Detecting Outliers in Data with Correlated Measures

Yu-Hsuan Kuo
Dept. of Computer Science & Engineering
Pennsylvania State University
yzk5145@cse.psu.edu

Zhenhui Li
College of Information Sciences & Technology
Pennsylvania State University
jessieli@ist.psu.edu

Daniel Kifer
Dept. of Computer Science & Engineering
Pennsylvania State University
dkifer@cse.psu.edu

ABSTRACT

Advances in sensor technology have enabled the collection of large-scale datasets. Such datasets can be extremely noisy and often contain a significant amount of outliers that result from sensor malfunction or human operation faults. In order to utilize such data for real-world applications, it is critical to detect outliers so that models built from these datasets will not be skewed by outliers.

In this paper, we propose a new outlier detection method that utilizes the correlations in the data (e.g., taxi trip distance vs. trip time). Different from existing outlier detection methods, we build a robust regression model that explicitly models the outliers and detects outliers simultaneously with the model fitting.

We validate our approach on real-world datasets against methods specifically designed for each dataset as well as the state of the art outlier detectors. Our outlier detection method achieves better performances, demonstrating the robustness and generality of our method. Last, we report interesting case studies on some outliers that result from atypical events.

CCS CONCEPTS

• **Mathematics of computing → Robust regression**; • **Computing methodologies → Anomaly detection**; **Mixture models**;

KEYWORDS

Contextual outlier detection; Robust regression

ACM Reference Format:
Yu-Hsuan Kuo, Zhenhui Li, and Daniel Kifer. 2018. Detecting Outliers in Data with Correlated Measures. In *2018 ACM Conference on Information and Knowledge Management (CIKM'18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271798

1 INTRODUCTION

With the development in sensor technology, increasing amount of data collected from sensors become publicly available. Analyzing such data could benefit many applications such as smart city, transportation, and sustainability. For example, New York City (NYC) has released a massive taxi data set [2] including information such as pickup and dropoff locations and time, trip cost, and trip distance. Such data have been used for studies such as characterizing urban dynamics [24], detecting events in city [37], and estimating travel time [31].

In these large-scale sensor datasets, there could be a significant amount of outliers due to sensor malfunction or human operation faults. For example, in NYC taxi data, we have observed trips with extremely long moving distances but unreasonably low trip fares. There are also trips with short displacements between pickup and dropoff locations but have a long trip distance. In a recent work on travel time estimation [31], Wang et al. found that such outliers in the original datasets can break effective travel time estimation methods.

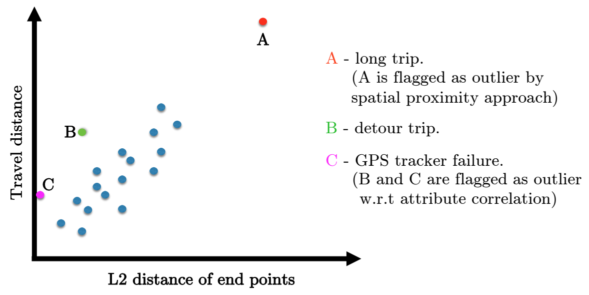


Figure 1: Taxi trip example: suspicious outlying trip

There have been many methods proposed in literature on outlier detection [10]. Typical outlier detection methods define a sample as an outlier if it significantly deviates from other data samples. However, such definition may not apply in our case. Consider an example shown in Figure 1. There could be many interpretations of what is an outlier in this figure. One possibility is point A is an outlier while points B and C are more likely to be labeled as normal points based on the spatial proximity of every datum to its neighbors. However, another possibility is sample A could be a long but normal trip because the ratio between travel distance and L2 distance between end points is within the normal range. On the other hand, sample B and sample C, even though being closer to other data samples, could be outliers. Sample B could be a trip with detour because the travel distance is much longer than L2 distance between end points. Sample C has a nearly zero L2 distance (i.e., the same pickup and dropoff locations), which could be an outlier due to sensor malfunction.

Motivated by the observations on real-world data, we detect outliers based on empirical correlations of attributes, which is close to the *contextual outlier detection* proposed by Song et. al. [27]. For example, we expect correlations between attributes trip time and

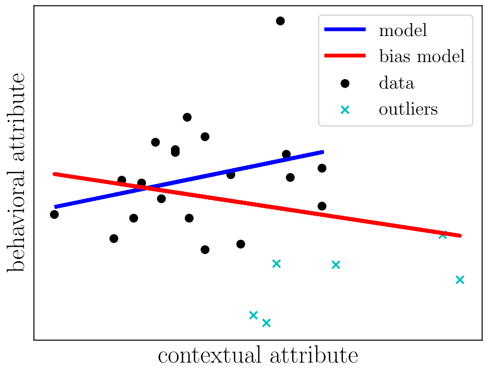


Figure 2: Biased Model Illustration

trip distance in taxi data, and between voltage and temperature in CPU sensor data. If the attributes of a data sample significantly deviate from expected correlations, this data sample is likely to be an anomaly. Domain experts can specify correlation templates so that the definition of an outlier can be customized to the application. We propose a robust regression model that explicitly models the non-outliers and outliers. We feed the algorithm domain knowledge about correlations (e.g., the fact that trip time should be predictable from trip distance & time of day) and it learns how to model them (e.g., how to predict trip time from trip distance and time of day). The model is robust (so outliers do not skew the model parameters) and automatically generates a probability for each data sample being as an outlier and also automatically generates a cut-off threshold on probabilities for outliers.

In literature, there is a series of contextual outlier detection methods that use the correlation between contextual attributes and behavioral attributes to detect outliers [15, 20, 27]. One problem with contextual outlier detection is that outliers can bias a model that is learned from noisy data. To the best of our knowledge, prior work on contextual outlier detection did not consider this issue. The biased model could end up marking outliers as non-outliers and non-outliers as outliers. Take Figure 2 as an example. The blue line indicates a model that would have been learned if it was trained on clean data. However, because clean data is not available, contextual outlier detection trains on noisy data. The red line shows the result. To address this problem, we propose a regression model that explicitly models for outliers and non-outliers.

We conduct experiments on four real-world datasets and demonstrate the effectiveness of our proposed method with comparison to classical regression methods and five existing outlier detection algorithms. With the help of our model, the root cause of outliers can be identified. For example, in the taxi dataset, we found that many outliers are from sensors produced by a certain manufacturer. We report case studies to support our detected outliers and provide insights into the data that can then be used to study new phenomena or devise ways to improve sensors reliability.

In summary, our key contributions are:

- We propose an outlier detection method that utilizes correlations between attributes. Such correlations can be specified by domain experts depending on the application. Different from existing work, our method is a robust regression model that explicitly considers outliers and automatically learns the probability for a data sample being an outlier. It intrinsically generates the thresholds for classification while being robust to parameter skewed by outliers, which is a common problem with other approaches.
- We conduct rigorous experiments on real-world datasets. For these datasets with missing ground truth, human annotation system is used to obtain labels. We design the machine learning task to show that outliers may bias the model trained on unsanitized dataset. We also inject synthetic outliers to validate the model's robustness to different types of outliers.
- We compare our approach against five recent outlier detectors (including other contextual outlier detection algorithms). Our method significantly outperformed competing methods and continues to perform well even in extremely noisy datasets (which are common in big data obtained from sensor measurements).

