Automatic Outlier Detection: A Bayesian Approach

Jo-Anne Ting*, Aaron D'Souza† Stefan Schaal*‡

*Computer Science, University of Southern California, Los Angeles, CA 90034
†Google, Inc. Mountain View, CA 94043
‡ATR Computational Neuroscience Labs, Kyoto 619-0288, Japan
Email: joanneti@usc.edu, adsouza@google.com, sschaal@usc.edu

***Abstract*— In order to achieve reliable autonomous control in advanced robotic systems like entertainment robots, assistive robots, humanoid robots and autonomous vehicles, sensory data needs to be absolutely reliable, or some measure of reliability must be available. Bayesian statistics can offer favorable ways of accomplishing such robust sensory data pre-processing. In this paper, we introduce a Bayesian way of dealing with outlier-infested sensory data and develop a "black box" approach to removing outliers in real-time and expressing confidence in the estimated data. We develop our approach in the framework of Bayesian linear regression with heteroscedastic noise. Essentially, every measured data point is assumed to have its individual variance, and the final estimate is achieved by a weighted regression over observed data. An Expectation-Maximization algorithm allows us to estimate the variance of each data point in an incremental algorithm. With the exception of a time horizon (window size) over which the estimation process is averaged, no open parameters need to be tuned, and no special assumption about the generative structure of the data is required. The algorithm works efficiently in real-time. We evaluate our method on synthetic data and on a pose estimation problem of a quadruped robot, demonstrating its ease of usability, competitive nature with well-tuned alternative algorithms and advantages in terms of robust outlier removal.**

I. INTRODUCTION

Robotic systems and their control mechanisms rely crucially on the quality of sensory data in order to make robust control decisions. While certain sensors such as potentiometers or optical encoders are inherently easy to assess in their noise characteristics, other sensors such as visual systems, GPS devices and sonar sensors can provide measurements that are infested by outliers. Thus, robust and reliable outlier removal is mandatory in order to include these types of data in control processes. Our particular application domain of legged locomotion is especially vulnerable to perceptual data of poor quality, as one undetected outlier can potentially disturb the balance controller to the point that the robot loses stability.

An outlier is generally defined as an observation that "lies outside some overall pattern of distribution" [1]. Outliers may arise from sensor noise (producing values that fall outside the valid range of values), from temporary sensor failures, or from unanticipated disturbances in the environment (e.g., a brief change of lighting conditions for a visual sensor). A typical approach of detecting outliers is to characterize what normal observations look like, and then to single out samples that deviate from these normal properties. Existing methods for outlier detection include methods that classify a data point based on a (Mahalanobis) distance from the expected value, approaches that use information-theoretic principles, such as selecting the subset of data points that minimize the prediction error, and techniques that assume that the data was generated by some special generative model.

Outlier classification based on a Mahalanobis distance can work quite well, but tends to require the setting of some threshold that defines whether a point is an outlier or not. This threshold value typically needs to be tuned manually beforehand in order to determine its empirically optimal value for the system. In information-theoretic approaches, outlier detection may be done through active learning [2], clustering (such as k-means [3]) [4] [5] or mixture models [6] [7]. These methods may require sampling, the setting of certain parameters (i.e. the optimal k in k-means), and may not all lend themselves to a real-time implementation, as required in robotics, where sensor data are made available one at a time and need to be discarded once they have been observed. Another commonly used method in this second category is the Random Sample Consensus (RANSAC) algorithm [8]. RANSAC tries to find the subset of data points that produces the lowest error in an iterative fashion. Unfortunately, this may be too computationally intensive for real-time applications and may involve heuristic methods to narrow down the searchable space of subsets. An example of an approach that falls in the third (and second) class of methods is mixture models. It assumes that the data was generated by some underlying structure (e.g. a mixture of a Gaussian distribution and a uniform distribution [9] [10] [11]). The probabilistic assumptions of this approach, however, can be potentially restrictive and may not work as well on datasets where outliers and inliers are not demarked by a large margin.

