Practical Bayesian optimization in the presence of outliers

Ruben Martinez-Cantin
SigOpt
Centro Universitario de la Defensa, Zaragoza

Kevin Tee
SigOpt

Michael McCourt
SigOpt

Abstract

Inference in the presence of outliers is an important field of research as outliers are ubiquitous and may arise across a variety of problems and domains. Bayesian optimization is method that heavily relies on probabilistic inference. This allows outstanding sample efficiency because the probabilistic machinery provides a *memory* of the whole optimization process. However, that virtue becomes a disadvantage when the memory is populated with outliers, inducing bias in the estimation. In this paper, we present an empirical evaluation of Bayesian optimization methods in the presence of outliers. The empirical evidence shows that Bayesian optimization with robust regression often produces suboptimal results. We then propose a new algorithm which combines robust regression (a Gaussian process with Student-t likelihood) with outlier diagnostics to classify data points as outliers or inliers. By using an scheduler for the classification of outliers, our method is more efficient and has better convergence over the standard robust regression. Furthermore, we show that even in controlled situations with no expected outliers, our method is able to produce better results.

1 INTRODUCTION

Sample efficient optimization plays an important role in many aspects of science and engineering, where each sample or trial might represent a large cost in time, energy or resources. In recent years, Bayesian optimization has emerged as the *de facto* method for these kind of problems; it provides a black-box solution of the global optimization problem without the need for gradient information [28]. The underlying mechanism which makes Bayesian optimization methods so powerful is the use of a probabilistic surrogate model, which incorporates all of the data which has been observed over the course of the optimization. This model provides a comprehensive "memory" of the progress of the optimization which empowers Bayesian optimization to often outperform other black-box optimization methods [19].

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 so that they do not hamper the construction of the surrogate model.

Outlier management and detection is an intensive area of research in many disciplines because of the importance of outliers in practice. Outliers are often problem dependent, and are therefore defined differently across different applications. In the context of hyperparameter tuning and design of computer experiments, outliers and gross errors might appear from random bugs, I/O or networking errors, convergence issues for certain sets of parameters, etc. In the case of physical experiments, a user typically has to calibrate the experiment according to the suggested set of parameters and then report the performance, which might result in human mistakes while translating the numbers. Furthermore, in the case of real experiments, external factors might randomly influence and modify certain results. The methods employed to deal with outliers can be classified in two areas: robustness to outliers of inferences and outlier diagnostics [25].

Robust inference strategies consist of developing models that can incorporate outliers without allowing them to dominate non-outlier data. In the Bayesian framework, we can reduce the influence of the outliers

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

by replacing the *non-robust* population model (e.g., Gaussian) by a longer-tailed distribution, which allows a greater possibility of extreme observations (e.g., Student-t) [7]. In the Bayesian optimization context, the surrogate model is typically a Gaussian process (GP) [24], which is a regression model from inputs $\mathbf{x}$ to targets $y = f(\mathbf{x})$. If we consider that the outliers appear in the target variable y, we will need to incorporate a long tailed observation model, like a Student-t likelihood. Robust methods are usually more computationally expensive and provide lower accuracy because of the fact that they need to accommodate the long-tailed data. Section 2 analyzes in more detail the literature from robust regression, especially as applied to GP regression. Note that errors in the input variables $\mathbf{x}$ are addressed by *sensitivity analysis* [7], which has been already studied in Bayesian optimization [21].

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. Standard methods such as those found in statistical software are known to be problematic and limited to simpler models [25]. More advanced techniques can be based on statistical learning of outliers. Supervised learning of outliers requires knowledge of labeled outliers in the same dataset [9]. Unsupervised learning requires a good *a priori* model able to represent the data. Details of our diagnostics methodology will be discussed in Section 3.

Methods such as RANSAC, which is popular in computer vision and robotics, combine both strategies: first, a robust model is built accommodating all points, which is then employed for classifying into outliers and inliers. Once the outliers are identified, a non-robust model is used with the inlier data points because it has better statistical properties. Inspired by this methodology, our contribution also includes a two step algorithm that combines robust regression and outlier classification.

1.1 Bayesian optimization

Bayesian optimization methods [15, 28] try to minimize, over some domain $\mathcal{X} \subset \mathbb{R}^d$, a function $f : \mathcal{X} \to \mathbb{R}$ while sampling f as little as possible (this is what is meant by "sample efficient"). The optimization is generally initialized with p evaluations by sampling with low discrepancy sequences [4] such as latin hypercube sampling.

