ActiveClean: Interactive Data Cleaning For Modern Machine Learning

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

ABSTRACT

Data cleaning is often an important step to ensure that predictive models, such as regression and classification, are not affected by errors such as inconsistent, out-of-date, or outlier data. Identifying dirty data is often a manual and iterative process, and can be challenging on large datasets. However, many data cleaning workflows can introduce subtle biases into the training processes due to violation of independence assumptions. We propose ActiveClean, a progressive cleaning approach where the model is updated incrementally instead of re-training and can guarantee accuracy on partially cleaned data. ActiveClean supports a popular class of models called convex loss models (e.g., linear regression and SVMs) and record-by-record user-defined data cleaning operations. ActiveClean also leverages the structure of a user's model to prioritize cleaning those records likely to affect the results. Evaluation on four real-world datasets suggests that for a fixed cleaning budget, ActiveClean returns more accurate models than uniform sampling and Active Learning when corruption is systematic and sparse.

1. INTRODUCTION

Building distributed frameworks to facilitate model training on large and growing datasets is a key data management challenge with significant interest in both industry and academia [1, 2, 4, 5]. While these frameworks abstract much of the difficult details of distributed Machine Learning (ML), they seldom offer the analyst any support in terms of constructing the model itself, such as which features to use or how to represent their data. The model construction process is still highly iterative, where through trial-and-error an analyst makes these choices eventually converging onto a model with the desired accuracy. To further complicate matters, data often arrives *dirty*, including missing, incorrect, or inconsistent attributes, due to faulty sensors, software, time delays, or hardware. Thus, part of the iterative model construction process involves identifying potentially dirty data, understanding how they affect the model, and applying techniques to mitigate their effects. While data cleaning is an extensively studied problem, the high dimensionality of many models can amplify even a small amount of erroneous records [7], and the relative complexity (in comparison to SQL analytics) can make it difficult to trace the consequnces of an error.

In prior work, we have noted the choice of data cleaning algorithm can significantly affect results even when using robust ML techniques [3, 6]. In one fraud prediction example, we found that simply applying Entity Resolution before model training improved true positive detection probabilities from 62% to 91%. Despite this importance, in theory and in practice, the academic community has decoupled the data cleaning problem from featurization and ML. This is problematic because many ML techniques often make assumptions about data homogeneity and the consistency of sampling, which can be easily violated if the analyst applies data cleaning in an arbitrary way.

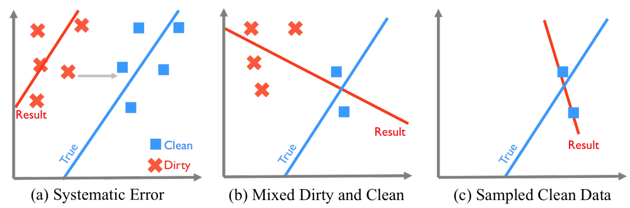

Figure 1: (a) Systematic corruption in one variable can lead to a shifted model. (b) Mixed dirty and clean data results in a less accurate model than no cleaning. (c) Small samples of only clean data can result in similarly inaccurate models.

To understand how this may happen, consider an anlyst training a regression model on dirty data. At first, she may not realize that there are outliers and train an initial model directly on the dirty data. As she starts to inspect the model, she may realize that some records have a large residual value (not predicted accurately). Once she confirms that those records are indeed dirty, she has to design data cleaning rules or scripts to fix or remove the offending records. After cleaning, she re-trains the model–iterating until she no longer finds dirty data. This iterative process is the de facto standard, and in fact encouraged by the design of the increasingly popular interactive "notebook" ML development environments (e.g., IPython), but makes the implicit assumption that model training commutes with incremental data cleaning. This assumption is wholly incorrect; due to the well-known Simpson's paradox, models trained on a mix of dirty and clean data can have very misleading results even in simple scenarios (Figure 1).