The rest of the paper is organized as follows. Related work is discussed in Section 2. Section 3 describes the system overview. We present our outlier model in Section 4. We then empirically evaluate our methods in Section 5. We present conclusions in Section 6.

2 RELATED WORK

We outline the progress related to two categories: *unsupervised outlier detection for numerical datasets* and *contextual outlier detection*.

2.1 Unsupervised Outlier Detection

Typical unsupervised outlier detection methods aim to find data samples that are significantly different from other samples. Yamanishi et al. [35] assume that data is generated from an underlying statistical distribution. The notion of outlier is captured by a strong deviation from the presumed data dependent probabilistic distribution. In distance-based outlier work [18, 25, 28], they measure the distance of a data point to its neighbors. The assumption is that normal objects have a dense neighborhood, thus the outlier is the one furthest from its neighbors. Similar approaches using the spatial proximity are density-based [8, 17, 21]. These works adopt the concept of neighbors by measuring the density around a given datum as well as its neighborhood. Breunig et al. [8] introduce a local outlier factor (LOF) for each object in the dataset, indicating its degree of outlierness. The outlier factor is *local* in the sense that the degree depends on how isolated the object is with respect to only neighboring points. These outlier algorithms consider different characteristics and properties of anomalous objects in a dataset. These outlying properties can vary largely on the type of data and the application domain for which the algorithm is being developed. However, all these studies do not consider the outlying behavior with respect to a given context, assuming every attribute contributes equally to the feature vector.

2.2 Contextual Outlier Detection

Another line of works related to our correlation templates is contextual/conditional outlier detection where one set of attributes defines the context and the other set is examined for unusual behaviors. Song et al. [27] propose conditional anomaly detection that takes into account the user-specified environmental variables. Hong et al. [15] model the data distribution by multivariate function and transform the output space into a new unconditional space. Lang et al. [20] model the relationship of behavioral attributes and contextual attributes from local perspectives (i.e., contextual neighbors) as well as global perspectives. However, none of these works build their models under the awareness/assumption of outlier and thus the training process is limited to clean data.

There are also contextual outlier detection for graphs [30, 32] and categorical data [29]. Valko et al. [30] proposed a non-parametric graph-based algorithm to detect conditional anomalies. However it assumes the labeled training set is available. Wang et al. [32] address the problem of detecting contextual outliers in graphs using random walk. Tang et al. [29] identify contextual outliers on categorical relational data by leveraging data cube computation techniques. But they are not applicable to numerical data used in our work.

3 NOTATIONS AND SYSTEM OVERVIEW

A dataset $\mathcal{I}$ is a collection of n records $\{\vec{z}_1, \dots, \vec{z}_n\}$ where each $\vec{z}_i$ has m attributes $\vec{z}_i[1], \dots, \vec{z}_i[m]$.

A *correlation template* is a pair (j, S) where j is a behavior attribute and $S \subseteq \{1, \dots, m\}$ is a set of contextual attributes. This means that the value $\vec{z}_i[j]$ can be predicted from attributes $\vec{z}_i[s]$ for $s \in S$.

To avoid heavy use of sub-subscripts, we will also use the following renaming. For a correlation (j, S), we set y_i to be $\vec{z}_i[j]$ and $\vec{x}_i$ to be the vector of the attribute values in S (i.e. $\vec{x}_i = [\vec{z}_i[s] \text{ for } s \in S]$).

An overview of the outlier detector, called Doc, is shown in Fig. 3. It contains an outlier detector that flags suspicious records.

The inputs to the outlier detector are $\mathcal{I}$ and a set $Corr$ of C correlation templates $Corr = \{(j_c, S_c)\}_{c=1}^{C}$. In different applications, some attributes j are usually associated with outlier behavior; but if its relevant attributes S are not specified by domain experts, the system will take the rest of attributes as S, serving as the context of the behavior.

In the outlier detector, a *filter* is a model that learns how to predict $\vec{z}[j]$ from the $\vec{z}[s]$ for $s \in S$. The goal of each filter is to assign a score t_i to every record indicating its estimated probability that the record is an outlier (this is described in Section 4). Higher score implies its higher probability of being an outlier. The expected number of outliers K is the sum of these scores t_i, and the top K records are flagged as outliers by the filter. When using multiple filters, a record is marked as an outlier if at least one filter marks it as an outlier. We average outlier scores returned from multiple filters as an overall outlier score of a record. The result is a dataset $\tilde{\mathcal{I}}$ in which every record $\vec{z}_i$ has a flag ℓ_i indicating whether it should be considered an outlier ($\ell_i = 1$) or not ($\ell_i = 0$).

The summary of notations is in Table 1.

4 OUTLIER DETECTOR

The job of the outlier detector is to take each correlation template (j, S) and learn a model that, for each record $\vec{z}_i$, can predict $\vec{z}_i[j]$ from the attributes $\vec{z}_i[s]$ for $s \in S$. It then assigns an outlier score t_i to each record $\vec{z}_i$. This score is the estimated probability that the record is an outlier and is based on how much the actual value $\vec{z}_i$ deviates from its prediction.

We do this by modeling the prediction error as a mixture of light-tailed distributions (for non-outliers) and heavy-tailed distributions

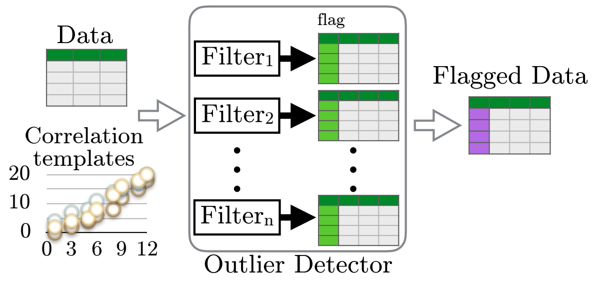


Figure 3: Doc System Overview

Table 1: Notations

(j, S)	Correlation template taken by a filter
$y_i = \vec{z}_i[j]$	Behavioral attribute j value of record $\vec{z}_i$
$\vec{x}_i = [\vec{z}_i[s] \text{ for } s \in S]$	Contextual attributes S of record $\vec{z}_i$
t_i	Outlier score of record z_i provided by a filter.
K	# records flagged as outliers by a filter.
ℓ_i	Outlier flag of record $\vec{z}_i$

(for outliers). Similar noise mixtures are used in robust statistics [9, 16, 26, 33], and typically M-estimators or MCMC inference are used to find model parameters. Instead, we specifically use a variant of expectation-maximization (EM) [13] because it produces variables that, as explained in Section 4.1, can be interpreted as outlier probabilities t_i. Indeed, we are more interested in these t_i than in the model parameters themselves.

We provide an algorithm for linear models in Section 4.1. Linear models are popular because they are not as restrictive as they initially seem – features can be transformed (e.g., by taking logs, square roots, etc.) so that they have an approximately linear relationship with the target. The ideas from Section 4.1 can be extended to more complex models, such as generalized linear models, and learned with variations of the expectation-maximization framework (EM) [13].

