate the gradient. To construct these sampling probabilities, we first need 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* μ *is the mean* $\hat{A}$, *the sampling distribution that minimizes the variance of* μ, *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 12.3) □

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

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

However, in our case, it leads to a chicken-and-egg problem. The optimal sampling distribution requires knowing $(x_i^{(c)}, y_i^{(c)})$, however, we have to sample and clean those points to get those values. In the next section, we discuss how to inexpensively approximate this optimal distribution. As our technique can work with *any* distribution, we are guaranteed convergence no matter how inaccurate this approximation is. However, a better approximation will lead to an improved convergence rate.

7. ESTIMATION

In this section, we make the sampling result of the previous section practical, and estimate the cleaned data.

7.1 Challenges and Goal

The optimal sampling distribution is dependent on a value that we cannot know 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 the data that we have cleaned. 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 we have cleaned a small amount of data, the estimator will be inaccurate. In ActiveClean, we want to be able to make as much progress as possible in the early iterations so this technique may not work. We take an alternative approach, where we try to exploit what we know about data cleaning to produce an estimate for groups of similarly corrupted records. To define "similarly corrupted", we are going to show how we can use the detection step to make this problem tractable.

7.2 Estimation For A Priori Detection

EXAMPLE 6. *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 we have 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, we have a new record (*r_3,$\{1, 2, 3, 6\}$*). We want to 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, let us look at the gradient for linear regression:

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

We see that 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

We can try to approximate the gradient in a way that the effects of features on the gradient are decoupled. Recall, that when we formalized the a priori detection problem, we ensured 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. We will show how we can use this property to construct a coarse estimate of the clean value. The main idea is that if we can calculate average changes for each feature, then given an uncleaned (but dirty) record, we can add these average changes to correct the gradient.

Let us formalize this intuition. Instead of computing the actual gradient with respect to the true clean values, let us compute the conditional expectation given that a set of features and labels f_r, l_r are corrupted:

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

What we mean by corrupted features is 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. We will show that the sampling distribution can be estimated in 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 Δ_{rx} and Δ_{ry} are average change vectors for the corrupted features in r. Without this approximation, if we were to calculate the expected value conditioned on f_r, l_r, we would have to condition on all the combinatorial possibilities.

7.2.2 Deriving M_x, M_y

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 $f'(d) \cdot (d - c)$ 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 12.4 for derivation). Recall that we have a d dimensional feature space and l dimensional label space. 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 12.5 for an example derivation). Δx is a d dimensional vector where each component represents a change in that feature and Δy is an l dimensional vector that represents the change in each of the labels.

7.2.3 More Accurate Early Error Estimates

We chose to use this linearization over alternatives since it avoids amplifying estimation error. If we consider the linear regression gradient:

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

We can rewrite this as a vector in each component:

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

We see that 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 ϵ^2, if ϵ 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 our technique in Section 8.5 against alternatives, and we find that indeed we provide 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. ActiveClean is optimized for small cleaning budgets.

7.2.4 Maintaining Decoupled Averages

This linearization allows us to maintain per feature (or label) average changes and use these changes to center the optimal sampling distribution around the expected clean value. We know how to estimate $\mathbb{E}(\Delta x)$ and $\mathbb{E}(\Delta y)$.

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. We can now make this a vector to represent changes over all features and labels. For those records, that have a corruption in feature or label i, we have a non zero entry. This leads to the following lemma:

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 Δ_{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 Δ_{ry} which is constructed as follows:

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

We finally have an approximation to our sampling weights:

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

7.3 Estimation For Adaptive Case

The same logic still holds in the adaptive setting, however, we have to reformulate what we mean by "similarly" corrupted. Here, we use u dirtiness classes. Instead of conditioning on the features that are corrupted, we condition the classes. So for each error class, we compute a Δ_{ux} and Δ_{uy}. These are the average change in the features given that class and the average change in labels given that class respectively.

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

8. EXPERIMENTS

There are a number of different axes on which we can evaluate ActiveClean. First, we take real datasets and generate various types of errors to illustrate the value of data cleaning in comparison to robust statistical techniques. Next, we explore different prioritization and model update schemes for data cleaning samples. Finally, we evaluate ActiveClean end-to-end in a number of real-world data cleaning scenarios.

