Filtering Outliers in Bayesian Optimization

Ruben Martinez-Cantin[1,2], **Kevin Tee**[1], **Michael McCourt**[1], **Katharina Eggensperger**[1,3]
SigOpt[1], Centro Universitario de la Defensa, Zaragoza[2], Albert-Ludwigs-Universität Freiburg[3]
{ruben, kevin, mccourt, katharina}@sigopt.com

Abstract

We propose a Bayesian optimization algorithm to autonomously manage situations where outlier data may be encountered. These outliers can pollute the surrogate model, reducing the efficiency of the optimization. Our algorithm uses robust regression (a Gaussian process with Student-t likelihood) to classify data points as outliers and diminish their effect by excluding them from consideration. We present an empirical evaluation of optimization methods showing that our method is able to overcome the presence of outliers. Furthermore, we also show that our method compares favorably to Bayesian optimization with robust surrogate models on optimization benchmarks and machine learning hyperparameter tuning problems.

1 Introduction

In recent years, Bayesian optimization [20] has emerged as the *de facto* method for problems which demand sample efficiency, i.e., each sample or trial requires a great investment of time or resources. Bayesian optimization (BO) provides a derivative-free strategy for executing this black-box optimization through the use of a probabilistic surrogate model which describes the relationship between suggested parameter configurations and their associated value or performance. The strength of this model comes from its comprehensive "memory" of the progress of the optimization [12], allowing Bayesian optimization to often outperform other black-box optimization methods by effectively utilizing all the information from previously observed results.

In the presence of faulty or outlier data, this memory can actually cause problems and slow (or even prevent) convergence because outliers are never forgotten. Other methods, such as gradient descent or evolutionary algorithms, have an effectively short memory making them naturally resilient to outlier data. For Bayesian optimization methods to manage outliers, steps must be taken to either alter the construction of the surrogate model or its interpretation.

Outlier management and detection is an intensive area of research in many disciplines because of the prevalence of outliers in practical circumstances. Outliers are often problem dependent, and are therefore defined and interpreted differently across different applications. In the context of hyperparameter tuning and design of computer experiments, outliers might appear from random bugs, I/O or networking errors, convergence issues for certain sets of parameters, etc. In the case of physical experiments, human implementation mistakes (e.g., equipment calibration) or external factors (e.g., unexpected seismic activity) could unpredictably impact experimental results.

The methods employed to deal with outliers can be classified in two areas: robustness of inference to outliers and outlier diagnostics [17]. Robust inference strategies consist of developing models which are often more computationally expensive and less accurate but can incorporate outliers without allowing them to dominate non-outlier data. Outlier diagnostics methods generally consist of preprocessing data through statistical analysis to classify outliers and exclude them from a subsequent model built with standard (non-robust) methods.

In this article, we propose a strategy for managing outliers during Bayesian optimization using ideas developed in the regression community. Periodically, during the optimization, we use a Gaussian Process (GP) with the Student-t likelihood to diagnose outliers. All previously observed results are

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

then classified as acceptable or outliers and only the acceptable data is analyzed through the standard Bayesian optimization process. This process is depicted in Figure 1.

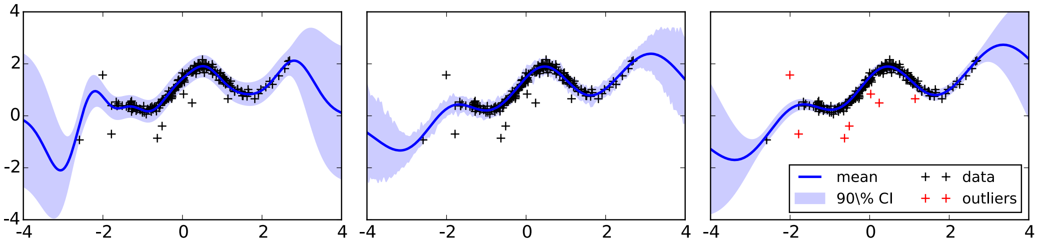

Figure 1: A depiction of the filtering process. *Left*: The basic GP surrogate model with biased estimates and high variance. *Center*: The Student-t likelihood allowing for identification of outliers. *Right*: The GP model fit after excluding the outliers.

