A modified dividing local Gaussian processes algorithm for theoretical particle physics applications

Timo Lohrmann[1], Anders Kvellestad[2], Riccardo De Bin[1]

[1] Department of Mathematics, University of Oslo, Norway
[2] Department of Physics, University of Oslo, Norway

E-mail for correspondence: timolo@math.uio.no

Abstract: We explore the use of dividing local Gaussian processes in the context of theoretical particle physics. We adapt an existing algorithm to the specific problem, trading some speed for a better precision. An intensive sensitivity analysis is performed.

Keywords: Gaussian processes; dividing local Gaussian processes; theoretical particle physics.

1 Substantive problem and approach

Physicists have proposed a wide range of new theories that extend the Standard Model of particle physics with new types of fundamental particles and new interactions. Such new theories are collectively known as Beyond-the-Standard-Model (BSM) theories. A BSM global fit refers to a large-scale parameter estimation study, in which the preferred values or ranges for the parameters of a BSM theory are determined by simultaneously comparing the theory's predictions to the results from all relevant experiments. The basis for this parameter estimation is a joint likelihood function, which is a function of the parameters of the BSM theory. The evaluation of the likelihood, however, involves many time-consuming physics calculations and simulations, which limit the scope of current BSM global fits. Our goal is to introduce an algorithm that can provide fast, per-point surrogate models for these time-consuming physics computations, or for the likelihood function directly. For simplicity, we will focus on the latter case. Since the set of relevant experimental results that enter the likelihood function is usually

This paper was published as a part of the proceedings of the 36th International Workshop on Statistical Modelling (IWSM), Trieste, Italy, 18–22 July 2022.

updated between every new BSM global fit, a key requirement for the algorithm is that it is able to train the surrogates *on-the-fly* during the execution of the BSM global fit.

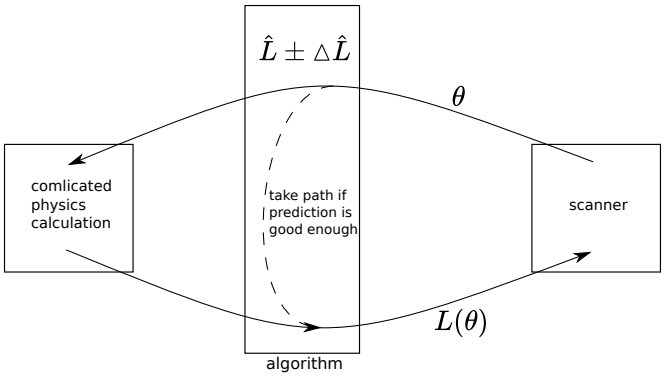

FIGURE 1. Conceptual sketch of how the algorithm fits into the computational framework of a BSM global fit.

Figure 1 provides a sketch of how the algorithm fits into the computational framework of a BSM global fit. An adaptive sampling algorithm (the "scanner") is responsible for sampling parameter points θ from the BSM parameter space. Each such point is passed to a code responsible for carrying out the expensive physics calculations that results in a single likelihood value $\mathrm{L}(\theta)$ for the given point. This likelihood is then passed back to the scanner, which uses it to determine how to pick the next θ point. Our algorithm will operate as a middle layer, that can intercept the communication between the scanner and the true likelihood computation. By learning from the continuous stream of θ and $\mathrm{L}(\theta)$ values, the algorithm can gradually learn the likelihood function in the regions of θ space explored by the scanner. Once the algorithm has obtained a good enough estimate $\hat{\mathrm{L}}(\theta)$ for $\mathrm{L}(\theta)$ in a certain region, it can short-circuit the likelihood calculation for any future θ points in this region by passing the fast estimate $\hat{\mathrm{L}}$ directly back to the scanner. However, for θ points where the estimate $\hat{\mathrm{L}}$ is still highly uncertain, the algorithm will pass the point on to the true likelihood calculation and learn from the $\mathrm{L}(\theta)$ value being passed back.

2 Data

We use data from a recent BSM global fit by the GAMBIT Collaboration (2019). Here the target likelihood is a function of four free BSM input parameters. To emulate the global fit we pass the data points through our algorithm one by one, in the order they were sampled by the differential evolution scanner used in the GAMBIT study. For comparison with earlier

work, discussed below, we also perform tests with the SARCOS dataset available from `www.gaussianprocess.org/gpml/data`.