8.1 Experimental Setup and Notation

Our main metric for evaluation is a relative measure of trained model with ActiveClean (or an alternative technique) and the model if all of the data is cleaned.

Relative Model Error. Let θ be the model trained on the dirty data, and let θ^* 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

We apply ActiveClean to the following scenarios:

Income Classification (Adult): In this census dataset, our task is to predict the income bracket (binary) from 12 numerical and categorical covariates. There are 45552 data points in this dataset. We use a SVM classifier to predict the income bracket of the person.

Seizure Classification (EEG): In this dataset, our task is to predict the on set of a seizure (binary) from 15 numerical covariates. There are 14980 data points in this dataset. We chose this dataset since the classification task is hard with an accuracy in clean data of 65%. The model that we use is a thresholded Linear Regression.

Handwriting Recognition (MNIST) [2]: In this dataset, our task is to classify 60,000 images of handwritten images into 10 categories. The unique part of this dataset is the featurized data consists of a 784 dimensional vector which includes edge detectors and raw image patches. We use this dataset to explore how we can corrupt the raw data to affect subsequent featurization. The model is an one-to-all multiclass SVM classifier.

Dollars For Docs: We introduced this dataset earlier in the paper, and here we highlight some of the details. The dataset has 240089 records with 5 textual attributes and one numerical attribute. We used a bag-of-words featurization model for the textual attributes which resulted in a 2021 dimensional feature vector. We use this feature vector via a binary SVM to classify whether a research contribution is "not covered" under the declared experimental protocol from the attributes.

8.1.2 Compared Algorithms

Here are the alternative methodologies that we consider:

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 12.6.1)

Discarding Dirty Data. As a baseline we explore model accuracy when dirty data is discarded.

SampleClean (SC) [44]. In SampleClean, we take a sample of data, apply data cleaning, and then train a model to completion.

Active Learning (AL) [20]. We compare against an Active Learning algorithm that integrates with stochastic optimization (See details in Appendix 12.6.2).

ActiveClean Oracle (AC+O): In ActiveClean Oracle, we importance sample points by their clean gradient. This represents the theoretical best that our algorithm could hope to achieve given perfect estimation.

8.2 Does Data Cleaning Matter?

Before we evaluate ActiveClean, we first evaluate the benefits of cleaning on 2 of our example datasets (EEG and Adult). We first explore this problem without sampling to understand which types of data corruption are amenable to data cleaning and which are better suited for robust statistical techniques. We compare 4 schemes: (1) full data cleaning uncorrupted, (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: We simulated high-magnitude random outliers. We select 5% of the examples and features uniformly at random and replace a feature with 3 times the highest feature value.

Systematic Corruption: We simulated innocuous looking (but still incorrect) systematic corruption. We trained the model on the clean data, find the three most important feature (highest weighted). We sort examples by each these features and corrupt the top of examples with the mean value for that feature. At the end, we have corrupted 5% of the examples. It is important to note the some examples can have multiple corrupted features.

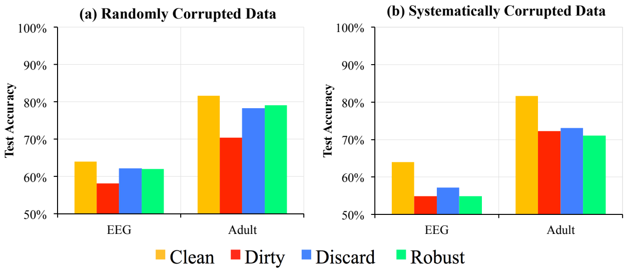


Figure 5: (a) Robust techniques and discarding data work when corrupted data are random and look atypical. (b) Data cleaning can provide reliable performance in both the systematically corrupted setting and randomly corrupted setting.

We plot the test accuracy for models trained on both types of data with the different techniques, since it is a binary classification task with balanced class distributions we cut the plots at 50% (random accuracy) (Figure 5). The robust method performs well on the random high-magnitude outliers with 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 we do not 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, we use the systematic corruption.

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

8.3 ActiveClean: A Priori Detection

The next set of experiments evaluate different approaches to cleaning a sample of data. In this set of experiments, we use the *a priori* case for detection. We assume that we know all of the corrupted records in advance but do not know how to repair them.