The ideal algorithm should detect and remove outliers in real-time—without the need for sampling, model assumptions or any action on the user's part. In addition, the algorithm should adjust to drifting functions, since the working conditions of real sensors usually change over time. In this paper, we propose a novel Bayesian algorithm that has many of these properties. It is able to automatically detect outliers in general linear models in a "black box"-like way. We consider linear regression to start, since nonlinear functions can be treated with locally weighted methods [12] in a similar fashion. We first introduce our Bayesian algorithm, before presenting a modified version that can be implemented in real-time (i.e. it can handle data arriving sequentially, point by point, over time). Finally, we evaluate our algorithm on both synthetic and robotic data, demonstrating how it performs at least as well as other standard approaches. In certain cases, it outperforms well-tuned alternative methods.

II. OUTLIER DETECTION IN LINEAR REGRESSION

Let us assume that we have a dataset $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$ that has N data points, each consisting of a d-dimensional input vector $\mathbf{x}_i$ (where d is the number of input dimensions) and a scalar output y_i. Only the input and output data are observed. We can arrange the input vectors $\mathbf{x}_i$ in the rows of the matrix $\mathbf{X}$ and set the corresponding scalar outputs y_i to be the coefficients of the vector $\mathbf{y}$. A general model for linear regression is then:

$$y_i = \beta^T \mathbf{x}_i + \epsilon_{y_i} \tag{1}$$

where β is a d-dimensional vector and ϵ_{y_i} is additive mean-zero noise. The Ordinary Least Squares (OLS) estimate of the regression vector β_{OLS} is $\left(\mathbf{X}^T\mathbf{X}^{-1}\right)\mathbf{X}^T\mathbf{y}$. However, it is not uncommon for observed data to have outliers, and if outliers are not removed, the regression estimate β_{OLS} will be biased.

A. Bayesian Regression for Automatic Outlier Detection

Now, let us take (1) and modify the model so that the observed outputs $\mathbf{y}$ have heteroscedastic variances, i.e. unequal variances. We adopt a weighted linear regression model and introduce a weight w_i for each y_i such that the variance of y_i is weighted with w_i, as done in [13]. However, the weighted least squares regression model in [13] assumes that the weights are known and given. Using incorrect estimates for the weights may lead to deteriorated performance, and common approaches for estimating the weights include modeling the weights to be inversely proportional to the sample output variance [14]. In comparison, we adopt a different approach and treat the weights $\mathbf{w}$ probabilistically. This Bayesian approach is similar to the variational Bayesian algorithm for robust regression proposed in [11] by Faul & Tipping. However, [11] models the regression vector β probabilistically with a Gaussian prior, and hyperparameters are introduced in order to have automatic relevance determination (ARD) [15] on the input data features. This approach adopts a mixture model to explain outliers, using either a uniform or Gaussian distribution to capture them.

Our model is a Bayesian treatment of weighted regression that is able to detect and eliminate outliers automatically. We make the following standard assumptions about the probability distributions of the random variables:

$$\begin{aligned} y_i &\sim \text{Normal}\left(\beta^T\mathbf{x}_i, \sigma^2/w_i\right) \\ \beta &\sim \text{Normal}\left(\beta_0, \mathbf{\Sigma}_{\beta,0}\right) \\ w_i &\sim \text{Gamma}\left(a_{w_i}, b_{w_i}\right) \end{aligned} \tag{2}$$

where β_0 is the prior mean of β and a d-dimensional vector; $\mathbf{\Sigma}_{\beta,0}$ is the prior covariance of β and a d by d diagonal matrix; and σ^2 is the variance of the mean-zero normally distributed output noise. We can treat this entire regression problem as an Expectation-Maximization-like (EM) learning problem [16] [17]. Our goal is to maximize the log likelihood $\log p(\mathbf{y}|\mathbf{X})$, which is called the "incomplete" log likelihood, since the hidden probabilistic variables are marginalized out. However, due to analytical issues, we do not have access to this incomplete log likelihood, but instead, only a lower bound of it. The lower bound is based on an expected value of the "complete" data likelihood $\langle \log p(\mathbf{y}, \beta, \mathbf{w}|\mathbf{X})\rangle$ [1], formulated over all variables of the learning problem, where $\log p(\mathbf{y}, \beta, \mathbf{w}|\mathbf{X})$ is:

$$\sum_{i=1}^N \log p(y_i|\mathbf{x}_i, w_i, \beta) + \log p(\beta) + \sum_{i=1}^N \log p(w_i) \tag{3}$$