Our experimental results show that our two-step method for outlier classification is sufficient for enabling Bayesian optimization in the presence of outliers. Furthermore, our results also show that this method is preferable to simply using a robust regression model as was previously suggested (e.g., Shah et al. [19]). To the authors' knowledge, this is the first work on Bayesian optimization with experimental results addressing the presence of outliers.

2 Bayesian Optimization with Outliers

To motivate our strategy for conducting Bayesian optimization in the presence of outliers, we turn to the field of robust regression, i.e., designing surrogate models that are robust to outliers. The simplest GP surrogate model (described in Appendix A) assumes that observed function values take the form $Y_{\mathbf{x}} = f(\mathbf{x}) + \epsilon$ where f is the function to be minimized and $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ for some fixed $\sigma_n > 0$. Outliers can corrupt such surrogate models because they do not follow this assumed distribution. In particular, the Gaussian assumption for ϵ cannot properly fit the outliers without introducing bias in the function estimate. It has been proven that predictions from the model will be increasingly biased proportional to the magnitude of deviation in an outlier [14].

Choosing a heavy-tailed distribution for the observation noise ϵ, such as the Laplace, the hyperbolic secant, or the Student-t likelihoods, can help produce models which are robust to outliers. Natural robust replacements for the standard GP model include: the use of the Student-t distribution to model the observation noise ϵ [27]; or the use of a multivariate t-distribution to jointly model observed data (rather than the multivariate normal distribution used in a GP) [19]. More information is provided in Appendix B. In practice, we found these strategies to be computationally demanding and numerically unstable when computing expected improvement (EI) [11] which led us to develop our method (encapsulated in Algorithm 1).

Algorithm 1 BO with outliers **Input:** Total budget T, rejection threshold α

```
Initial design of $p$ points (e.g.: LHS):   $\mathbf{X} \leftarrow \mathbf{x}_{1:p}$   $\mathbf{y} \leftarrow y_{1:p}$
for $t = p+1 \dots T$ do
    if schedule($t$) then
        $\mathbf{\Theta_t} \leftarrow$ fitGPwithTlik($\mathbf{X}, \mathbf{y}$)
        $\mathbf{X}_{in}, \mathbf{y}_{in} \leftarrow$ filterOutliers($\mathbf{X}, \mathbf{y}, \mathbf{\Theta_t}, \alpha$)
    if length($\mathbf{y}_{in}$)/length($\mathbf{y}$) $< 0.5$ or not schedule($t$) then   $\mathbf{X}_{in} \leftarrow \mathbf{X}$,   $\mathbf{y}_{in} \leftarrow \mathbf{y}$
    $\mathbf{\Theta_g} \leftarrow$ fitGPwithGlik($\mathbf{X}_{in}, \mathbf{y}_{in}$)
    $\mathbf{x}_t = \arg\max_{\mathbf{x}}$ EI($\mathbf{x} | \mathbf{X}_{in}, \mathbf{y}_{in}, \mathbf{\Theta_g}$)
    $y_t \leftarrow f(\mathbf{x}_t)$   $\mathbf{X} \leftarrow add(\mathbf{x}_t)$   $\mathbf{y} \leftarrow add(y_t)$
```

We still fit a robust model with the Student-t likelihood, represented in the algorithm as `fitGPwithTlik`. Then, we use this robust model to *identify and remove* outliers with the function `filterOutliers`. To accomplish this, we define an α term such that points with values less likely than α (as determined by the robust model) are classified as outliers. Periodically, we conduct this filtering on *all* the observed data and exclude observed results which are sufficiently unlikely. After the filtering excludes outliers from consideration, the remaining acceptable data is passed through the standard Bayesian optimization process (described in Appendix A) to generate a next suggested $\mathbf{x}_t$.