Assuming records are independent, the expected number of outliers K is the sum of the outlier probabilities of each record: $K = \lfloor \sum_{i=1}^{n} t_i \rfloor$. This means we can take the records with the top K outlier probabilities and flag them as outliers. Since the system can accept many correlation templates as input, it will be learning many models, and a record is labeled as an outlier if any of these models flag it as an outlier.

4.1 Outlier Data Modeling

For the purpose of simplicity and clearness, we use the following renaming in this section. For a correlation (j, S), we set y_i to be $\vec{z}_i[j]$ and $\vec{x}_i$ to be the vector of the attribute values in S (i.e. $\vec{x}_i = [\vec{z}_i[s] \text{ for } s \in S]$).

Linear models have a weight vector $\vec{w}$, a noise random variable ϵ_i, and the functional form

$$y_i = \vec{w} \cdot \vec{x}_i + \epsilon_i \tag{1}$$

The noise distribution ϵ_i for record i is modeled as follows. We assume that there is a probability p that a data point is an outlier. Hence, the error ϵ_i is modeled as a mixture distribution – with probability $1-p$ it is a zero mean Gaussian with unknown variance σ^2, and with probability p it is a Cauchy random variable. Note that the Gaussian distribution has probability density

$$f_G(\epsilon_i;\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{\epsilon_i^2}{2\sigma^2})$$

The Cauchy distribution with scale parameter b is a heavy-tailed distribution with undefined mean and variance, hence it is ideal for modeling outliers. It is equivalent to the Student's t distribution with 1 degree of freedom [19].

A sample ϵ_i from this distribution can be obtained by first sampling a value τ_i from the Gamma(0.5, b) distribution then sampling ϵ_i from the Gaussian(0, $1/\tau_i$) distribution [6]. The probability of this joint sampling is

$$f_C(\epsilon_i,\tau_i;b) = \frac{b^{0.5}}{\Gamma(0.5)}{\tau_i}^{0.5-1}e^{-b\tau_i}\frac{\sqrt{\tau_i}}{\sqrt{2\pi}}\exp(-\frac{\tau_i\epsilon_i^2}{2})$$

Given $\vec{x}_i$ and y_i, we introduce a latent indicator χ_i to denote where the error of $\vec{x}_i$ comes:

$$\chi_i = \begin{cases} 1 & \text{if the error of } \vec{x}_i \text{ is generated from the Cauchy} \\ 0 & \text{if the error of } \vec{x}_i \text{ is generated from the Gaussian} \end{cases}$$

The expected value of χ_i is denoted by t_i and is automatically computed by the EM algorithm. With the model parameters $\vec{w}$ and unknown noise parameters σ^2 (variance of non-outliers), p (outlier probability), b (scale parameter of outlier distribution), the likelihood function is:

$$\begin{aligned} &L(\vec{w},\sigma^2,p,b,\vec{\chi},\vec{\tau}) \\ &= \prod_{i=1}^{n}\left[(1-p)\frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(y_i-\vec{w}\cdot\vec{x}_i)^2}{2\sigma^2})\right]^{1-\chi_i}\times \\ &\left[p\frac{b^{0.5}}{\Gamma(0.5)}{\tau_i}^{0.5-1}e^{-b\tau_i}\frac{\sqrt{\tau_i}}{\sqrt{2\pi}}\exp(-\frac{\tau_i(y_i-\vec{w}\cdot\vec{x}_i)^2}{2})\right]^{\chi_i} \end{aligned} \quad (2)$$

We iteratively update the estimates of σ^2, p, b, τ_i and t_i (the expected value of χ_i) using the EM framework as described below. Note that the scale parameter b of the Cauchy distribution cannot be estimated using maximum likelihood, so we update it using the interquartile range (the standard technique for Cauchy [5]) as explained below.

4.2 Model Parameters Learning

We employ EM algorithm [13] to solve the above likelihood function L. We iteratively update parameters so we add a superscript $^{(k)}$ to parameters to denote their values at the k^{th} iteration. The E and M steps are described next.

4.2.1 *E step.*

- τ_i update:
  In Eq. (2), τ_i only appears in $e^{-\tau_i(b+0.5(y_i-\vec{w}\cdot\vec{x}_i)^2)}$ (after cancellation), which shows that τ_i (conditioned on the rest of the variables) follows exponential distribution. The conditional expected value of τ_i is

$$\frac{1}{b+0.5(y_i-\vec{w}\cdot\vec{x}_i)^2}$$

  By replacing τ_i with this expectation, the likelihood function L in Eq. (2) is reduced to

$$\begin{aligned} &L(\vec{w},\sigma^2,p,b,\vec{\chi}) \\ &= \prod_{i=1}^{n}\left[(1-p)\frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(y_i-\vec{w}\cdot\vec{x}_i)^2}{2\sigma^2})\right]^{1-\chi_i}\times \\ &\left[p\frac{\sqrt{b}}{\sqrt{\pi}}e^{-1}\frac{1}{\sqrt{2\pi}}\right]^{\chi_i} \end{aligned} \quad (3)$$

- t_i update (here sigmoid$(z) = \frac{1}{1+e^{-z}}$):

$$t_i^{(k+1)} = \text{sigmoid}(\log(\frac{p^{(k)}}{1-p^{(k)}})+0.5\log(\frac{b^{(k)}\sigma^{2(k)}}{\pi e^2})+\frac{(y_i-\vec{w}^{(k)}\cdot\vec{x}_i)^2}{2\sigma^{2(k)}}) \quad (4)$$

- b update: $b^{(k+1)} = \frac{1}{Median(\vec{\xi})}$ where $\vec{\xi}$ is the vector of absolute error $|y_i-\vec{w}^{(k)}\cdot\vec{x}_i|$ for the top K records with highest $t_i^{(k)}$ values (note $K = \lfloor\sum_{j=1}^{n}t_j^{(k)}\rfloor$).

4.2.2 *M step.*

For each iteration before convergence, we update the estimated outlier probability p, the variance of non-outliers σ^2, and the coefficients $\vec{w}$. The updated parameters are listed below.

- p update:

$$p^{(k+1)} = \frac{1}{n}\sum_{i=1}^{n}{t_i}^{(k+1)}$$

- σ^2 update:

$$\sigma^{2(k+1)} = \frac{\sum_{i=1}^{n}(1-{t_i}^{(k+1)})(y_i-\vec{w}^{(k)}\cdot\vec{x}_i)^2}{n-\sum_{i=1}^{n}{t_i}^{(k+1)}}$$

- $\vec{w}$ update: $\vec{w}^{(k+1)}$ is the solution to the weighted least square problem where we give each (y_i, $\vec{x}_i$) a weight (1-$t_i^{(k+1)}$). Specifically, this weight (1-$t_i^{(k+1)}$) tells how much the model should rely on this datum. Thus, if one example has its $t_i^{(k+1)}$ as 1 (with probability 1 as an outlier), then it does not contribute to the $\vec{w}^{(k+1)}$ update coefficients.
  The update for $\vec{w}^{(k+1)}$ is a weighted least squares update:

$$\vec{w}^{(k+1)} \leftarrow \left[{X^\star}^T X^\star\right]^{-1}{X^\star}^T\vec{y^\star}$$

  where $X^\star = VX$, $\vec{y^\star} = V\vec{y}$ and V is a n by n diagonal matrix with $V_{ii} = \sqrt{1-t_i^{(k+1)}}$.