The expectation of this complete data likelihood should be taken with respect to the true posterior distribution of all hidden variables $Q(\beta, \mathbf{w})$. Since this is an analytically intractable expression, a lower bound can be formulated using a technique from variational calculus where we make a factorial approximation of the true posterior as follows: $Q(\beta, \mathbf{w}) = Q(\beta)Q(\mathbf{w})$. While losing a small amount of accuracy, all resulting posterior distributions over hidden variables become analytically tractable. The final posterior EM-update equations are listed below:

$$\mathbf{\Sigma}_\beta = \left(\mathbf{\Sigma}_{\beta,0}^{-1} + \frac{1}{\sigma^2}\sum_{i=1}^N \langle w_i\rangle \mathbf{x}_i\mathbf{x}_i^T\right)^{-1} \tag{4}$$

$$\langle\beta\rangle = \mathbf{\Sigma}_\beta\left(\mathbf{\Sigma}_{\beta,0}^{-1}\beta_0 + \frac{1}{\sigma^2}\sum_{i=1}^N \langle w_i\rangle y_i\mathbf{x}_i\right) \tag{5}$$

$$\langle w_i\rangle = \frac{a_{w_i,0} + \frac{1}{2}}{b_{w_i,0} + \frac{1}{2\sigma^2}\left(y_i - \langle\beta\rangle^T\mathbf{x}_i\right)^2 + \frac{1}{2\sigma^2}\mathbf{x}_i^T\mathbf{\Sigma}_\beta\mathbf{x}_i} \tag{6}$$

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^N\left[\left(y_i - \langle\beta\rangle^T\mathbf{x}_i\right)^2 + \mathbf{x}_i^T\mathbf{\Sigma}_\beta\mathbf{x}_i\right] \tag{7}$$

These update equations need to be run iteratively until all parameters and the complete log likelihood converge to steady values.

Examining (6) reveals that if the prediction error in y_i is so large that it dominates over the other denominator terms, then the weight $\langle w_i\rangle$ of that point will be very small. As this prediction error term in the denominator goes to ∞, $\langle w_i\rangle$ approaches 0. As can be seen in both (4) and (5), a data point with an extremely small weight will have a smaller contribution to the calculation of the regression estimate $\langle\beta\rangle$. This effect is equivalent to the detection and removal of an outlier if the weight of the data point $(\mathbf{x}_i, y_i)$ is small enough.

A few comments should be made regarding the initialization of the priors used in (4) to (7). First of all, the prior covariance of β, $\Sigma_{\beta,0}$, need only to be set to a large

[1] Note that $\langle\rangle$ denotes the expectation operator

enough value (e.g., $10^3\mathbf{I}$, where $\mathbf{I}$ is the identity matrix), which corresponds to an uninformative prior on β (i.e. the probability distribution is a relatively flat Gaussian). $\Sigma_{\beta,0}$ in (4) can be interpreted to be a stabilizing ridge-like value, similar to that of ridge regression, to ensure that the regression does not break down in the presence of collinear input data. Secondly, β_0 is usually initialized to zero, unless informative prior knowledge is available. As β_0 is multiplied by $\Sigma_{\beta,0}^{-1}$, it does not have any real influence on the update equations unless $\Sigma_{\beta,0}$ is chosen to be informative. Thirdly, the prior scale parameters $a_{w_i,0}$ and $b_{w_i,0}$ should be selected so that the weights $\langle w_i \rangle$ are 1 with some confidence. That is to say, we start by assuming that all points are inliers. For example, we can set $a_{w_i,0} = 1$ and $b_{w_i,0} = 1$ so that $\langle w_i \rangle$ has a prior mean of $a_{w_i,0}/b_{w_i,0} = 1$ with a variance of $a_{w_i,0}/b_{w_i,0}^2 = 1$. By using these values, the maximum value of $\langle w_i \rangle$ is capped at 1.5. This set of prior parameter values is generally valid for any application and/or data set and does not need to be modified, unless the user has good reason to insert strong biases towards particular parameter values, which we will not address in this paper.

The key insight towards this Bayesian treatment of weighted regression with heteroscedastic variance is that each data point will be assigned a posterior weight that is indicative of the amount of variance it has, relative to the average variance of the dataset. Consequently, a data point will be downweighted if its variance is much higher than that of the average variance. This algorithm does not require any tuning of threshold values or any user intervention beforehand, performing automatic outlier detection and removal in a black box-like way.