The `schedule` function manages the frequency with which we reclassify outliers. In our experimentation, we found that a fairly aggressive (high) $\alpha = 0.05$ threshold worked well because we reclassify all observed results as either outliers or acceptable. Thus, no observation is permanently rejected. We also monitor the ratio of acceptable points $\texttt{length}(\mathbf{y}_{in})$ / $\texttt{length}(\mathbf{y})$; if more than half of the points are excluded in `filterOutliers`, we do not filter and try again in a subsequent iteration.

Finally, for the `schedule`, we have used an initial delay of 10 iterations before trying to classify any outliers and the filtering was performed every 2 iterations. These decisions must be made by the user and will impact the balance between computation cost of the optimization process (because of the cost of `fitGPwithTlik`) and the necessary budget T (because of the stagnation observed when outliers are not filtered). More detail is provided in Appendix C. Our proposed method is more efficient than the natural robust replacements described above because the robust model is fit less frequently (according to the `schedule`) and because the standard GP model (whose posterior has a closed form) is used for the EI maximization.

3 Results

We evaluated our method on a set of numerical benchmarks and realistic hyperparameter optimization problems involving outliers. In all experiments, we have compared *our method* (Algorithm 1) to a *Baseline*, standard Bayesian optimization with a Gaussian ϵ distribution (described in more detail in Appendix A). The baseline is equivalent to Algorithm 1 run with $\texttt{schedule}(t) \equiv \texttt{False}$, i.e., never test for outliers. We also compare to standard Bayesian optimization applied with *No outliers*. The percentile value $\alpha = .05$, defined in Section 2, was used for filtering outliers in our method. Our covariance kernel is the C^4 Matérn.

3.1 Numerical benchmarks

To construct numerical benchmarks we used the methodology from Henning and Schuler [5] to generate a set of 8D random functions from two types of Gaussian processes. We tested samples from a GP with the same C^4 Matérn kernel as used in the surrogate model, as well as from a GP with a rational quadratic kernel. Outliers were i.i.d. sampled from a uniform distribution $y_{\text{outlier}} \sim \mathcal{U}(1, 2)$. Results are shown in Figure 2. We also compared to the natural robust replacements described in Section 2: the *BO with t-likelihood* and the *BO with t-process* method.

Our method outperforms both robust regression methods and it is able to reach performance comparable to the *No outliers* scenarios. We also see that failing to consider the outliers has a devastating effect, resulting in the optimization stagnating for the *Baseline*. Between the two robust regression methods, there appears to be no clear winner.

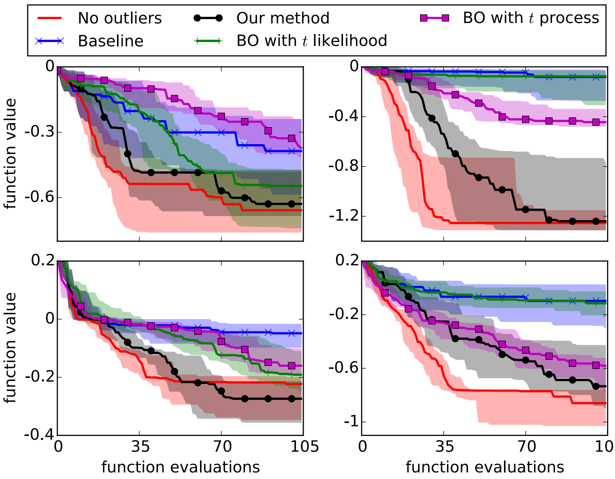

Figure 2: The median and interquartile range of 20 trials is plotted with each BO method on 8D random functions. *left*: 10% outliers. *right*: 20% outliers. *top*: Matérn generated. *bottom*: rational quadratic generated.