After the initialization, the sequential component of the optimization begins. At iteration t, all previously observed data $\mathbf{y} = \mathbf{y}_{1:t}$ at points $\mathbf{X} = \mathbf{X}_{1:t}$ is used to construct a probabilistic surrogate model $s_{\mathbf{y},\mathbf{X}}$. The next location $\mathbf{x}_{t+1}$ is determined by optimizing a chosen *acquisition function* which measures the benefit or utility associated with evaluating a proposed $\mathbf{x} \in \mathcal{X}$. In this article, we restrict our focus to only considering expected improvement [18],

$$EI(\mathbf{x}) = \mathbb{E}_{p(y|s_{\mathbf{y},\mathbf{X}}(\mathbf{x}))} \left[\max(0, y^* - y)\right], \qquad (1)$$

where $y^* = \max(y_1, \dots, y_t)$.

1.2 Gaussian processes for surrogate modeling

In Section 1.1 we explained the need for the probabilistic surrogate model $s_{\mathbf{y},\mathbf{X}}$ to drive the optimization but not the means by which it is constructed. Most frequently, this takes the form of a GP, although other alternatives have been presented, such as random forests [10], kernel density estimators [1] or Bayesian neural networks [30, 31]. For the remainder of the paper we only consider a GP with zero mean and covariance $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as the surrogate model.

Observation model We choose to build our GP models on the belief that data has been observed in the presence of homoscedastic noise $y = f(\mathbf{x}) + \epsilon$, with *Gaussian likelihood* $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, resulting in a GP posterior model. We can also rewrite the likelihood as $y|f \sim \mathcal{N}(f, \sigma_n^2)$, where $f \equiv f(\mathbf{x})$. However, as we will see in Section 2, this model is not robust to outliers and we will replace the Gaussian likelihood for a more suitable distribution, that is, the Student-t.

In this setting, after t observations (as explained in Section 1.1), the GP posterior model gives predictions at a query point $\mathbf{x}_q$ which are normally distributed $y_q \sim \mathcal{N}(\mu(\mathbf{x}_q), \sigma^2(\mathbf{x}_q))$, such that

$$\begin{aligned} \mu(\mathbf{x}_q) &= \mathbf{k}(\mathbf{x}_q, \mathbf{X})^T \mathbf{K}^{-1} \mathbf{y}, \\ \sigma^2(\mathbf{x}_q) &= k(\mathbf{x}_q, \mathbf{x}_q) - \mathbf{k}(\mathbf{x}_q, \mathbf{X})^T \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}_q, \mathbf{X}), \end{aligned} \qquad (2)$$

where

$$\begin{aligned} \mathbf{k}(\mathbf{x}_q, \mathbf{X}) &= \begin{pmatrix} k(\mathbf{x}_q, \mathbf{x}_1) & \dots & k(\mathbf{x}_q, \mathbf{x}_t) \end{pmatrix}^T, \\ \mathbf{K} &= \begin{pmatrix} \mathbf{k}(\mathbf{x}_1, \mathbf{X}) & \dots & \mathbf{k}(\mathbf{x}_t, \mathbf{X}) \end{pmatrix} + \mathbf{I}\sigma_n^2. \end{aligned} \qquad (3)$$

The kernel is chosen to be the Matérn kernel with $\nu = 5/2$, also called C^4 Matérn kernel [5],

$$k(\mathbf{x}, \mathbf{x}') = \left(1 + r + r^2/3\right) \mathrm{e}^{-r}, \qquad (4)$$

where $r = \|\mathbf{x} - \mathbf{x}'\|_\Lambda$ for some positive definite matrix Λ. The automatic relevance determination kernel which we use here restricts Λ to being diagonal. The hyperparameters of Λ are estimated by maximum likelihood, although MCMC could be used instead [29].

Contribution In this article, we propose a strategy for managing outliers during Bayesian optimization using ideas developed in the regression community. At certain steps during the optimization, we use a GP with the Student-t likelihood to perform an outlier diagnostic. All previously observed results are then classified as acceptable or outliers and only the acceptable data is analyzed through the standard Bayesian optimization process. Our experimental results show that our two-step method for outlier data 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 in [27], by accommodating the outliers in a robust Bayesian optimization engine. To the authors' knowledge, this is the first work on Bayesian optimization with experimental results addressing the presence of outliers.

2 ROBUST REGRESSION FOR GAUSSIAN PROCESSES

Standard GP-based Bayesian optimization uses an observation model for noisy data with a Gaussian likelihood, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ as defined in (3); this allows for closed form inference but, as shown in Figure 1, is very sensitive to outliers. In this section, we review literature on robust regression [25] to draw inspiration on how to alter the standard GP used in Bayesian optimization to create a version robust to outliers.

The key change in creating such a regression model is using a large tail distribution as the observation likelihood in lieu of the standard Gaussian likelihood; possible options include the Laplace, the hyperbolic secant, or the Student-t likelihoods. All those probability distributions are robust to the presence of outliers, with the Student-t likelihood usually providing the best results [11, 14]. O'Hagan proved that the Student-t distribution can reject up to m outliers tending to infinity (or negative infinity) provided that there are at least $2m$ observations at all. At the same time, he also showed in [22] that the Gaussian distribution is *non-robust*, meaning that if an outlier is not rejected, the larger the error present in the outlier, the larger the estimate bias will be.