B. Incremental Version

The algorithm above is suitable if the data D is available in batch form. However, as in most robotic systems, data is often available from sensors one sample at a time, and filtering of the data needs to be done in a real-time, incremental (i.e. online) fashion. Hence, we take the Bayesian weighted model from (2) and modify it to make it an online algorithm. As typical in online algorithms, we introduce a forgetting rate to specify the window over which we wish to average data [18]. We use a scalar forgetting rate, λ, where $0 \leq \lambda \leq 1$, to exponentially discount data collected in the past. The forgetting rate enters the algorithm by accumulating the sufficient statistics of the batch algorithm in an incremental way. The sufficient statistics can be extracted by examining the EM update equations in (4) to (7). As the kth data point becomes available from the sensors, we can calculate the update equations for β and σ^2 as follows:

$$\boldsymbol{\Sigma}_{\beta_k} = \left(\boldsymbol{\Sigma}_{\beta,0}^{-1} + \frac{1}{\sigma^2} \mathrm{sum}_k^{wxx^T} \right)^{-1} \tag{8}$$

$$\langle \beta_k \rangle = \boldsymbol{\Sigma}_{\beta_k} \left(\boldsymbol{\Sigma}_{\beta,0}^{-1} \beta_0 + \frac{1}{\sigma^2} \mathrm{sum}_k^{wyx} \right) \tag{9}$$

$$\sigma_k^2 = \frac{1}{N_k} \Big[\mathrm{sum}_k^{wy^2} - 2\mathrm{sum}_k^{wyx} \langle \beta \rangle + \langle \beta \rangle^T \mathrm{sum}_k^{wxx^T} \langle \beta \rangle + \mathbf{1}^T \mathrm{diag} \left\{ \mathrm{sum}_k^{wxx^T} \boldsymbol{\Sigma}_\beta \right\} \Big] \tag{10}$$

where the sufficient statistics, exponentially discounted by λ, are:

$$\begin{aligned} N_k &= 1 + \lambda N_{k-1} \\ \mathrm{sum}_k^{wxx^T} &= \langle w_k \rangle \mathbf{x}_k \mathbf{x}_k^T + \lambda \mathrm{sum}_{k-1}^{wxx^T} \\ \mathrm{sum}_k^{wyx} &= \langle w_k \rangle y_k \mathbf{x}_k + \lambda \mathrm{sum}_{k-1}^{wyx} \\ \mathrm{sum}_k^{wy^2} &= \langle w_k \rangle y_k^2 + \lambda \mathrm{sum}_{k-1}^{wy^2} \end{aligned}$$

and all of $N_k, \mathrm{sum}_k^{wxx^T}, \mathrm{sum}_k^{wyx}, \mathrm{sum}_k^{wy^2}$ are 0 for $k = 0$. Notice that the calculation of the posterior covariances of β in (4) and (8) requires a matrix inversion, resulting in a computational complexity of $O(d^3)$. This will be fine for low-dimensional systems. However, for systems where the data has a large number of input dimensions, the matrix inversion becomes computationally prohibitive. In such situations, (8) can be re-written recursively, as in Recursive Least Squares [18] [19], in order to reduce the computational complexity to $O(d)$ per EM iteration. Given knowledge of the frequency of incoming data, the value of λ can be set accordingly, since the number of data samples that is not “forgotten” is $1/(1-\lambda)$. Additionally, the regression estimates come with a measure of confidence (the posterior covariance of β), such that the quality of the estimates and predictions can be judged.

Naturally, this incremental approximation of the batch Bayesian algorithm comes at a cost, since data points that initially appeared to be outliers may actually have been inliers (once we have collected enough data samples to realize this). If the forgetting rate λ used is small enough, then this effect will be less pronounced, since the window size of past data samples we are averaging over will be small as well. Hence, if this inlier falls outside the window of the past $1/(1-\lambda)$ data samples, the effect of mistaking an inlier as an outlier will be less pronounced. At the same time, λ should not be too small in order to ensure that the discrepancy in results between the incremental and batch versions is not too great. This trade-off between preserving equivalency with the batch version and discounting past events is a known issue with the use of forgetting factors for incremental algorithms.

III. RESULTS

We evaluate our algorithm's ability to automatically detect outliers on a synthetic dataset, before implementing it on a robotic quadruped dog, LittleDog, manufactured by Boston Dynamics Inc. (Cambridge, MA). We demonstrate the algorithm's performance by comparing it to four other standard techniques for outlier detection. These are described below:

- The first calculates the Mahalanobis distance for each data point from the expected value and based on an optimal hand-tuned threshold, classifies it as an outlier if it exceeds this threshold.