In a parallel trend, the dimensionality of the features used in ML models is also rapidly increasing. It is now common to use 100,000s of features in image processing processing problems with techniques such as Deep Learning. Empirically, such feature spaces have facilitated breakthroughs in previously hard classification tasks such as image classification, robot actuation, and speech recognition. However, the pitfall is that the standard approaches for debugging and reasoning about data error may lose their intuition for higher dimensions. In other words, it is often not obvious how an analyst should select which records to clean.

As it stands, there are two key problems in interactive model construction, (1) correctness, and (2) dirty data identification. We address these to problems in a system called ActiveClean which facilitates interactive training-cleaning iteration in a safe way (with expected monotone convergence guarantees) and automatically selects the most valuable data for the analyst to inspect even in the complex models popular in modern ML pipelines. The selection technique applied in ActiveClean uses pointwise gradients to generalize the outlier filtering heuristics to select potentially dirty data even in complex models. The analyst initializes an ActiveClean with an ML model, a featurization function, and the base data, and the ActiveClean initially returns the model trained on the dataset. ActiveClean also returns an array of data sampled from the model that are possibly dirty. The analyst can apply any value transformations to the data and then prompt the system to iterate.

We demonstrate ActiveClean with a visual interface allows analysts to debug complex ML models and understand the effects of dirty data. This interface will allow the analyst to specify the desired model and featurization. It will then visualize a sample of potentially dirty records and allow the analyst to clean the data appropriately. In our demonstration, we will present three experimental scenarios where the models are affected by dirty data:

EXAMPLE 1 (EVENT DETECTION WITH SVMs).

EXAMPLE 2 (VIDEO SEGMENTATION WITH CNNs).

EXAMPLE 3 (TOPIC MODELING WITH LDA).

2. ARCHITECTURE

This section presents the ActiveClean architecture. Figure **??** illustrates the ActiveClean architecture.

2.0.1 Required User Input

Model: The user provides a predictive model (e.g., SVM) specified as a convex loss optimization problem $\phi(\cdot)$ and a featurizer $F(\cdot)$ that maps a record to its feature vector x and label y.

Cleaning Function: The user provides a function $C(\cdot)$ (implemented via software or crowdsourcing) that maps dirty records to clean records as per our definition in Section **??**.

Batches: Data are cleaned in batches of size b and the user can change these settings if she desires more or less frequent model updates. We empirically find that a batch size of 50 performs well across different datasets and use that as a default. A cleaning budget k can be used as a stopping criterion once $C(\dot{)}$ has been called k times, and so the number of iterations of ActiveClean is $T = \frac{k}{b}$. Alternatively, the user can clean data until the model is of sufficient accuracy to make a decision.

2.0.2 Basic Data Flow

The system first trains the model $\phi(\cdot)$ on the dirty dataset to find an initial model $\theta^{(d)}$ that the system will subsequently improve. The *sampler* selects a sample of size b records from the dataset and passes the sample to the *cleaner*, which executes $C(\cdot)$ for each sample record and outputs their cleaned

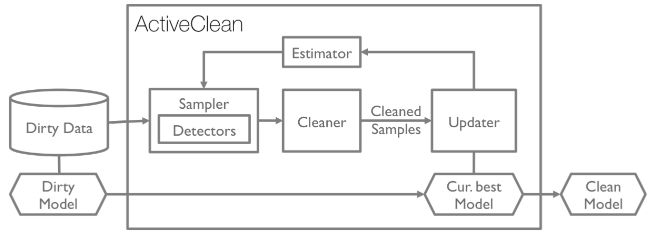

Figure 2: ActiveClean allows users to train predictive models while progressively cleaning data. The framework adaptively selects the best data to clean and can optionally (denoted with dotted lines) integrate with predefined detection rules and estimation algorithms for improved conference.

versions. The *updater* uses the cleaned sample to update the weights of the model, thus moving the model closer to the true cleaned model (in expectation). Finally, the system either terminates due to a stopping condition (e.g., $C(\cdot)$ has been called a maximum number of times k, or training error convergence), or passes control to the *sampler* for the next iteration.