However, the Student-t likelihood, as well as the alternative distributions mentioned, do not allow closed form inference of the posterior. Therefore, we need to find an approximation that will provide a suitable posterior in the form of a GP or similar.

Related work: Vanhatalo et al. suggested to use the Laplace approximation to compute the posterior inference of a GP with Student-t likelihood [35]. The same authors later compared different strategies: MCMC [20], variational inference, and a modification of the expectation propagation (EP) algorithm with double-loop[11]. They showed that their modification of the EP is the most robust estimation method, although it has an increased computational cost. It is important to note that the vanilla version of EP does not converge at all for the Student-t likelihood [24]. In a different approach, Shah et al. [27] had the surprising result that a Student-t process prior with additive noise in the kernel behaves like a Gaussian or Student-t process posterior with a long-tailed likelihood, similar to the Student-t distribution. The surprise arise for the fact that the Student-t process prior is, by definition, robust to input variables $\mathbf{x}$ but not target variables y. The advantage of this method is that it is analytical, removing the extra cost of the iterative approximation. However, the actual statistical properties of the method were unclear. This idea was later proved to have the same marginal likelihood as a Student-t process with dependent Student-t noise, giving a probabilistic interpretation of the results [34]. This dependency in the noise might be a strong assumption for certain applications.

Furthermore, the critical parameter controlling robustness of the Student-t distribution it is the degrees of freedom ν, which is recommended to be at least 4 in practice [11, 14]. However, this parameter cannot be tuned independently in the dependent case. Furthermore, in the case of noisy data, learning the noise level is harder in the additive noise model due to the entanglement of the variables. This issue was recently addressed by Tang et al. [33] by using again the Laplace approximation from Vanhatalo et al. [35] to obtain an independent t noise model in a Student-t process.

In the present work, we have decided to compare both approaches: numerical approximation of the Student-t likelihood and the use of a Student-t process with additive noise.

2.1 Numerical approximation of Student-t likelihood

First, we will use the Student-t likelihood from Vanhatalo et al. [35]. The Student-t distribution has the form

$$t(y; f, \sigma_0, \nu) = \frac{\Gamma\left(\nu + \frac{1}{2}\right)}{\Gamma(\frac{\nu}{2})\sqrt{(\nu\pi)}\sigma_0}\left[1 + \frac{(y-f)^2}{\nu\sigma_0^2}\right]^{-\nu-\frac{1}{2}}, \tag{5}$$

where $f \equiv f(\mathbf{x})$, ν is the degrees of freedom and σ_0 is the scale parameter. In the Bayesian context, the Student-t distribution usually arises from a normal distribution with a conjugate hyperprior on the variance

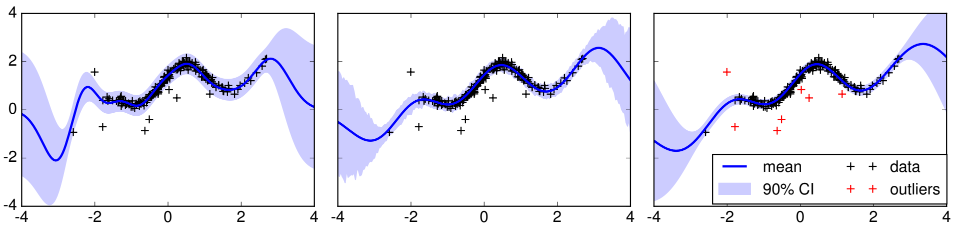

Figure 1: *Left*: Regression with outliers using the Gaussian likelihood can yield biased estimates and high variance. *Center*: The Student-t likelihood process allows for a better regression, but the estimates with respect to the non-corrupted values is biased and numerically instable. *Right*: We use the Student-t likelihood to remove the outliers and use a Gaussian likelihood with the remaining points. Note how the uncertainty in the left side of the plot is vastly underestimated for the Student-t, having the lower bound at ≈ -2 versus the lower bound of the filtered GP ≈ -4.

variable, such as the inverse-χ^2, the inverse gamma or even the Jeffreys prior [26]. For example, in this case, the model $y|f \sim t(y; f, \sigma_0, \nu)$ is equivalent to the original Gaussian likelihood with an hyperprior on the noise term σ_n. That is:

$$\begin{aligned} y|f, \sigma_n &\sim \mathcal{N}(f, \sigma_n^2) \\ \sigma_n^2|\nu, \sigma_0^2 &\sim \chi^{-2}(\nu, \sigma_0^2) \end{aligned} \tag{6}$$

Jylänki et at. [11] present different approximation methods for which we implemented the Laplace method (the simplest and most extended method) [35]. These works were mostly intended for regression applications where large amounts of data are available at once. In contrast, Bayesian optimization seeks to minimize the number of data points, often resulting in less data than most regression applications. Furthermore, observations arrive sequentially. In this context, we found the Laplace approximation to be reliable and numerically stable, because the lack of data resulted in a regularization effect. We also implemented the modified double-loop EP algorithm from [11], but preliminary results resulted in poor performance with many iterations converging to the wrong solution or not converging at all. We conjecture that this effect is because of the limited data available, and further research is required. For brevity, we present results in Section 5 using only the Laplace approximation.