3.2 Hyperparameter tuning

Besides the previously mentioned errors and failures that might appear, we have found that hyperparameter tuning problems have some intrinsic sources of outliers. For example, early stopping during training might be used to guard against overfitting and reduce computational cost by reallocating resources based on the performance at early stages [8]. However, random initialization of variables resulting in very different behaviors at early stages [24]. Therefore, early stopping may result in some configurations having an outlier value (based on the random initialization rather than the expected hyperparameter performance).

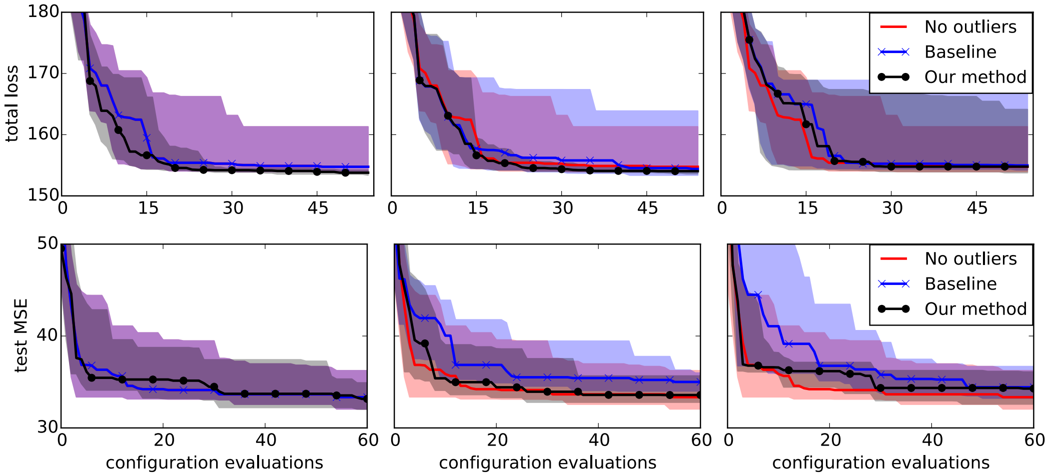

Figure 3: Median and interquartile range of each method. *top*: Optimization of the variational autoencoder on the MNIST dataset. *bottom*: Optimization of the feed-forward neural network on the Boston housing dataset. *left*: No outliers (thus the "Baseline" coincides with the "No outliers case"). *center*: 10% outliers. *right*: 20% outliers.

Variational autoencoder (VAE) In this experiment we train a VAE (a generative method based on neural networks) on the MNIST dataset [4]. We tune the number of nodes in the hidden layer and learning rate, learning rate decay, and ϵ constant for the Adam optimizer. We simulated outliers as I/O failures where the VAE is trained only on a subset of the data (randomly generated between 100 and 1000 images). The results are shown in Figure 3. In the case without externally induced outliers and a small level of simulated outliers (10%), the upper quantile shows our method is more robust than the baseline, reinforcing the theory that there are already outliers present. For a high level of outliers (20%) the performance drops, suggesting that the number of simulated outliers in addition with the existing intrinsic outliers reaches a point near the limit of robustness.

Feedforward network We apply gradient descent to train a single layer feedforward neural network on the Boston housing dataset (inspired by Wang et al. [28]). The 4 hyperparameters we tuned were the number of nodes in the hidden layer, learning rate, learning rate decay, and ρ, the parameter that controls the exponential decay rate from RMSprop. Figure 3 shows the results of the optimization for 10% outliers. An outlier in this optimization consisted of running the neural network 5 epochs, rather than the standard 20 epochs. In the case with no simulated outliers, the performance is no better than the baseline method. When simulated outliers are added, the behavior is similar to the VAE, consistently providing a clear benefit in comparison to the baseline.