The algorithm will terminate when parameters $\vec{w},\sigma,p,b$ converge. Since each iteration involves finding the median absolute error in b update, the time complexity is $O(n\log n\cdot T)$ where T is the number of iterations.

4.3 Outlier Labeling

As mentioned before, every filter model assigns to every record i a score t_i indicating an estimated probability that it is an outlier and an estimated fraction of outliers p. The filter then labels a record an outlier if it has one of the top K values of t_i where $K = \lfloor \sum_{i=1}^{n} t_i \rfloor \approx p \times n$.

Each filter model identifies different types of outliers. After the data pass through i filters, each record receives i labels $\ell_1, \ell_2, \cdots \ell_i$ from i filters where $\ell_i = 1$ indicates it is an outlier flagged by filter i and $\ell_i = 0$ otherwise. At the end, we add this record to the outlier set if $\ell_1 \vee \ell_2 \vee \cdots \vee \ell_i = 1$.

5 EXPERIMENTS

The outlier detector was implemented using MapReduce. The rest of experiments used a machine with 2.00GHz Intel(R) Xeon(R) CPU and 48 GB RAM.

5.1 Datasets

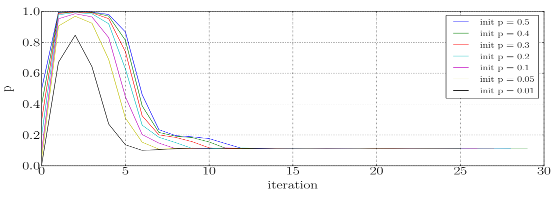
We apply our filtering model to four real-world unlabeled datasets. We assume that records in ElNino and Houses datasets are not corrupted which is also an assumption in [20, 30], so we inject synthetic outliers. Other datasets such as Bodyfat [1] and Algae [3] used in [30] exhibit correlations between attributes. However, we do not consider them in our experiments because the data size is too small.

5.1.1 NYC Taxi. A large-scale 22GB public New York City taxi dataset [2] is collected from more than 14,000 taxis, which contains 173, 179, 771 taxi trips from 01/01/2013 to 12/31/2013. Each record is a trip with attributes: medallion number (anonymized), hack license (anonymized), vendor, rate code, store and forward flag, pickup location, pickup datetime, drop off location, dropoff datetime, passenger count, payment type, trip time, trip distance, fare amount, tips, tax, tolls, surcharge, and total amount. We use the subset of 143,540,889 trips which are within the Manhattan borough (the boundary is queried from wikimapia.org). We examine the outlying behavior in the trip time, distance, and fare.

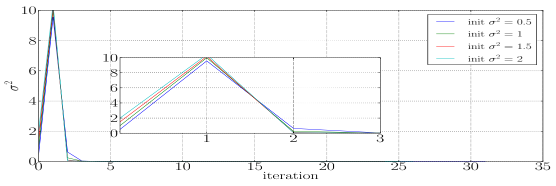
5.1.2 Intel Lab Sensor. This is a public Intel sensor dataset [23] containing a log of about 2.3 million readings from 54 sensors deployed in the Intel Berkeley Research lab between 02/28/2004 to 04/05/2004. Each record is a sensor reading with date, time, sequence number, sensor id, temperature (°C), humidity, light, voltage, and the coordinates of sensors' location. We consider two behavioral attributes as humidity and temperature.

5.1.3 ElNino. This dataset is from UCI repository [11] with 93,935 records after removing records with missing values. These readings are collected from buoys positioned around equatorial Pacific. The sea surface temperature is used as behavior variable while the rest of the oceanographic and meteorological variables are contextual variables.

5.1.4 Houses. This dataset is from UCI repository [4] with 20,640 observations on the housing in California. The house price is used as behavioral attributes and other variables such as median income, housing median age, total rooms, etc. are contextual attributes.



(a) parameter p sensitivity.



(b) parameter σ^2 sensitivity.

Figure 4: Parameter sensitivity of one filter used for sensor dataset.

5.2 Initial Parameters Setting and Sensitivity

We describe how we decide the initial value for p, b, σ^2, and $\vec{w}$ used in the outlier detector.

- $p = 0.05$. We start with 5% as initial value. We observed that the final converged value is not very sensitive to initial settings. Figure 4(a) gives an example of how p changes in the iterations on NYC taxi dataset, with different starting points, to converge to approximately the same value 0.114.
- $\sigma^2 = 1$. As σ^2 represents the variance of non-outliers, it is preferred to initially be a small number. In NYC taxi data, Figure 4(b) shows the convergence path of σ^2 with different starting values in [0.5, 2].
- $\vec{w} = (1, 0, \cdots, 0)$. Let the feature variable be a j dimension vector $\vec{x} = (\vec{x}[1], \vec{x}[2], \cdots, \vec{x}[j])$ and the target variable y. Suppose there is a linear relationship between variables $\vec{x}[1]$ and y. The initial coefficient for $\vec{x}[1]$ is set to be 1, i.e., $\vec{w}[1] = 1$. Others are initialized as 0.
- $b = \pi e^2$. The b value only affects t_i. We choose this setting so that the term $0.5\log(\frac{b\sigma^2}{\pi e^2})$ in the t_i update in Equation 4 equals 0, and thus each data point's t_i value in the beginning of the algorithm is dominated by the error $(y_i - \vec{w} \cdot \vec{x}_i)$.

5.3 Outlier Detection Baselines

We evaluate Doc against the state-of-the-art algorithms including traditional outlier detection, contextual outlier detection, regression models and methods specifically designed for outliers in taxi data.

- **density-based method**. A widely referenced density-based algorithm LOF [8] outlier mining. We implement this method and adopt the commonly used settings for neighbor parameter $k = 10$.
- **distance-based method**. A recent distance-based outlier detection algorithm with sampling [28]. We use the provided code and the default sample size $s = 20$.
- **OLS**. The linear regression with ordinary least square estimation. The outlier score of record i is its Cook's distance $\mathcal{D}_i =$

$\frac{e_i^2}{s^2 p}\left[\frac{h_i}{(1-h_i)^2}\right]$ where e_i is the error of the ith record, s^2 is the mean squared error of the ordinary linear regression model, p is the dimension of feature vector $\vec{x_i}$ and the leverage of record i is $h_i = \vec{x}_i[X^T X]^{-1}\vec{x}_i^{\,T}$.