We are interested in computing the predictive posterior from equation (2) with the new likelihood function. The Laplace approximation for the conditional posterior of the latent function, which we write as $p(f|\mathbf{y}, \mathbf{X}, \Lambda, \sigma_0^2, \nu)$, is constructed from the second order Taylor expansion of log posterior around the mode $\hat{f}$, which results in a Gaussian approximation:

$$p(f|\mathbf{y}, \mathbf{X}, \Lambda, \sigma_0^2, \nu) \approx \mathcal{N}(f|\hat{f}, \Sigma), \tag{7}$$

where $\hat{f} = \arg\max p(f|\mathbf{y}, \mathbf{X}, \Lambda, \sigma_0^2, \nu)$ is the maximum *a posteriori* and $\Sigma^{-1} = \mathbf{K}^{-1} + \mathbf{W}$ the Hessian of the negative log conditional posterior at the mode with, $\mathbf{W} = diag_i \left(\nabla_{f_i} \nabla_{f_i} \log p(y|f_i, \sigma, \nu)|_{f_i = \hat{f}_i} \right)$.

Finally, the new predictive distribution can be computed by marginalization of equation (7). That is:

$$\begin{aligned} \mu(\mathbf{x}_q) &= \mathbf{k}^T \mathbf{K}^{-1} \hat{f}, \\ \sigma^2(\mathbf{x}_q) &= k - \mathbf{k}^T \left(\mathbf{K} + \mathbf{W}^{-1} \right)^{-1} \mathbf{k}, \end{aligned} \tag{8}$$

where $k = k(\mathbf{x}_q, \mathbf{x}_q)$ and $\mathbf{k} = \mathbf{k}(\mathbf{x}_q, \mathbf{X})$. We refer to Vanhatalo et al. [35] for implementation details.

2.2 Student-t process with additive kernel noise

For comparison, we also include the Student-t process from Shah et al. [27], which we will the define in terms of the conditional posterior in the form of a multivariate Student-t distribution. This process completely changes the model presented in Section 1.1. For brevity we do not include the equations for the predictive distribution, hyperparmenter optimization and expected improvement with the new model. These can be found in the literature [27, 15, 26, 37].

In this case, the Student-t process is generated by placing an inverse gamma prior[1] on the scale parameter of the kernel matrix [15], that is, at the stage we replace the kernel matrix from equation (3) to

$$\mathbf{K} = \sigma_s^2 \left[\left(\mathbf{k}(\mathbf{x}_1, \mathbf{X}) \quad \ldots \quad \mathbf{k}(\mathbf{x}_t, \mathbf{X}) \right) + \mathbf{I}\sigma_n^2 \right]$$

with $\sigma_s^2 \sim \mathcal{IG}(a, b)$. This is the multivariate generalization of equation (6). Note also how the signal σ_s^2 and noise σ_n^2 variances become entangled. As reported by Shah et al. [27], this results are analogous to the

[1] Note that the inverse gamma is also equivalent to the scaled inverse χ^2 distribution $\chi_\nu^{-2}(\sigma_0^2) = \mathcal{IG}\left(\frac{\nu}{2}, \frac{\nu\sigma_0^2}{2}\right)$, which will define the Student-t in terms of the degrees of freedom [23] as mentioned in Section 2.

method of using the inverse Wishart process as a prior on $\mathbf{K}$. The multivariate Student-t distribution that generate the corresponding process is defined as:

$$t(\mathbf{y}; m, \mathbf{\Sigma}, a, b) = \frac{\Gamma\left(a + \frac{n}{2}\right)}{\Gamma(a)} \frac{1}{\sqrt{(2a\pi)^n \left|b^{-1}\mathbf{K}\right|}} \left[1 + \frac{b(\mathbf{y}-m)^T \mathbf{K}^{-1}(\mathbf{y}-m)}{2a}\right]^{-a-\frac{n}{2}} \tag{9}$$

where $m = m(\mathbf{x})$ is the mean function, which is generally assumed to be $m(\mathbf{x}) = 0$ and a and b are the parameters of the inverse gamma. Again, we refer to the literature for implementation details about the posterior inference [27, 15, 26, 37].