8.3.1 Active Learning and SampleClean

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

In our first experiment, we evaluate the samples-to-error tradeoff between three alternative algorithms: ActiveClean (AC), SampleClean, Active Learning, and ActiveClean +Oracle (AC+O). We plot the model error and test accuracy as a function of the number of cleaned records in Figure 6. 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. We find that these gains also correlate well to improvements in test accuracy.

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

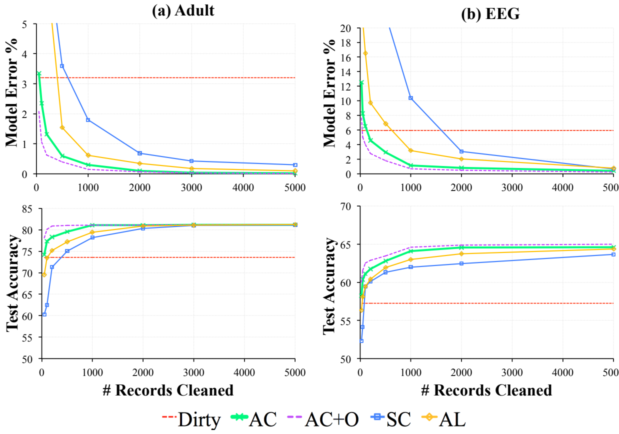


Figure 6: ActiveClean converges with a smaller sample size to the true result in comparison to Active Learning and SampleClean. We show the relative model error as a function of the number of examples cleaned.

8.3.2 Source of Improvements

We pick a single point on the curves shown in Figure 6 that corresponds to 500 records cleaned and compare the performance of ActiveClean with and without various optimizations. This is a vertical slice of the plots in the previous experiments. We denote ActiveClean without detection as (AC-D) (that is at each iteration we sample from the entire dirty data) and ActiveClean without detection and importance sampling as (AC-D-I). In Figure 7, we plot 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. When we remove the important sampling through estimates, ActiveClean is slightly worse (but still comparable to) than Active Learning.

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

8.3.3 Partial Cleaning

We argue that training a model on partially cleaned is an unreliable methodology lacking the same guarantees as Active Learning or SampleClean even in the simplest of cases. It is easy to construct errors for which cleaning more data may counter intuitively increase model error. For thoroughness, we include the tradeoff curves in comparison to ActiveClean for the errors that we generated. In Figure 8, we plot the same curves as the previous experiment comparing ActiveClean, Active Learning, and two partial cleaning algorithms. First, we randomly sample data, clean, and write-back the cleaned data (denoted as PC). Next, we randomly sample data from our dirty data detector, clean, and write-back the cleaned data (denoted as PC+D). We find that 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.

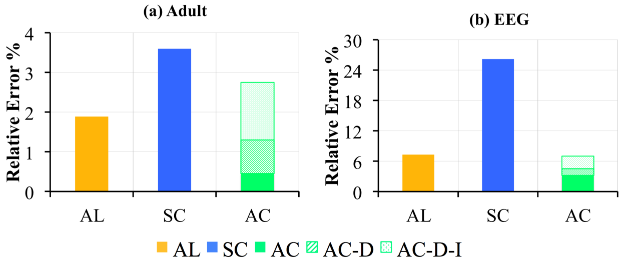


Figure 7: -D denotes no detection, and -D-I denotes no detection and no importance sampling. Both optimization significantly help ActiveClean outperform SampleClean and Active Learning.

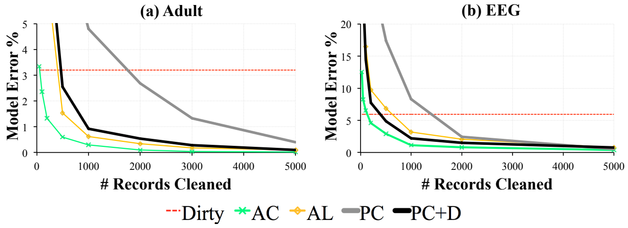


Figure 8: ActiveClean converges with a smaller sample size to the true result in comparison to partial cleaning (PC,PC+D). We show the relative model error as a function of the number of examples cleaned.