4 Conclusions

We have presented a method to extend Bayesian optimization in the presence of outliers. The method combines robust regression with a Student-t likelihood on a GP, and outlier analysis to classify inlier and outlier data points. We have extensively evaluated the proposed method on both synthetic benchmarks and realistic applications showing that our method is suitable for practical Bayesian optimization in the presence of outliers. Furthermore, we have shown how our method does not incur negative outcomes in situations without induced outliers. This highlights the potential value of this approach even in supposedly controlled environments and lab conditions. Finally, we have experimentally demonstrated that Bayesian optimization with a robust surrogate model designed to accommodate outliers produces inferior results.

References

[1] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, pages 115–123, 2013.

[2] Josef Dick and Friedrich Pillichshammer. *Digital Nets and Sequences: Discrepancy Theory and Quasi–Monte Carlo Integration*. Cambridge University Press, 2010.

[3] Gregory Fasshauer and Michael McCourt. *Kernel-based Approximation Methods using Matlab*, volume 19. World Scientific Publishing Co Inc, 2015.

[4] François Chollet. Variational autoencoder with Keras. `https://github.com/fchollet/keras/blob/master/examples/variational_autoencoder.py`, 2017.

[5] Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

[6] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of Learning and Intelligent Optimization*, pages 507–523, 2011.

[7] Pasi Jylänki, Jarno Vanhatalo, and Aki Vehtari. Robust Gaussian process regression with a Student-t likelihood. *Journal of Machine Learning Research*, 12(Nov):3227–3257, 2011.

[8] Aaron Klein, Stefan Falkner, Jost T. Springenberg, and Frank Hutter. Learning curve prediction with Bayesian neural networks. In *International Conference on Learning Representations*, 2017.

[9] Kenneth L. Lange, Roderick J. A. Little, and Jeremy M. G. Taylor. Robust statistical modeling using the t distribution. *Journal of the American Statistical Association*, 84(408):881–896, 1989.

[10] Ruben Martinez-Cantin. Bayesopt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, 15(1):3735–3739, 2014.

[11] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. In *Towards Global Optimisation*, volume 2, pages 117–129. Elsevier, 1978.

[12] Andrew W. Moore and Jeff Schneider. Memory-based stochastic optimization. In *Advances in Neural Information Processing Systems 9*, volume 8, pages 1066–1072, 1996.

[13] Radford M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report 9702, University of Toronto, Department of Statistics, 1997.

[14] Anthony O'Hagan. On outlier rejection phenomena in Bayes inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 358–367, 1979.

[15] Anthony O'Hagan. Some Bayesian numerical analysis. *Bayesian Statistics*, 4:345–363, 1992.

[16] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[17] Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection*, volume 589. John Wiley & Sons, 2005.

[18] Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments*. Springer-Verlag, 2003.

[19] Amar Shah, Andrew Gordon Wilson, and Zoubin Ghahramani. Student-t processes as alternatives to Gaussian processes. In *Artificial Intelligence and Statistics*, pages 877–885, 2014.

[20] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[21] Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2960–2968, 2012.

[22] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pages 2171–2180, 2015.

[23] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems 29*, pages 4134–4142, 2016.

[24] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

[25] Qingtao Tang, Li Niu, Yisen Wang, Tao Dai, Wangpeng An, Jianfei Cai, and Shu-Tao Xia. Student-t process regression with Student-t likelihood. In *International Joint Conference on Artificial Intelligence*, pages 2822–2828, 2017.

[26] Qingtao Tang, Yisen Wang, and Shu-Tao Xia. Student-t process regression with dependent Student-t noise. In *European Conference on Artificial Intelligence*, 2016.

[27] Jarno Vanhatalo, Pasi Jylänki, and Aki Vehtari. Gaussian process regression with student-t likelihood. In *Advances in Neural Information Processing Systems 22*, pages 1910–1918, 2009.

[28] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635, 2017.

[29] Brian J. Williams, Thomas J. Santner, and William I. Notz. Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, 10(4):1133–1152, 2000.