3 OUTLIER DIAGNOSTICS

This part of our method is independent of the robust regression model selected before (see Section 2.1 and Section 2.2), although for clarity we will assume that we are using the GP with Student-t likelihood from Section 2.1. Once we have built the robust regression model, we are able to identify the outliers from the rest of the data. As can be seen in Figure 1, the mean function computed with the robust regression (center) is not biased like the nonrobust regression (left). Therefore, we can determine that the outliers are the points in the tail of the predictive distribution. For example, note how the point close to $(-2, 2)$ introduces a large bias in the nonrobust regression. As a result, in the nonrobust regression, the mean prediction is much closer to the point.

For that purpose, we compute the upper and lower α-percentile of the predicted distribution as a classification threshold, where α is the assumed level of outliers. The selection of this parameter will determine the number of false positives and false negatives. High values of α will classify many points as outliers, reducing the effective sample size for Bayesian optimization. On the other hand, low values of α reduce the robustness of the method by misclassifying actual outliers.

No permanent rejection In theory, assuming that a single observation arrives per iteration, only that last observation should be questioned. However, because new data helps improve the model, we found that reclassifying all the points worked better, as new information allows better classification over past observations. Sometimes, points that initially were considered outliers can be found part of the model while, more frequently, points that were initially misclassified as acceptable are properly detected with a better model. For Bayesian optimization, the general assumption is that data points are *expensive* in some sense (such as time or energy), thus no point is permanently deleted or ignored.

Scheduling diagnostics Although the Student-t likelihood is able to identify m outliers out of $2m$ points, we have found that in practices it is reasonable to wait for a certain number of iterations before starting classifying data. Therefore, we propose to use the Student-t likelihood and posterior data filtering after n_{init} points and, then only once out of each n_s subsequent iterations. The motivation is to have a proper regression model with a correct estimate of the hyperparameters. We also found that, because of the sequential nature of Bayesian optimization, if the last point is misclassified as an outlier and removed in the Bayesian optimization, there is a large probability that the will be selected again in the next iteration, which will again might result in a misclassification and so on, wasting valuable resources. By waiting several steps before classifying data, we avoid this situation. Finally, the computational cost of the Student-t likelihood is more expensive than the Gaussian likelihood, thus, avoiding the use of the Student-t likelihood in most iterations results in lower overall cost.

Finally, once the outliers are classified and removed, the optimization is performed with a standard GP computed only with the remaining points, because it produces more stable and fast solutions (see Figure 1). This proved especially true at early stages, when the regression model is noisy and inaccurate, and some large misclassifications might happen. Knowing that there is a limitation on the number of outliers that the Student-t distribution is robust, we are able to detect if there has been a failure in the filtering process by checking if the number of outliers is larger than m for a total of $2m$ points.

4 BAYESIAN OPTIMIZATION WITH OUTLIERS

Our method is summarized in Algorithm 1. For those familiar with Bayesian optimization will recognize lines 1, 2 and 9-11 as the standard procedure: draw some initial points and select each new point based on the expected improvement computed using a fitted GP. As pointed out before, our contribution uses elements from the robust regression literature and the outlier diagnostics. Again, at this point, we assume that we are going to use the robust regression method from Section 2.1 which we have represented as the function `fitGPwithTlik`, although the algorithm would work with the Student-t process from Section 2.2 by replacing the function.

Algorithm 1 BO with outliers

```
Input: Total budget $T$, rejection threshold $\alpha$
Initial design of $p$ points (e.g.: LHS)
        $\mathbf{X} \leftarrow \mathbf{x}_{1:p}$        $\mathbf{y} \leftarrow y_{1:p}$

for $t = p+1 \ldots T$ 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 $length(\mathbf{y}_{in}) < \lfloor length(\mathbf{y})/2 \rfloor$ or
     not schedule($t$) then
        $\mathbf{X}_{in} \leftarrow \mathbf{X}$        $\mathbf{y}_{in} \leftarrow \mathbf{y}$

    $\mathbf{\Theta_g} \leftarrow$ fitGPwithGlik($\mathbf{X}_{in}, \mathbf{y}_{in}$)
    $\mathbf{x}_t = \arg\max_{\mathbf{x}} EI(\mathbf{x}|\mathbf{X}_{in}, \mathbf{y}_{in}, \mathbf{\Theta_g})$
    $y_t \leftarrow f(\mathbf{x}_t)$        $\mathbf{X} \leftarrow add(\mathbf{x}_t)$        $\mathbf{y} \leftarrow add(y_t)$
```

Next, we filter the outliers with the function `filterOutliers` introduced in Section 3. For this purpose, we need a predefined α term which defines the part of the tail that belongs to outliers. For our results, we have used $\alpha = 0.05$ which correspond to the 5% percentile. We also considered lower values, such as, $\alpha = 0.01$ which reduced the number of false positives. We found than an agressive threshold worked better in practice because we are not permanently rejecting any point. False positives are usually correctly classified in subsequent iterations, when more data is available. Furthermore, as commented in Section 2, the Student-t model assumes that at least half of the points are inliers. We can monitor at every iteration if the number of inliers `length`($\mathbf{y}_{in}$) is at least half of the total number of points `length`($\mathbf{y}$), if that is not the case, either the modelling or the data assumptions are broken and we decide to wait for more data.