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

8.3.4 Corruption Rate

Both Active Learning and ActiveClean outperform SampleClean in our experiments. In our next experiment (Figure 9), we try to understand how much of this performance is due to the initialization (i.e., SampleClean trains a model from "scratch"). We vary the systematic corruption rate and plot the number of records cleaned to achieve 1% relative error for SampleClean and ActiveClean. SampleClean does not use the dirty data and thus is not dependent on this rate. We find that SampleClean outperforms ActiveClean only when corruptions are very severe (45% in Adult and nearly 60% in EEG).

Summary: SampleClean is beneficial in comparison to ActiveClean

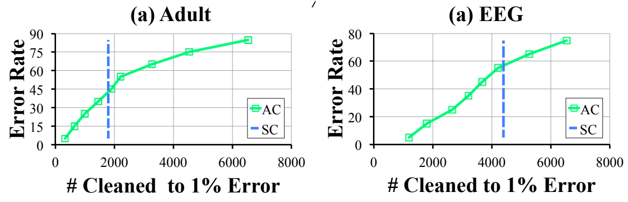


Figure 9: ActiveClean performs well until the corruption is so severe that the dirty model is not a good initialization.

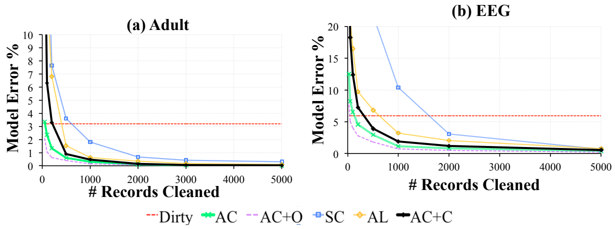


Figure 10: Even with a classifier ActiveClean converges faster than Active Learning and SampleClean.

8.4 ActiveClean: Adaptive Detection

In this experiment, we explore how the results of the previous experiment change when we use an adaptive detector rather than the a priori detector. Recall, in our systematic corruption, we corrupted 3 of the most informative features at random, thus we group these problems into 3 classes. We use an all-versus-one SVM to learn the categorization.

8.4.1 *Basic Performance*

In Figure 10, we overlay 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. We find an interesting tradeoff where initially ActiveClean is comparable to Active Learning, as our 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 this increased error is about 10%.

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.

8.4.2 *Classifiable Errors*

The adaptive case depends on being able to predict corrupted records. For example, random corruption that look like other data may be hard to learn. As corruption becomes more random, the classifier becomes increasingly erroneous. We run an experiment where we start with the systematic corruption described earlier. We increasingly make this 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). In Figure 11, we plot the relative error reduction using a classifier. We find that when the corruption is about 50% random then we reach a break even point where no detection is better.

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

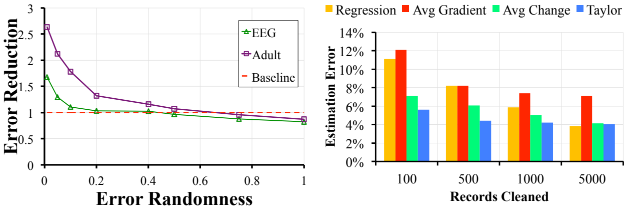


Figure 11: (a) Data corruptions that are less random are easier to classify, and lead to more significant reductions in relative model error. (b) The Taylor series approximation gives more accurate estimates when the amount of cleaned data is small.

8.5 Estimation

In Figure 11b, we explore how estimation technique performs compared to a few alternative estimation techniques: (1) "linear regression" where we train a linear regression model that predicts the clean gradient as a function of the dirty gradient, (2) "average gradient" change where we do not use the detection to inform how to apply the estimate, (3) "average feature change" where we do not linearize but use detection, and (4) our Taylor series linear approximation. We only how accurately the technique can estimate the gradient for sampling. More accurate gradient estimates will result in faster convergence.

Using the EEG dataset, we measure the relative estimation error (relative L2 error with the true gradient) for different amounts of data cleaned. The Taylor series approximation proposed in this work gives more accurate for small cleaning sizes. We give the intuition in Section 7.2.3 where the linearization prevents amplification of errors due to quadratic terms. 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 Scenarios

We evaluate ActiveClean in two real scenarios, one demonstrating the a priori case and one for the adaptive detection case.