- The second is a simple mixture model that attempts to capture the dataset's structure with a two-component mixture model: a Gaussian distribution for inliers and and a uniform distribution for outliers.
- The third method is robust least squares regression. In particular, this version of robust least squares fits with bisquare weights [20], where the weight of each data point is a function of how far the data point is from the fitted line. Points close to the fitted line get full weight, while those further get smaller weight. The Matlab function `robustfit()` was used for implementation of robust least squares in experiments.
- The last technique is Faul & Tipping's variational Bayesian algorithm for robust regression, described in Section II-A. This iterative EM-like algorithm adopts a mixture model approach for explaining outliers and assumes outliers are generated by a uniform distribution. The EM update equations can be found in [11].

The thresholding approach collects the input and output data samples, treats them jointly and calculates the Mahalanobis distance for each data sample. It then selects a threshold such that a data sample with a Mahalanobis distance greater than the threshold value is classified as an outlier and excluded from the dataset. The optimal threshold value is found manually for a particular dataset. The final regression is done using the remaining samples in the data.

Similarly, the mixture model approach treats the input and output data jointly and attempts to cluster the data with a Gaussian distribution and a uniform distribution. For each data sample, two probabilities are inferred: the probability of the data sample belonging to the Gaussian distribution and the probability of the data sample belonging to the uniform distribution. If the probability of the data sample belonging to the Gaussian distribution exceeds some threshold, then the data sample is included in the dataset. Otherwise, it is an outlier and is removed.

The third approach, robust least squares regression with bisquare weights, uses an iteratively re-weighted least squares approach. First, it runs weighted least squares, before calculating the adjusted residuals and standardizing them. The weight estimates, $\mathbf{w}$, are a function of the standardized adjusted residuals, u. That is, if the standardized adjusted residual of the data sample i is u_i, the data sample's weight w_i is:

$$w_i = \begin{cases} \left(1-u_i^2\right)^2, & |u_i| < 1 \\ 0, & |u_i| \geq 1 \end{cases}$$

The algorithm is repeated until convergence.

A. Synthetic Dataset

First, we evaluated the algorithms on a linear regression problem, where the data is available in batch form. The synthetic dataset had 5 input dimensions, 1000 data points and additive Gaussian noise with a signal-to-noise ratio (SNR) of 10. Each data point had a 20% probability—drawn from a uniform distribution—of being an outlier. Outliers were created such that they were at least some distance $k\sigma$ from the true mean of the outputs, where σ is the standard deviation of the true conditional mean of the outputs and k is a scaling factor (e.g. $k = 1, 2, 3, ...$). Since approximately 95% of data values in a Gaussian distribution lie within $\pm 2\sigma$ from the mean, this ensured that the outlying data points were far away enough to be classified as outliers. Fig. 1 plots the outputs of a representative dataset with only one input dimension in order to visualize the data samples in a 2D plot. The true outputs are denoted in solid squares, noisy outputs are marked with circles, and outliers are denoted by solid stars. Notice that the outliers were generated to be $+3\sigma$ away from the true output mean—avoiding a scenario where outliers below the data cloud could potentially diminish the effects of overestimation.

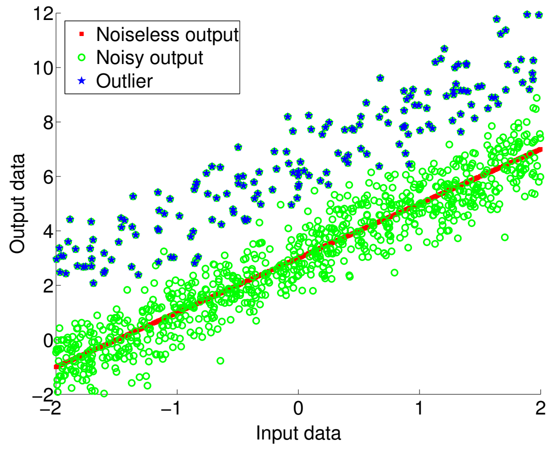


Fig. 1. A sample dataset showing data points from a linear function with 1-dimensional inputs. True noiseless output data, Y, is in squares; noisy outputs are marked with circles; and outliers are denoted by solid stars. (Outliers are at least 3σ from the true conditional output mean, SNR of the output data is 10.)