- **GBT**. The gradient boosting tree regression model [14]. We select parameters from a validation set. The outlier score is defined as the absolute difference between the predicted value and the true value.
- **CAD [27]**. Conditional Anomaly Detection. A Gaussian mixture model U is used to model contextual attributes x where U_i denotes the i-th component. Another Gaussian mixture model V is used to model behavioral attributes y with V_j. Then, a mapping function $p(V_j|U_i)$ is used to compute the probability of V_j being selected under the condition that its contextual variables are generated from U_j. We set the number of Gaussian components as 30. The outlier score is defined as an inverse of the probability computed from this approach.
- **ROCOD [20]**. Robust Contextual Outlier Detection. An ensemble of local expected behavior and global expected behavior is used to detect outliers. For the local behavior, a neighbor-based locality sensitive hashing is used to locate contextual neighbors and an average of neighbors' behavior attribute is considered as expected local estimation. A linear ridge regression or non-linear tree regression is chosen to model the global expected behavior. We chose the non-linear model as its global estimation because it is the best performance in Houses and Elnino datasets reported in their work. The outlier score is computed as the absolute value of a weighted average of global and local estimates minus the true value.
- **SOD [36]**. Smarter Outlier Detection. A method specifically designed for taxi dataset. SOD works by snapping the pickup and dropoff locations to the nearest street segments. The trips which fail to be mapped to the street are considered type I outliers. Next, it computes the shortest path distance and compares that to actual trip distance to detect outliers (called type II outliers). It is worth noting that our outlier filtering model can also employ road network for detecting outliers by simply using the shortest path distance as an input feature. However, we do not do this so that we can give SOD an advantage, while seeing how other features in the data can be used to detect anomalies.

5.4 Experiments on Intel Sensor Data

We describe the filters and confirm detected outliers with Scorpion [34], which also uses the sensor data for evaluation.

5.4.1 Sensor filters. In this dataset, the temperature is correlated with voltage and humidity. We use 2 filters below. The first filter marks 5.2% of total records as outliers ($p = 0.052$) while the second filter marks 11.4% ($p = 0.114$). Among these marked records, 44% are captured by both filters. Note that sensor's age refers to the days or weeks since these sensors were deployed.

(1) log(humidity) = $w_1 \times$ log(temperature) + $\vec{w_a} \cdot \vec{a_d} + \beta_1$, where $\vec{a_d}$ is sensor's age measured in days.
(2) log(temperature) = $w_2 \times$ log(voltage) + $\vec{w'_a} \cdot \vec{a_w} + \beta'_1$, where $\vec{a_w}$ is sensor's age measured in weeks.

5.4.2 Method for Comparison. Because the dataset does not contain ground truth, we validate our detected Intel sensor outliers with findings of Wu and Madden in the Scorpion system [34] where they, using domain knowledge, manually identify one type of outliers.

The problematic sensors claimed in Scorpion are temperature readings $\in$ (90°C, 122°C) generated from sensor 15 and sensor 18 and they account for 5.6% of records in the whole dataset. Approximately 11% of records are flagged by our system Doc, including manually identified outliers in the Scorpion paper. While those manual annotations provide some ground truth (i.e. have high precision), they may not have flagged all outliers (i.e. recall is unknown).

We also compare with linear regression model with ordinary least squares estamatation (OLS). We apply the Cook's distance ($\mathcal{D}$) to estimate the influence, or the combination of leverage and residual values, of each record. Points with large Cook's distance are considered to have further examination. We flag outliers as points with $\mathcal{D} > 4/n$ where n is the number of observations [7]. The result shows that 4.13% of records are flagged as outliers by OLS. We do not choose other outlier detection methods listed in Section 5.3 because none of them provides a threshold in outlier score for users to flag outliers.

5.4.3 Evaluation Metric. We validate flagged outliers by machine learning tasks. In cases where ground truth is missing, it is customary to divide data into training/testing sets. We run our outlier detector on the training set and use the flagged training records to modify the training data (i.e., remove or downweight records suspected of being outliers). Then we build various machine learning models on the modified training data. The goal is to compare these accuracy of the models on a common testing set. The main intuition is that uncaught outliers will degrade the training of the models and thus hurt testing accuracy; better outlier detection algorithms are therefore more likely to result in good training datasets that yield models to perform better on testing data.

To follow this intuition, we use 5-fold cross validation and design prediction tasks with linear and non-linear regression models. The evaluation metrics are mean absolute error (MAE) and mean relative error (MRE). Since there are also anomalous records in the testing data, we also use the median absolute error (MedAE) and median relative error (MedRE). In Table 2, we employ linear regression (LR), support vector regression with quadratic error function (SVR), and decision tree regression (DTR) and we put all attributes as features for the prediction task1 – predicting temperature where temperature is the variable involved in 2 outlier detecting filters. We train the models on four different training sets – all training set, all training set minus Scorpion outliers, all training set minus Doc outliers, and all weighted training set. Note that we use the scikit-learn [22] implementation for the model LR, SVR and DTR.

5.4.4 Results. Results are presented in Table 2. In Task1 with models LR and SVR, removing our detected outliers from training set or down-weighting those outliers results in lower error. We also conduct the Paired Student's t-test and Wilcoxon signed rank test to show that it is statistically significant that the MAE of our modified training set (i.e., training set minus our detected outliers) is lower than the MAE of Scorpion's modified training set.

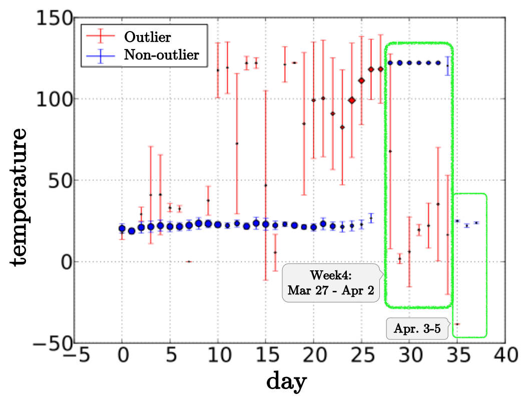


Figure 5: The average temperature readings sequence from 2/28/2004 to 4/5/2004

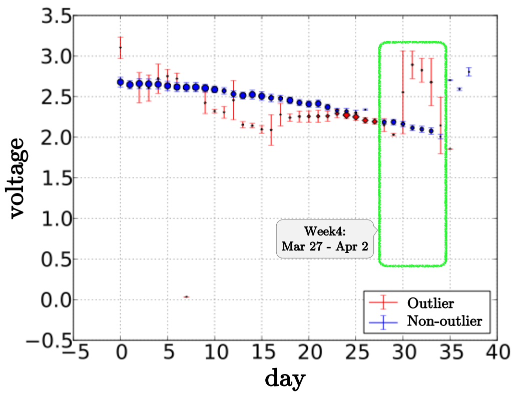


Figure 6: The average voltage readings sequence from 2/28/2004 to 4/5/2004

Note that with the DTR model, using all the training data gets the lower *mean* absolute and relative error. However, because the testing data does contain outliers, the mean can be skewed by them. The median errors (MedAE, MedRE) are more robust measures of performance and show that taking out outliers in training set (or downweighting them) leads to more accurate prediction.

5.4.5 Case Study. 57.2% of flagged outliers are associated with anomalous temperature reading in Week 3. Doc observes a general sensor's malfunction pattern as it is unlikely to be real temperature in the lab – fifty five out of total fifty eight sensors exhibit that the temperature reading is increasing until it reaches around 122°C and it keeps generating 122°C or above in Week 3 (as shown in Figure 5). However, on Week 4, almost all sensors generate temperature $\in [122.15, 175.68)$ – hence that is the norm in the data for that week. This is a very common pattern in the data that 92% of records produced on week 4 generates temperature $\in [122.15, 175.68)$. Hence they are classified as normal by Doc. Note that Scorpion refers to sensors generating high temperature as problematic sensors.