8.6.1 *A Priori: Constraint Cleaning*

In our first real scenario, we explore the Dollars for Docs dataset published by ProPublica that we described throughout the paper. We treat this problem as a constraint-based cleaning problem where we encode these problems as data quality constraints. To run this experiment, we manually cleaned the entire dataset up front, and simulated sampling from the dirty data and cleaning by looking up the value in the cleaned data (see Appendix 12.7 for constraints, errors, and cleaning methodology). Figure 12a 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 12b shows the true positive rate (fraction of non-covered research contributions identified) of the classifier as a function of the number of records cleaned. We find that on the dirty data, we can only classify 66% of the positive examples correctly (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: In a real a priori detection scenario, ActiveClean significantly reduces the number of records to clean to train an accurate model.

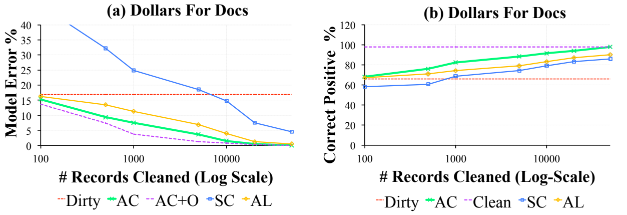


Figure 12: (A) The relative model error as a function of the number of cleaned records. (B) The true positive rate as a function of the number of cleaned records.

8.6.2 *Adaptive: Replacing Corrupted Data*

We consider the following scenario with 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. We devise an "adaptive detection" ActiveClean scenario for this example.

The MNIST dataset consists of 64x64 grayscale images. We run two experiments, in which we have two types of corruptions: (1) 5x5 block removal where take a random 5x5 block from the image and set its pixel values to 0, and (2) Fuzzy where we run a 4x4 moving average over the entire image. We applied these corruptions to a random 5% of the images. We constructed these corruptions to mimic the random vs. systematic corruption that we studied before. The 5x5 block removal behaves much more like a systematic corruption. Typical image processing features are based on edges and corrupting edges leads to ambiguities. On the other hand, the making the image fuzzy is more like a random corruption. We use a 10 class classifier (one for each digit) to detect the corruption.

Figure 13 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, we 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 a real adaptive detection scenario, ActiveClean significantly reduces the number of records to clean to train an accurate model.

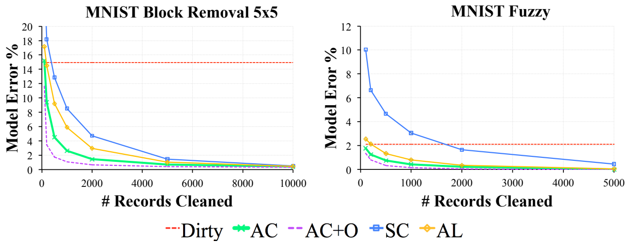


Figure 13: In a real adaptive detection scenario with the MNIST dataset, ActiveClean outperforms Active Learning and SampleClean.

9. RELATED WORK

We highlight some of the key relevant work and how this relates to our proposal.

Stochastic Optimization and Active Learning: Zhao and Tong recently proposed using importance sampling in conjunction with stochastic gradient descent [47]. However, calculating the optimal importance sampling distribution is very expensive and is only justified in our case because data cleaning is even more expensive. Zhao and Tong use a different approximation to work around this problem. This work is one of many in an emerging consensus in stochastic optimization that not all data are equal (e.g., [34]). This line of work builds on prior results in linear algebra that show that some matrix columns are more informative than others [14], and Active Learning which shows that some labels are more informative that others [36]. Active Learning largely studies the problem of label acquisition [36], 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 our method.

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 our work comes from efficiently tuning such a process for expensive data cleaning applications – costs not studied in this field. 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 [15,42]. Learning is performed on a noisy variant of the data which mitigates privacy concerns. The goal is to extrapolate the insights from the noisy data to the hidden real data. Our results are applicable in this setting in the following way. Imagine, we were allowed to query k true data points from the real data, which points are the most valuable to query. This is also related work in adversarial learning [30], where the goal is to make models robust to adversarial data manipulation.