We ran all algorithms on a 5 dimensional training dataset. We constructed a test dataset in a similar way as the training set, except no noise or outliers were present in the test outputs. Then, we calculated the predictions of each algorithm on the test dataset, using the regression estimate inferred from training on the noisy dataset. Table I shows the normalized mean squared prediction error of the noiseless test dataset for all algorithms, averaged over 10 trials and as a function of how far the outliers were from the inliers. The results show that Bayesian weighted linear regression achieves the lowest average normalized mean squared error (NMSE). Thresholding appears to work quite well when the threshold value is hand-tuned optimally, while mixture models, robust least squares and Bayesian robust regression seem to be less robust to outliers. Notice that the error values for the mixture model and Bayesian robust regression are quite similar. This may be explained by the fact that both methods rely on a uniform distribution to capture outliers. It is clear that learning the weights, instead of modeling them with a heuristic function, is the more preferable and powerful approach to outlier detection. Unsurprisingly, the error values are lower when outliers are closer to inliers. In this scenario, failure to detect an outlier has a less adverse effect on the performance of the algorithms, since outliers may be indistinguishable from noisy inliers.

TABLE I

AVERAGE NORMALIZED MEAN SQUARED ERROR (NMSE) ON A LINEAR FUNCTION WITH 5 INPUT DIMENSIONS, EVALUATED IN BATCH FORM OVER 10 TRIALS FOR THRESHOLDING USING MAHALANOBIS DISTANCE, A MIXTURE MODEL, ROBUST LEAST SQUARES, FAUL & TIPPING'S BAYESIAN ROBUST REGRESSION AND BAYESIAN WEIGHTED REGRESSION: σ IS THE STANDARD DEVIATION OF THE TRUE CONDITIONAL OUTPUT MEAN AND SNR OF THE OUTPUTS IS 10.

Algorithm	Distance of outliers from mean		
	$+3\sigma$	$+2\sigma$	$+\sigma$
Thresholding	0.0903	0.0503	0.0232
Mixture Model	0.1327	0.0688	0.0286
Robust Least Squares	0.1890	0.1518	0.0880
Robust Regression	0.1320	0.0683	0.0282
Bayesian weighted regression	0.0273	0.0270	0.0210

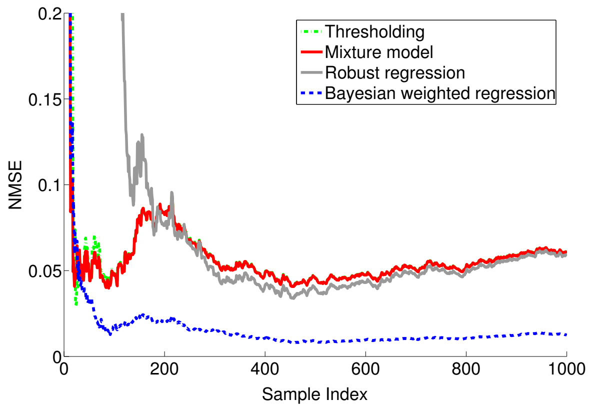
In the case where the data contains outliers that lie $\pm\sigma$, $\pm 2\sigma$ or $\pm 3\sigma$ from the true output mean, the average NMSE values will be reduced due to the effects of cancellation by outliers above and below the data cloud. Regardless, the trend of performance among the algorithms remains unchanged, with Bayesian weighted linear regression as the most competitive.

Next, we evaluated the algorithms on the same synthetic training datasets used in the first experiment, but in real-time, making data samples available sequentially one at a time and using a forgetting rate of $\lambda = 0.999$. All algorithms were made to be incremental through the use of forgetting rates. The robust least squares algorithm was omitted in this comparison since it is a batch algorithm, and making it recursive is non-trivial. Fig. 2(a) and 2(b) track the error in the predicted outputs on the training data for two types of datasets: the first has outliers that lie at least 3σ from the true output mean, while the second has outliers that lie at least 2σ from the true output mean. As Fig. 2 illustrates, the Bayesian weighted algorithm, shown in the dark dotted line, reduces the error to a value that is lowest, compared to the other algorithms. Thus, this matches the pattern shown in Table I and confirms that even in a real-time, incremental setting, the Bayesian weighted regression algorithm outperforms the other methods.