Finally, for the `schedule`, in all the experiments we have used an initial delay of $n_{\text{init}} = 10$ iterations and the filtering was performed one every $n = 2$ iterations. However, we found that the results were not fundamentally different using different schedules and it is more of a tradeoff of the computation cost, convergence speed and total budget.

5 RESULTS

We evaluated our method on a set of benchmarks and realistic applications. For the benchmarks we have compared a set of different methods: *our method* according to algorithm Algorithm 1; BO with *t-likelihood* and *t-process* which corresponds to the methods presented in Section 2.1 and Section 2.2 respectively where all the outliers are accommodated in the regression model and no rejection step is performed; *baseline* which uses the standard Gaussian likelihood as presented in Section 1.2. We also included a *no outliers* experiment which corresponds to the ideal scenario with no outliers present. For reproducibility purpose, the outliers were artificially generated in all cases so that the distribution is equivalent for all the methods. Common random numbers were used between different methods. All the experiments were repeated for 10% and 20% outlier proportion using exactly the same configuration parameters to illustrate the robustness of the application to different levels of outliers. All plots represent the average outcome and 95% confidence bounds over 20 trials.

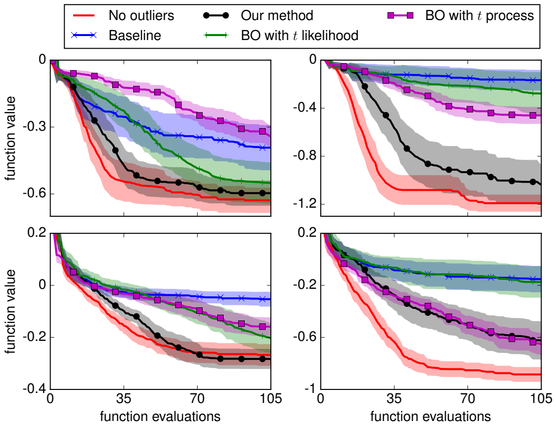

Figure 2: 8D randomly generated functions with outliers with various Bayesian Optimization strategies. *left*: 10% outliers. *right*: 20% outliers. *top*: Matérn generated (within model comparison). *bottom*: rational quadratic generated (out-of-model comparison).

5.1 Numerical benchmarks

For the numerical benchmarks we used the methodology from Henning and Schuler [8]. We generated a set of 8D random functions from two types of Gaussian processes. For the *within model comparison*, we have generated the samples from a GP with the same C^4 Matérn kernel used in optimization; while for the *out-of-model* comparison, we generated the samples from a GP with a rational quadratic kernel with $\alpha = 2$. Outliers were iid sampled from a uniform distribution $y_{outlier} \sim \mathcal{U}(1, 2)$. Results can be seen in Figure 2. For each experiment configuration (kernel and ratio of outliers), we generate a different random function, resulting in different ranges for the vertical axis. Our method outperforms both robust regression methods and it is able to reach performance comparable to not having outliers at all. We can also see that if not considering the outliers, the effect is devastating, resulting

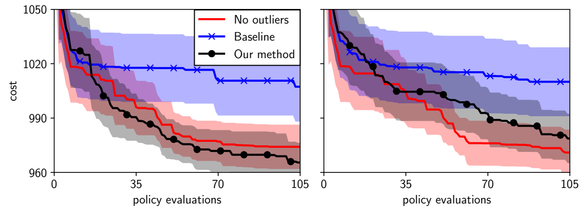

Figure 3: Optimization of the robot walking policy. *left*: For a 10% outlier rate, the Student-t likelihood is able to prune some of the out-of-model points which allows better refinement than the standard GP baseline. *right*: When the number of outliers is larger (20%), the Student-t likelihood allows us to roughly recover the performance in the absence of outliers.

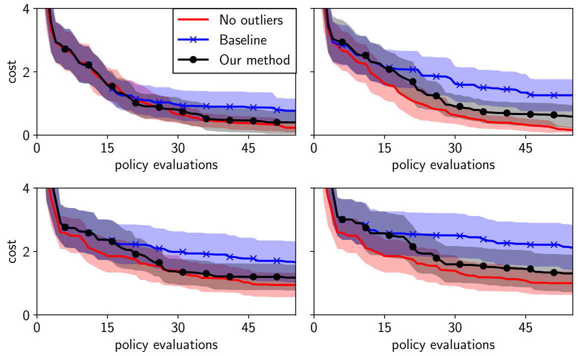

Figure 4: Optimization of the robot pushing policy. *top left*: 3D input, 10% outliers. *top right*: 3D input, 20% outliers. *bottom left*: 4D input, 10% outliers. *bottom right*: 4D input , 20% outliers.

in the optimization being stuck. Between the robust regression methods there is no clear winner. We found the Student-t likelihood to be more reliable between experiments, thus we decided to use that approach within our own method as pointed out in Section 4.