In figure 6, we see that there is a decreasing trend in voltage for this batch of sensors and this helps to justify the fact that records with voltage ≥ 2.8 in Week 4 are identified as outliers by Doc.

5.5 Experiments on NYC Taxi

First we describe five filters we used to detect outliers. We validate the results with human-annotated trips and compare with a method called Smarter Outlier Detection (SOD) [36], which was specifically designed for this dataset. In Section 5.5.5 we also compare against the state of the art outlier detection algorithms.

5.5.1 Taxi filters. We used the following filtering models (where w_i is the coefficient and β_i is offset).

(1) Trip time = $w_1\times$(dropoff time-pickup time)+β_1
(2) Fare = $w_2\times$(total amount-tips-tax-toll-surcharge)+β_2
(3) log(Trip time) = $w_3\times$log(Fare)+β_3
(4) log(Trip distance) = $w_4\times$log(L2)+β_4
(5) log(Trip time) = $w_5 \times$ log(L2) + $\vec{w}_t \cdot \vec{ts} + \beta_5$, where $\vec{ts}$ is the vector of 24-dimension temporal features described below.

Filter 1 and 2 encode what should be functional dependencies. However, they may differ due to software bugs, data entry errors, or device miscalibration. For example, trip time might be recorded by the taxi meter while pickup and drop-off times might be recorded by a gps unit with a separate clock. In Filters 3, 4, 5, trip time/distance/fare/L2-displacement are all positively correlated and we expect their variance to grow proportionally with the length of a trip. For this reason, we use logs (so that multiplicative error becomes additive error). Also, trip time may depend on the time of day (e.g., rush hour), so we include those components in Filter 5. We partition time of day into 2-hour time slots and separate out weekends from weekdays (this giving $12\times2 = 24$ temporal features). We note that filters 3 and 5 did not have overlap in the records they flagged as outliers, thus showing that correlated sensor readings can fail in different ways.

5.5.2 Taxi Outlier Detection Method Comparison. In total, Doc flagged 7% of records as outliers (the dataset is known to be noisy). The code for SOD was not available, so we reproduced it with a different software package.[1] We discarded trips whose end points are not on roads. The dataset can be categorized into four disjoint sets: $M\widetilde{S}$ - records flagged as outliers by Doc but not SOD, $\widetilde{M}S$ - records flagged as outliers by SOD but not Doc, MS - records flagged as outliers by both methods, and $\widetilde{M}\widetilde{S}$ - records not flagged by any method.

5.5.3 Evaluation Metric. We designed a human labeling system for experienced taxi riders to determine outlier trips and to provide reasons to support their judgements. We provided the labelers with the taxi fare rate information from the NYC Taxi & Limousine Commission. Each trip is labeled by three people and we take the majority votes as the ground truth.

To provide a quantitative evaluation, each time the labeling webpage randomly selects 10 trips from each of the 4 sets for a person to label.

5.5.4 Results. In all, 6517 trips were labeled and the results are shown in Table 3. In set $M\widetilde{S}$, 92% of trips were labeled by humans as outliers and these are consistent with our approach.

In set $\widetilde{M}S$, humans only labeled 16% of the records as outliers. Thus, when Doc and SOD disagreed, humans tended to agree with

[1]http://project-osrm.org

Table 2: Performance with/without Outlier

Model	LR		SVR		DTR				DTR (test set without outlier)			
Train Set	MAE	MRE	MAE	MRE	MAE	MRE	MedAE	MedRE	MAE	MRE	MedAE	MedRE
Task1: temperature prediction												
all train	11.94	0.66	14.10	0.75	**3.53**	**0.21**	1.12	0.05	2.15	0.1	1.13	0.05
- Scorpion	11.94	0.66	11.23	0.65	3.54	0.21	1.12	0.05	2.15	0.1	1.14	0.05
- OLS	9.8	0.6	14.83	0.74	3.75	0.38	1.1	0.05	1.88	0.09	1.11	0.05
- Doc	**9.56**	**0.53**	9.49	0.54	5.7	0.36	**0.9**	**0.04**	**1.18**	**0.05**	**0.87**	**0.04**
weighted	**9.56**	**0.53**	**6.91**	**0.46**	5.66	0.37	**0.9**	**0.04**	**1.19**	**0.05**	**0.87**	**0.04**

the classification provided by Doc. For MS and $\widetilde{M}\widetilde{S}$, both our method and SOD get the accuracy of 98% and 94% respectively.

Table 3: 4 Sets of Labeled Trips

Set_i	% trips in Set_i	# labeled trips	$\frac{\text{\# labeled outliers}}{\text{\# labeled trips in } Set_i}$
$M\widetilde{S}$	6.6%	1739	92%
$\widetilde{M}S$	0.093%	1698	16%
MS	0.416%	1570	98%
$\widetilde{M}\widetilde{S}$	90.27%	1510	6%

We use the following evaluation criteria for overall outlier detection performance: detection rate ($DR = \frac{TP}{TP+FN}$, i.e., fraction of outliers that are successfully detected as outliers), false positive rate ($FPR = \frac{FP}{FP+TN}$, i.e., fraction of normal records that are predicted to be outlier), precision ($Precision = \frac{TP}{TP+FP}$, i.e., fraction of detected outlier that are real outlier), true negative rate ($TNR = \frac{TN}{TN+FP}$, i.e., fraction of non-outlier that are detected as non-outlier).

Note that we use the labeled sampled trips to estimate the ground truth statistics for entire dataset. The estimation approach is as follow: suppose, in set i, the total number of trips is u; the number of sampled trips labeled is v; out of labeled trips v the outliers account for ϕ %. Hence $u \times \phi$ is the estimated outliers for set i. The evaluation on both the labeled trips (denoted as on labeled) and estimated results for all trips (denoted as on all) are presented in Table 4. From the results it is clear that Doc achieves much better detection rates for slightly larger false positive rates.

Table 4: Outlier Detection Performance

	Our method Doc		Competitor SOD	
	on all	on labeled	on all	on labeled
DR	**0.55**	**0.89**	0.035	0.52
FPR	0.006	**0.057**	**0.001**	0.484
Precision	**0.924**	**0.94**	0.83	0.55
TNR	**0.99**	**0.942**	0.99	0.515

5.5.5 *Outlier Detection Methods Comparison.*

We evaluate Doc against traditional outlier detection and contextual outlier detection approaches described in Section 5.3. We also adopt the following statistical-based method as baseline.

statistical-based method. Since we observe some detour trips in the taxi data. We fit the ratio of travel distance and L2 distance between end points into Gaussian distribution. The outlier score of point x is $1 - p(x)$ where $p(x)$ is the gaussian density function.