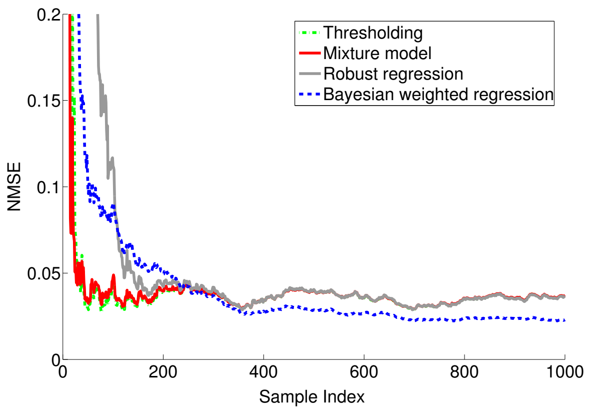
B. LittleDog Robot

We evaluated the algorithms on a 12 degree-of-freedom (DOF) quadruped robotic dog, LittleDog, as shown in Fig 3. The robot dog has two sources that measure its orientation: the motion capture (MOCAP) system and an on-board inertia measurement unit (IMU). Both provide a quaternion q of the robot's orientation: q_{MOCAP} from the MOCAP and q_{IMU} from the IMU. q_{IMU} drifts over time, since the IMU cannot provide stable orientation estimation but its signal is clean. q_{MOCAP} has outliers and noise, but no drift. We would like to estimate the offset between q_{MOCAP} and q_{IMU}, and this offset is a noisy, outlier-infested, slowly drifting signal. The drift that occurs in the IMU is quite common in systems where sensors collect data that need to be integrated. For example, given angular acceleration from a sensor, we may want to know what the angular velocity is, and we can calculate this by integrating the angular acceleration. Unfortunately, sensor data may contain bias, and in such a case, this will translate to an error in the angular velocity that will be propagated and amplified at each step that an integration operation is performed. Thus, the resulting angular velocity will have a drifting bias.

Fig. 3. Quadruped robotic dog (Boston Dynamics)



(a) Outliers are at least 3σ from true output mean



(b) Outliers are at least 2σ from true output mean

Fig. 2. Normalized mean squared error (NMSE) values for a linear function with 5 input dimensions (the same synthetic batch datasets used in Table I) evaluated in an incremental manner for 1000 data samples: $\lambda = 0.999$, SNR of output data is 10.

Fig. 4 shows the offset data between q_{MOCAP} and q_{IMU} for one of the four quaternion coefficients, collected over 6000 data samples. It plots the predicted outputs of the four incremental algorithms, along with the predicted outputs of the batch version of Bayesian weighted regression. Fig. 5 displays a magnified version of the results. The thresholding, mixture model and variational Bayesian robust regression approaches appear to be somewhat sensitive to outliers (oc-

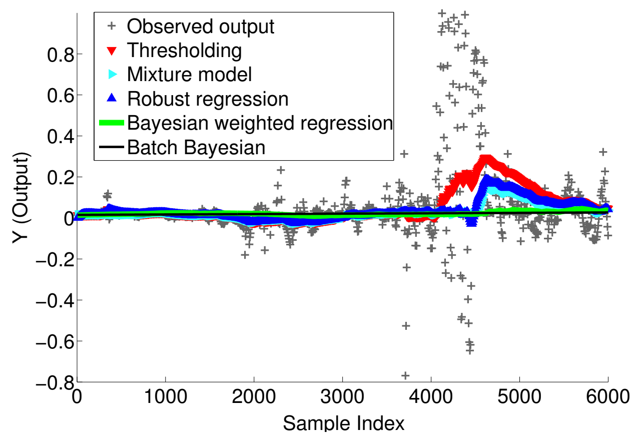


Fig. 4. Predicted versus observed outputs on the offset between the quaternion from the IMU and the quaternion from the MOCAP, shown for one of the four quaternion coefficients ($\lambda = 0.999$). Observed outputs are noisy and contain outliers.