5.2 Robot planning and control

Active policy search [17] is a reinforcement learning method to control a robot or autonomous agent by refining its policy using Bayesian optimization on the reward function. It has been successfully applied for robot walking in controlled environments [2]. In this case, the objective of the optimization is to find a stable policy, even in the presence of external perturbations. However, in some trials, the robot might find obstacles or perturbations that are physically impossible to overcome. Thus, the robot returns a poor reward even if the policy is good in other conditions.

For example, these days it is common to find experiments of robot learning walking patterns in laboratory conditions, where external interferences are reduced or controlled. In many cases, the objective of the learned controller is to be able to react to some of those external perturbations, like a light push or a terrain slope. However, in the near future, robots will have to learn and adapt in all kind of environments with uncontrolled conditions, some of which would be physically impossible to compensate. Thus, the learning process must be able to identify when the failure is due to a bad policy or a strong perturbation.

Robot walking For this experiment, we have used a full body dynamic simulator [3] of a robot walking along with a predefined set of controllers from which we selected 6 parameters to tune, the stance and swing acceleration terms. In the scenarios with outliers, we have simulated the failures as the robot reaching a insurmountable obstacle at a random time during the trajectory, resulting in the robot tripping and falling. Therefore, the resulting reward is similar to the reward obtained with a bad policy, which also results in a *crash* state at different times.

It has been shown that robot policy search is a complex problem for Bayesian optimization due to the nonstationary behavior of many reward functions [16]. A large number of nearly flat *crash* results yield combined with large variability near the optimum results in an underperforming GP model because all the results cannot agree to a single stationary function. As can be seen in Figure 3, the Student-t likelihood is also classifying some of the actual bad policies as outliers because they do not agree with the stationary model, resulting in a better convergency. However, for a larger number of outliers, the combination nonstationarity and outliers makes it harder for the filtering.

Robot pushing We have replicated the robot pushing experiments from Wang and Jegelka [36]. The experiment is based on the *pre-image* setup from Kaelbling and Lozano-Perez [12] and consist on performing active policy search to select the pushing action that minimize the distance of the pushed object to the goal location. The first function we tested has a 3D input: robot location (r_x, r_y) and pushing duration t_r. The second function has a 4D input: robot location and angle (r_x, r_y, r_θ), and pushing duration t_r. We select 20 random goal locations for each function to test if our method can learn where to push for these locations. In normal conditions, the goal was placed in a reachable position. Failures and outliers were modeled by placing the object just outside the reachable region to represent a configuration or sensor problem for which the distance to the goal is incorrectly measured. Figure 4 shows the results of both problems.

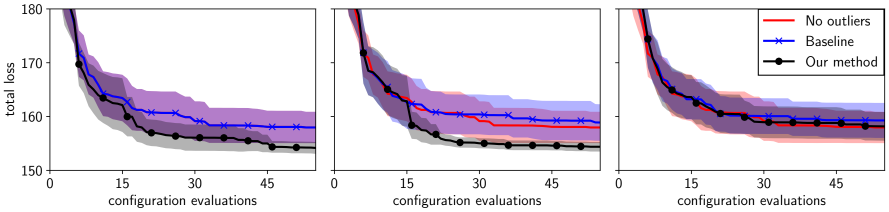

Figure 5: Optimization of the variational autoencoder on the MNIST dataset. *left*: No outliers (thus the “Baseline” coincides with the “No outliers case”). *center*: 10% outliers. *right*: 20% outliers.

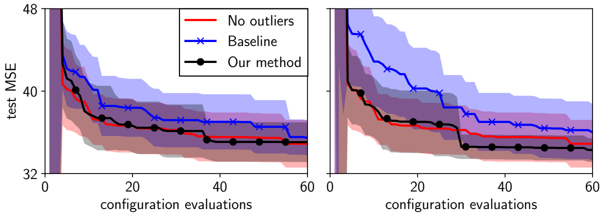

Figure 6: Optimization of the feed-forward neural network on the Boston housing dataset. *left*: 10% outliers. *right*: 20% outliers.