2.0.3 Optimizations

In many cases, such as missing values, errors can be efficiently detected. A user provided *Detector* can be used to identify such records that are more likely to be dirty, and thus improves the likelihood that the next sample will contain true dirty records. Furthermore, the *Estimator* uses previously cleaned data to estimate the effect that cleaning a given record will have on the model. These components can be used separately (if only one is supplied) or together to focus the system's cleaning efforts on records that will most improve the model. Section **??** describes several instantiations of these components for different data cleaning problems. Our experiments show that these optimizations can improve model accuracy by up-to 2.5x (Section **??**).

2.1 Example

The following example illustrates how a user would apply ActiveClean to address the use case in Section **??**:

EXAMPLE 4. *The analyst chooses to use an SVM model, and manually cleans records by hand (the $C(\cdot)$). ActiveClean initially selects a sample of 50 records (the default) to show the analyst. She identifies a subset of 15 records that are dirty, fixes them by normalizing the drug and corporation names with the help of a search engine, and corrects the labels with typographical or incorrect values. The system then uses the cleaned records to update the the current best model and select the next sample of 50. The analyst can stop at any time and use the improved model to predict donation likelihoods.*

3. THE INTERFACE

We will first describe the components of the ActiveClean interface.

3.1 Problem Specification

First, the analyst must specify the Machine Learning problem that they are addressing.

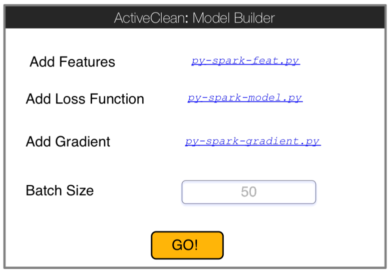

Figure 3: Model Builder

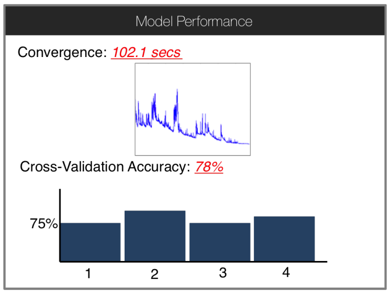

Figure 4: Performance Eval

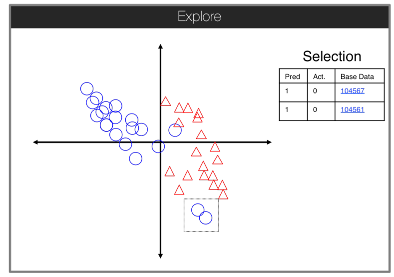

Figure 5: Diagnose Eval

3.2 Performance Evaluation

3.3 Diagnose Interface

3.4 Clean Interface

3.5 Iteration and Updates

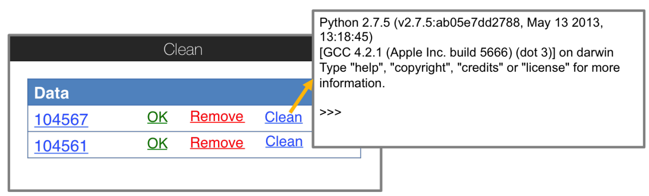

Figure 6: Clean Eval

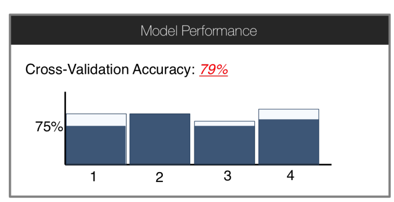

Figure 7: Clean Eval

4. REFERENCES

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

[2] 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), 2014.

[3] A. Arxiv. Activeclean: Arxiv. `http://arxiv.org`.

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

[5] G. Inc. Tensorflow. `https://www.tensorflow.org/`.

[6] 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 *CASE*, 2014.

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