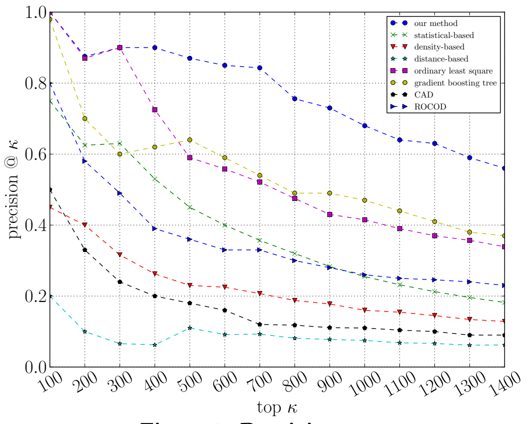


Figure 7: Precision at κ

Evaluation metrics. We randomly sampled 22,463 trips as input data to these outlier detectors which give the outlier rank to every trip. The labeling process uses the same type of methodology mentioned in Section 5.5.3. Given the outlier rank of the trips, we first select every 5 trips to be labeled (i.e., 5^{th}, 10^{th} ...). Thus we get a rough idea of the approximate number of outliers in this sampled dataset. We label the top 1400 trips for each method and use the following metrics for evaluation.

$$\text{Precision } @\kappa = \frac{\text{\# trips whose rank} \leq \kappa \text{ and label = Outlier}}{\kappa}$$

Fig. 7 shows that our method outperforms others. The top outlier trips detected by density-based and distance-based methods are long trips such as trips from upper to lower manhattan. Even though they have less neighbors than shorter trips, their trip information is considered as reasonable by the labeled outcome. In contrast, our top outlier trips are mainly from device error and thus it could be obviously identified by people. For the linear and non-linear regression model as well as the existing contextual outlier detection (CAD & ROCOD), they can identify extreme outliers in their top 100 outliers. But their precision drops with more false positives which is due to the biased prediction. We find that trips with rank greater than 1200 are mostly labeled as non-outliers. Hence the precision @κ drops around $\kappa = 1000$.

5.5.6 *Case Study.*

We describe the anomalous trips and interesting findings. First, we point out the payment systems and trip time tracking systems provided by Creative Mobile Technologies (CMT) are programmed differently from those provided by Verifone (VTS). We expect that the travel time should be consistent with the duration of dropoff time subtracted by pickup time. Our identified outliers show that this is not always the case and the discrepancies are almost always associated with vendor CMT. Similarly, the sum of cost fields (i.e.,Tip, Tax, Surcharge, Toll, Fare) should equal to total amount (Total). Those inconsistent cost related fields are all produced by the VTS payment system.

Second, we identify a type of outliers ranked lower as compared to those extreme corrupted records. These records contain trip fare < \$3. The metered fare regulated by Taxi & Limousine Commission (TLC) initially charges \$2.5 once a passenger gets in taxis and plus \$0.5 per 0.2 mile or \$0.5 per minute in slow traffic. We believe these records are outliers as they are even less than minimum taxi fare.

Last, 1% of detected anomalous records are trips with almost the same GPS coordinates from pickup to dropoff location. To investigate this further, we found the park-cemetery manhattan neighborhood and bridges linking manhattan to nearby boroughs (where gps signal might be weak) are highly correlated with these trips.

5.6 Experiments on Synthetic Outlier Data

For the Elnino and Houses datasets, we inject synthetic outliers into the original clean data. One perturbation scheme used in [20, 27] is that they first randomly select a sample $\vec{z_i} = (\vec{x_i}, y_i)$ then, from k data points of the entire dataset, select another sample $\vec{z_j} = (\vec{x_j}, y_j)$ where the difference between y_i and y_j is maximized. This new data point $(\vec{x_i}, y_j)$ is added as an outlier. We do not follow this scheme for several reasons. First, swapping the attribute values may not always obtain desired outliers. It is likely that most of the swaps could result in normal data. Second, as we observe many extreme outliers in the real-world datasets, swapping values between samples in a clean data is less likely to produce this extreme difference between y_i and y_j. Here we present another way to generate outliers and we explore different types of outliers where we give controls to where and how many outliers are injected or its degree of outlierness.

5.6.1 Perturbation Scheme. To inject $q \times N$ outliers into a dataset with N data samples, we randomly select $q \times N$ records $\vec{z_i} = (\vec{x_i}, y_i)$ to be perturbed. Let y_i be the target attribute for perturbation. Let $\vec{x_i}$ be the rest of attributes. For all selected records, a random number from $(0, \alpha)$ is added up to y_i as $y_i{}'$. Then we add new sample $\vec{z}\,' = (\vec{x_i}, y_i{}')$ into the original dataset and flag it as outlier. Note that original N data samples are flagged as non-outlier. In the experiments, we standardized the target attribute to range (18, 30) which are the min and max value of the behavioral attribute in Elnino dataset. Set α as 50 by default.

5.6.2 Evaluation Metric. Since all these outlier detection approaches considered in Section 5.3 give rank to each record according to outlier score, the Precision-Recall curve (PRC) is obtained by Precision @κ and Recall @κ for all possible κ where the first κ ranked records are determined to be outlier. The evaluation metric we use here is the Area Under the Curve (AUC) of the Precision-Recall curve instead of the Receiver Operating Characteristic (ROC) as it is less informative in imbalanced class problem [12].

5.6.3 Results. As up to 6% of records in Sensor dataset are flagged as outliers due to sensor malfunction, we vary the perturbation ratio q from 0.01 to 0.15 to see if our model is robust in the presence of a large fractions of anomalies. The performance in terms of AUC is shown in the following tables.

Table 5 presents the results when we perturb behavioral attributes to generate outliers. Doc consistently perform the best and its difference compared to other methods becomes significant when more outliers are involved ($q > 0.05$). Another type of synthetic outliers is produced by adding noise to contextual attributes. To see how it affects the performance, we select features with highest Pearson correlation to behavioral attribute for perturbation. In Table 6, we observe that a small fraction of outliers in contextual attribute could hurt the performance considerably for the other methods, especially for the tree-based approaches such as ROCOD and GBT on these two datasets. However, our method is robust and resistant to the fraction of outliers.

We next investigate degree of outlierness of the injected anomalies. As α increases, larger magnitude of noise will have more chance to be added to the original value. Consequently, there are more extreme outliers and our performance is increased as expected in Table 7.

6 CONCLUSIONS

Motivated by a real-world problem, we develop a system Doc which aims to detect outliers and explicitly considers outliers effect in modeling. It is a robust outlier detector as compared to the existing algorithms built on all the data records where their model parameters are skewed by outliers. Our method could potentially facilitate the public or research use of large-scale data collected from a network of sensors.

7 ACKNOWLEDGMENTS

The work was partially supported by NSF grants 1054389, 1544455, 1702760, 1652525. The views in this paper are those of the authors, and do not necessarily represent the funding institutes.

REFERENCES

[1] [n. d.]. Bodyfat Data from CMU statlib. http://lib.stat.cmu.edu/datasets/bodyfat.
[2] [n. d.]. New York City Taxi Data. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.
[3] [n. d.]. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Coil+1999+Competition+Data.
[4] [n. d.]. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/machine-learning-databases/housing/.
[5] Barry C Arnold and Robert J Beaver. 2000. The skew-Cauchy distribution. *Statistics & probability letters* (2000).
[6] Narayanaswamy Balakrishnan and Chin-Diew Lai. 2009. *Continuous bivariate distributions.* Springer Science & Business Media.
[7] Kenneth A Bollen and Robert W Jackman. 1985. Regression diagnostics an expository treatment of outliers and influential cases. *Sociological Methods & Research* (1985).
[8] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers *(In* SIGMOD*).*
[9] Rafael E. Carrillo, Tuncer C. Aysal, and Kenneth E. Barner. 2010. A Generalized Cauchy Distribution Framework for Problems Requiring Robust Behavior. *EURASIP J. Advances in Signal Processing* (2010).
[10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 15.
[11] Di Cook. [n. d.]. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/El+Nino.
[12] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning.* ACM, 233–240.
[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B* 39 (1977), 1–38.