5.3 Hyperparameter tuning

For the problem of hyperparameter tuning for which Bayesian optimization is growing in popularity, there are many possibilities for outliers: bugs, failures, etc. There are some sources of outliers that are intrinsic to the procedure. For example, during hyperparameter tuning, early stopping during training might be pushed to the limit to guarantee no overfitting and reduce the already expensive computational cost. There are methods which directly reallocate resources based on the performance at early stages [13]. On the other hand, the initialization is nontrivial [32], with random initialization of variables resulting in very different behaviors at early stages. For instance, the performance of a set of hyperparameters may seem poor after few epochs because the initialization occurred in a complex region (flat, saddle points, etc.) and not because the set of hyperparameters was worse than others. Therefore, early stopping results in some points being actual outliers. This effect is present in our experiments, where our method is able to achieve better performance than standard BO with no induced outliers.

Variational autoencoder Variational autoenconders (VAE) are a powerful generative method for deep learning. In this experiment we train a VAE for the MNIST dataset [6]. The hyperparameters we tune are the number of nodes in the hidden layer and learning rate, learning rate decay, and ϵ constant for the Adam optimizer. In this case, an outlier simulates an IO failure 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 5. We can see how even in the case where no outliers were induced externally, our method outperforms standard BO, which reinforces the theory that there are already outliers present. Besides, for a high level of outliers (20%) the performance drop, suggesting that the number of simulated outliers in addition with the existing outliers reaches a point near the limit of robustness.

Feedforward network Inspired by Wang and Jegelka [36], we use a single layer feedforward neural network on the Boston housing dataset. The 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. An outlier in this optimization consisted of running the neural network 5 epochs, rather than the standard 20 epochs. We can see in Figure 6 how the behaviour is similar to the VAE, improving the convergence over the case with no outliers.

6 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 in many benchmarks and realistic applications showing that our method is suitable for practical Bayesian optimization in the presence of outliers. Furthermore, we have seen how our method has been able to outperform standard Bayesian optimization in a controlled environment without induced outliers. This highlights the importance of this approach and the possible presence of outlier data even in supposedly controlled environments and lab conditions. Finally, we have experimentally proven that Bayesian optimization with a robust surrogate model designed to accomodate outliers produces suboptimal 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] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, 2016.

[3] Christopher G. Atkeson. Using Function Optimization To Find Policies: Walking. `http://www.cs.cmu.edu/~cga/dynopt/ass2/`, 2017.

[4] Josef Dick and Friedrich Pillichshammer. *Digital nets and sequences: Discrepancy Theory and Quasi–Monte Carlo Integration*. Cambridge University Press, 2010.

[5] Gregory Fasshauer and Michael McCourt. *Kernel-based approximation methods using Matlab*, volume 19. World Scientific Publishing Co Inc, 2015.

[6] François Chollet. Variational autoencoder with Keras. `https://github.com/fchollet/keras/blob/master/examples/variational_{}autoencoder.py`, 2017.

[7] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

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

[9] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, Oct 2004.

[10] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION-5*, page 507523, 2011.

[11] 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.

[12] Leslie Pack Kaelbling and Tomas Lozano-Perez. Implicit belief-space pre-images for hierarchical planning and execution. In *IEEE Conference on Robotics and Automation (ICRA)*, 2016.

[13] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with Bayesian neural networks. In *International Conference on Learning Representations (ICLR) 2017 Conference Track*, April 2017.

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

[15] 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.

[16] Ruben Martinez-Cantin. Bayesian optimization with adaptive kernels for robot control. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3350–3356, 2017.

[17] Ruben Martinez-Cantin, Nando de Freitas, Jose Castellanos, and Arnaud Doucet. Active policy learning for robot planning and exploration under uncertainty. In *Proc. of Robotics: Science and Systems*, 2007.

[18] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. In L.C.W. Dixon and G.P. Szego, editors, *Towards Global Optimisation 2*, pages 117–129. Elsevier, 1978.

[19] A.W. Moore and J. Schneider. Memory-based stochastic optimization. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1066–1072. The MIT Press, 1996.

[20] Radford M Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. *Technical Report 9702, Dept. of statistics and Dept. of Computer Science, University of Toronto, arXiv preprint physics/9701026*, 1997.

[21] José Nogueira, Ruben Martinez-Cantin, Alexandre Bernardino, and Lorenzo Jamone. Unscented Bayesian optimization for safe robot grasping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016.

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

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

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

[25] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*, volume 589. John Wiley & Sons, 2005.

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

[27] Amar Shah, Andrew Gordon Wilson, and Zoubin Ghahramani. Student-t processes as alternatives to Gaussian processes. In *AISTATS, JMLR Proceedings. JMLR.org*, 2014.

[28] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P 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.

[29] Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, pages 2960–2968, 2012.

[30] 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.

[31] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4134–4142. Curran Associates, Inc., 2016.

[32] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1139–1147, 2013.

[33] 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 *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2822–2828, 2017.

[34] Qingtao Tang, Yisen Wang, and Shu-Tao Xia. Student-t process regression with dependent Student-t noise. In *ECAI*, 2016.

[35] 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.

[36] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3627–3635, 2017.

[37] 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.