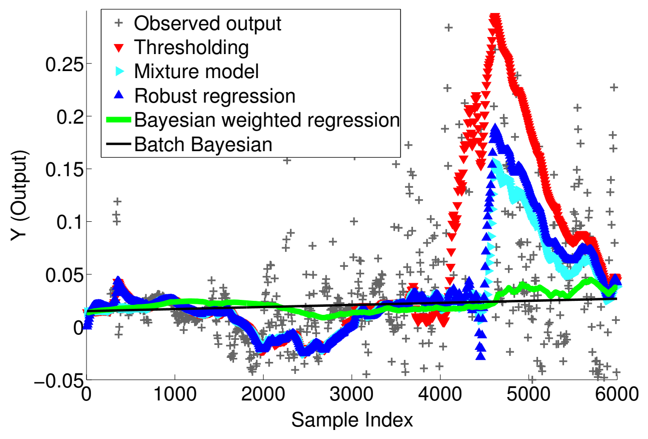


Fig. 5. Magnified view of Fig 4: Predicted versus observed outputs on the offset between the quaternion from the IMU and the quaternion from the MOCAP, shown for one of the four quaternion coefficients ($\lambda = 0.999$). Observed outputs are noisy and contain outliers.

curring between the 4000th and 5000th sample). In comparison, the incremental version of Bayesian weighted regression is far less sensitive to outliers. Given it is an incremental approximation of the batch version, the algorithm's predicted outputs track those of the corresponding batch version quite closely.

IV. CONCLUSION

We have introduced a Bayesian weighted regression algorithm that is able to automatically detect and eliminate outliers in real-time, without requiring any interference from the user, parameter tuning, sampling or model assumptions about the underlying data structure. We compared this algorithm to standard approaches for outlier detection, such as thresholding using Mahalanobis distance, mixture models, robust least squares with bisquare weights and an alternate variational Bayesian approach to robust regression. We evaluated all algorithms on synthetic and robotic data, demonstrating the effectiveness of the Bayesian weighted regression algorithm in performing real-time outlier detection. It is able to achieve a level of performance on par with and even exceeding that of standard approaches, providing a robust and competitive alternative to filtering sensor data.

V. ACKNOWLEDGMENTS

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, a NASA grant AC#98 − 516, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, and the ATR Computational Neuroscience Laboratories.

REFERENCES

[1] D. S. Moore and G. P. Mccabe. *Introduction to the Practice of Statistics Ise*. W.H. Freeman & Company, March 1999.

[2] N. Abe, B. Zadrozny, and J. Langford. Outlier detection by active learning. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 767–772, New York, NY, USA, 2006. ACM Press.

[3] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *In Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[4] M. Breitenbach and G. Z. Grudic. Clustering through ranking on manifolds. In *Proceedings of the 22nd international conference on Machine learning*, pages 73–80, New York, NY, USA, 2005. ACM Press.

[5] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *In Advances in Neural Information Processing Systems 14: Proceedings of the 2001.*, 2001.

[6] M. Aitkin and G. T. Wilson. Mixture models, outliers and the em algorithm. *Technometrics*, 22:325–331, 1980.

[7] D. W. Scott. Outlier detection and clustering by partial mixture modeling. In *COMPSTAT 2004 Symposium*, pages 453–465, Heidelberg, 2005. Physica-Verlag.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[9] D. Fox, W. Burgard, D. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI/IAAI*, pages 343–349, 1999.

[10] K Konolige. Robot motion: Probabilistic model; sampling and gaussian implementations; markov localization. Technical report, SRI International, 2001.

[11] Anita C. Faul and Michael E. Tipping. A variational approach to robust regression. In *International Conference on Artificial Neural Networks*, pages 95–102, 2001.

[12] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *AI Review*, 11:11–73, April 1997.

[13] A. Gelman, J. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 2000.

[14] T. P. Ryan. *Modern Regression Methods*. Wiley, 1997.

[15] R.M. Neal. *Bayesian learning for neural networks*. PhD thesis, Dept. of Computer Science, University of Toronto, 1994.

[16] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society. Series B*, 39(1):1–38, 1977.

[17] Z. Ghahramani and M.J. Beal. Graphical models and variational methods. In D. Saad and M. Opper, editors, *Advanced Mean Field Methods - Theory and Practice*. MIT Press, 2000.

[18] L. Ljung and T. Sonderstrom. *Theory and Practice of Recursive System Identification*. MIT Press, 1983.

[19] G. J. Bierman. Factorization methods for discrete sequential estimation. *Mathematics in Science and Engineering*, 128, 1977.

[20] D. C. Hoaglin. Letter values: A set of selected order statistics. In D. C. Hoaglin, F. Mosteller, and J. W. Tukey, editors, *Understanding Robust and Exploratory Data Analysis*, pages 33–57. Wiley, 1983.