Table 5: AUC of Precision Recall Curve w.r.t different fractions of synthetic outliers in behavioral attribute

	Elnino					Houses				
method	q=0.01	q=0.03	q=0.05	q=0.1	q=0.15	q=0.01	q=0.03	q=0.05	q=0.1	q=0.15
Doc	**0.96**	**0.97**	**0.98**	**0.98**	**0.98**	**0.93**	**0.92**	**0.93**	**0.95**	**0.96**
ROCOD (non-linear)	0.73	0.73	0.74	0.73	0.72	0.50	0.49	0.50	0.49	0.50
CAD	0.80	0.84	0.86	0.85	0.88	0.58	0.67	0.68	0.72	0.75
OLS	**0.96**	0.95	0.95	0.92	0.90	0.92	0.91	0.92	0.91	0.91
GBT	**0.96**	0.95	0.95	0.92	0.90	**0.93**	0.91	0.92	0.91	0.91
distance-based	0.81	0.74	0.77	0.83	0.60	0.76	0.19	0.57	0.4	0.39
density-based	0.21	0.38	0.45	0.38	0.34	0.84	0.58	0.46	0.53	0.58

Table 6: AUC of Precision Recall Curve w.r.t different fractions of synthetic outliers in contextual attribute

	Elnino					Houses				
method	q=0.005	q=0.01	q=0.03	q=0.05	q=0.07	q=0.005	q=0.01	q=0.03	q=0.05	q=0.07
Doc	**0.97**	**0.95**	**0.97**	**0.98**	**0.98**	**0.86**	**0.80**	**0.88**	**0.88**	**0.91**
ROCOD (non-linear)	0.01	0.01	0.02	0.02	0.03	0.03	0.01	0.02	0.04	0.05
CAD	0.80	0.83	0.86	0.88	0.87	0.51	0.54	0.56	0.61	0.63
OLS	0.92	0.86	0.68	0.45	0.32	0.84	0.75	0.71	0.59	0.50
GBT	0.11	0.15	0.28	0.37	0.40	0.04	0.04	0.08	0.11	0.15
distance-based	0.88	0.74	0.81	0.50	0.83	0.54	0.73	0.22	0.20	0.42
density-based	0.08	0.07	0.08	0.09	0.10	0.01	0.01	0.03	0.04	0.06

Table 7: AUC of Precision Recall Curve w.r.t degree of outlierness α in contextual attribute

	Elnino					Houses				
method	$\alpha = 20$	$\alpha = 30$	$\alpha = 50$	$\alpha = 100$	$\alpha = 300$	$\alpha = 30$	$\alpha = 50$	$\alpha = 100$	$\alpha = 300$	$\alpha = 500$
Doc	**0.91**	**0.94**	**0.95**	**0.98**	**0.99**	**0.75**	**0.8**	**0.94**	**0.97**	**0.99**
ROCOD (non-linear)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
CAD	0.78	0.8	0.83	0.87	0.93	0.37	0.54	0.58	0.74	0.85
OLS	0.88	0.89	0.86	0.85	0.73	0.72	0.75	0.87	0.86	0.83
GBT	0.17	0.17	0.15	0.17	0.17	0.04	0.04	0.03	0.02	0.01
distance-based	0.21	0.79	0.74	0.88	0.91	0.14	0.73	0.79	0.85	0.80
density-based	0.13	0.10	0.07	0.05	0.04	0.01	0.01	0.01	0.02	0.05

[14] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[15] Charmgil Hong and Milos Hauskrecht. 2015. MCODE: Multivariate Conditional Outlier Detection. *arXiv preprint arXiv:1505.04097* (2015).

[16] Peter J Huber et al. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics* (1964).

[17] Wen Jin, Anthony K. H. Tung, and Jiawei Han. 2001. Mining Top-n Local Outliers in Large Databases. In *KDD*.

[18] Edwin M. Knorr and Raymond T. Ng. 1998. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB*.

[19] Kenneth L Lange, Roderick JA Little, and Jeremy MG Taylor. 1989. Robust statistical modeling using the t distribution. *J. Amer. Statist. Assoc.* (1989).

[20] Jiongqian Liang and Srinivasan Parthasarathy. 2016. Robust Contextual Outlier Detection: Where Context Meets Sparsity. In *CIKM*. ACM.

[21] Spiros Papadimitriou, Hiroyuki Kitagawa, Philip B Gibbons, and Christos Faloutsos. 2003. Loci: Fast outlier detection using the local correlation integral. In *ICDE*.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[23] Carlos Guestrin Sam Madden Mark Paskin Peter Bodik, Wei Hong and Romain Thibaux. [n. d.]. Intel Lab Sensor Data. http://db.csail.mit.edu/labdata/labdata.html.

[24] Xinwu Qian, Xianyuan Zhan, and Satish V Ukkusuri. 2015. Characterizing urban dynamics using large scale taxicab data. In *Engineering and Applied Sciences Optimization*. Springer, 17–32.

[25] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*.

[26] GJM Rosa, Carlos R Padovani, and Daniel Gianola. 2003. Robust linear mixed models with normal/independent distributions and Bayesian MCMC implementation. *Biometrical Journal* (2003).

[27] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. 2007. Conditional anomaly detection. *ICDE* (2007).

[28] Mahito Sugiyama and Karsten Borgwardt. 2013. Rapid distance-based outlier detection via sampling. In *NIPS*.

[29] Guanting Tang, Jian Pei, James Bailey, and Guozhu Dong. 2015. Mining multidimensional contextual outliers from categorical relational data. *Intelligent Data Analysis* (2015).

[30] Michal Valko, Branislav Kveton, Hamed Valizadegan, Gregory F Cooper, and Milos Hauskrecht. 2011. Conditional anomaly detection with soft harmonic functions. In *ICDM*. IEEE.

[31] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2016. A simple baseline for travel time estimation using large-scale trip data. In *SIGSPATIAL*. ACM.

[32] Xiang Wang and Ian Davidson. 2009. Discovering contexts and contextual outliers using random walks in graphs. In *ICDM*. IEEE.

[33] Mike West. 1987. On scale mixtures of normal distributions. *Biometrika* (1987).

[34] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining away outliers in aggregate queries. In *VLDB Journal*.

[35] Kenji Yamanishi, Jun-Ichi Takeuchi, Graham Williams, and Peter Milne. 2000. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *KDD*.

[36] Jianting Zhang. 2012. Smarter outlier detection and deeper understanding of large-scale taxi trip records: a case study of NYC. In *KDD International Workshop on Urban Computing*.

[37] Yu Zheng, Huichu Zhang, and Yong Yu. 2015. Detecting collective anomalies from multiple spatio-temporal datasets across different domains. In *SIGSPATIAL*.