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 [6]. Volkovs et al. explored a related topic of maintaining data cleaning rules that change over time [41]. Recently, in works such as SampleClean [44], 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 [7]. 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 [13]. They explored value of information based prioritization of data acquisition for estimating aggregate queries of sensor readings. 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]

10. CONCLUSION

As analytics frameworks increasingly support Machine Learning, methodologies for optimizing across the entire pipeline from data cleaning to model training are required. In this paper, we propose ActiveClean, a framework that integrates existing data cleaning solutions with Machine Learning model training. ActiveClean information from the model to inform data cleaning on samples, and uses the information from clean samples to update the statistical model. Our experimental results are really promising as they suggest that these optimization can significantly reduce data cleaning costs. Techniques such as Active Learning and SampleClean are not optimized for this setting, and we are able to achieve sharper cost-accuracy tradeoffs than either technique.

There are several exciting, new avenues for future work. First, the data cleaning models that we explore 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. We hope to explore how we can integrate with such frameworks and what we can analyze about the performance. We will also explore what we can do if the analytics at the end of the pipeline is a black box function. Techniques proposed in Reinforcement Learning likely apply as we simultaneously have to learn the properties of the function as we are cleaning.

11. REFERENCES

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

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

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

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

[5] 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.

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

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

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

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

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Towards certain fixes with editing rules and master data. *PVLDB*, 3(1):173–184, 2010.

[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. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2917–2926, 2012.

[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] Z. Qu, P. Richtárik, and T. Zhang. Randomized dual coordinate ascent with arbitrary sampling. *arXiv preprint arXiv:1411.5873*, 2014.

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

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

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

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

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

[40] R. Verborgh and M. De Wilde. *Using OpenRefine*. Packt Publishing Ltd, 2013.

[41] 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.

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

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

[44] 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.

[45] 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.

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

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

12. APPENDIX

12.1 Extensions

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

EXAMPLE 8. *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 7 and Example 8.

12.1.2 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.

12.2 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,

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

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

□

12.3 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 λ and set it equal to zero:

$$\sum P - 1$$

Solving the system of equations, we get:

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

12.4 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:

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

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

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. Δx is a d dimensional vector where each component represents a change in that feature and Δy is an l dimensional vector that represents the change in each of the labels.

12.5 Example M_x, M_y

Linear Regression:

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

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:

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

where

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

we can rewrite this as:

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

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

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:

$$\nabla\phi(x,y,\theta)=\begin{cases}-y\cdot\boldsymbol{x}\text{ if }y\cdot\boldsymbol{x}\cdot\theta\leq1\\0\ \text{ if }y\,\boldsymbol{x}\cdot\theta\geq1\end{cases}$$

Therefore,

$$M_x[i,i]=\begin{cases}-y[i]\text{ if }y\cdot\boldsymbol{x}\cdot\theta\leq1\\0\ \text{ if }y\,\boldsymbol{x}\cdot\theta\geq1\end{cases}$$

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

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

12.6 Experimental Comparison

12.6.1 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.

12.6.2 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 [36]). 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.

12.7 Dollars For Docs Errors and Constraints

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.

12.8 MNIST Errors

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 64×64 matrix) and corrupting them by block removal and fuzzying.

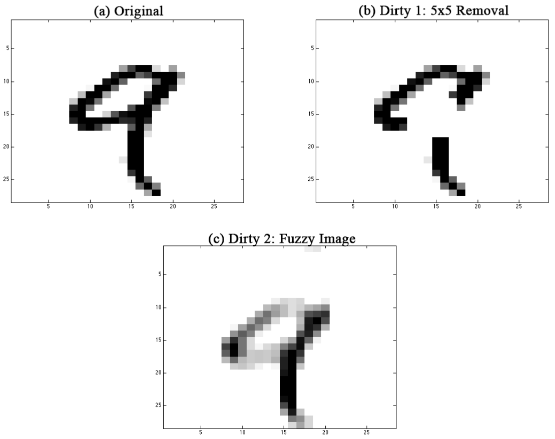


Figure 14: We experiment with two forms of corruption in the MNIST image datasets: 5x5 block removal and making the images fuzzy. Image (a) shows an uncorrupted “9", image (b) shows one corrupted with block removal, and image (c) shows one that is corrupted with fuzziness.