3 Model(s)

We start from the dividing local Gaussian process (DLGP) algorithm of Lederer et al (2020). The algorithm uses the stream of input data to dynamically divide the input space into sub-spaces and train a Gaussian process (GP) on each of them. This dynamic splitting can be viewed as a growing tree structure, where the outermost nodes, called *leaves*, each contain a part of the data and a corresponding GP. Many decisions have been made to implement the algorithm: A leaf splits into two child leaves when a certain number of input points $\bar{N}$ is reached. The split is performed along the input dimension with the largest variance. A smoothing effect is created by randomly assigning some points to the sibling leaf. The probability of assigning points decays linearly with the distance from the splitting value. The predictive distribution of the tree is obtained by computing the probability of assigning the point $\mathbf{x}$ to leaf j, $\tilde{p}_j(\mathbf{x}) = \prod_{i=1}^{\nu_j} p_{\lfloor \frac{j+1}{2^i} \rfloor - 1}(\mathbf{x})$, with depth ν_j, that leads to the predictive distribution

$$p_{\mathrm{DLGP}}(f(\mathbf{x})|\mathbf{x}, X, y) = \sum_j \tilde{p}_j(\mathbf{x}) p_{\mathrm{GP}_j}(f(\mathbf{x})|\mathbf{x}, D_j),$$

where p_{GP_j} is the prediction of the GP of the j-th leaf on the data set D_j. X and y are the data set and the target variable, respectively. The predictive distribution follows a normal distribution with mean μ_* and variance σ_*^2,

$$\mu_*(\mathbf{x}) = \sum_{j \in \mathcal{I}} \tilde{p}_j(\mathbf{x}) \mu_j(\mathbf{x}) \tag{1}$$

and

$$\sigma_*^2(\mathbf{x}) = \sum_j \tilde{p}_j(\mathbf{x}) \left(\sigma_j^2(\mathbf{x}) + \mu_j(\mathbf{x})\right) - \mu_*(\mathbf{x}), \tag{2}$$

where $\mu_j(\mathbf{x})$ and $\sigma_j^2(\mathbf{x})$ are the mean and variance of leaf j.

3.1 Extension of previous work

Our work extends this base-line implementation by offering additional options for the aforementioned decisions, regulating them by hyperparameters. We offer the chance to perform the splitting along the first principal component. We include a Gaussian "decay shape". We allow varying the covariance function from the squared exponential originally used in Lederer et al (2020) to Matérn kernels. And most importantly, we update the parameters of the GPs each time a new data point is added. This provides a slower, but more precise and flexible algorithm than the original approach of fixing the GP parameters based on the first 100 data points.

4 Results

4.1 Target quantities for model selection

We aim at getting as close as possible to the truth both in terms of the expected prediction (RMSE) and in terms of variability. For the later, we compute the mean difference between predicted uncertainty and standard deviation of the input noise σ_ϵ,

$$\Delta_\sigma = \frac{1}{N}\sum_i \sigma_*(\mathbf{x}_i) - \sigma_{\epsilon,i}. \tag{3}$$

4.2 DLGP performance

As we can see in Fig. 2, the RMSE steadily decreases with the number of input points. Moreover, we observe that the covariance function is crucial to the performance of the algorithm. Over the whole range of input points, the Matérn kernel with $\nu = \frac{3}{2}$ performs equally or better than a Gaussian kernel while keeping all other parameters the same.

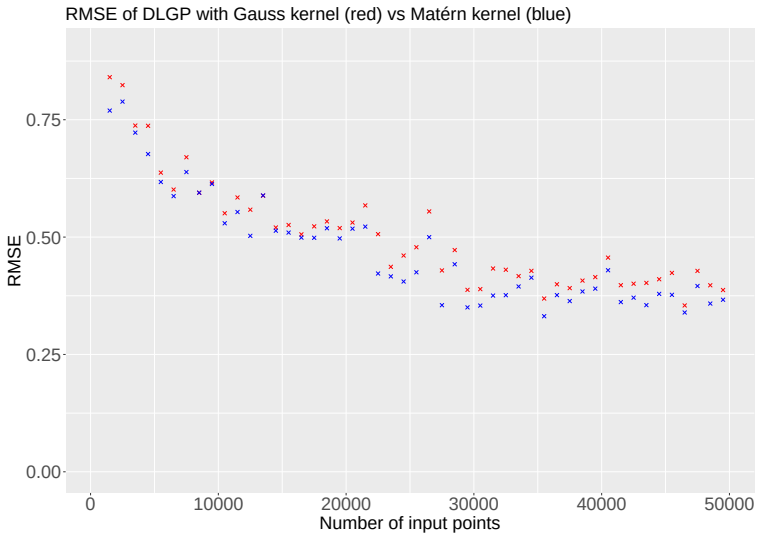

FIGURE 2. Example of the performance of our version of the DLGP on the GAMBIT data. The crosses are the RMSE aggregated over 1000 data points. The parameters of both DLGPs where identical except for the covariance function. In the case of the red crosses, a Gaussian kernel was used. Similarly, a Matérn kernel with $\nu = \frac{3}{2}$ was used for the blue crosses. We can clearly see that choosing the Matérn kernel leads to improved results. The other parameters for the DLGP are $\bar{N} = 100$, splitting along the first principal component while using the median as center. A linear overlap shape with 1% overlap was used.

In Fig. 3, we see that σ_* converges to σ_ϵ and remains in its vicinity afterwards. This shows that the DLGP has learned to estimate the variability of the training points to an adequate degree.

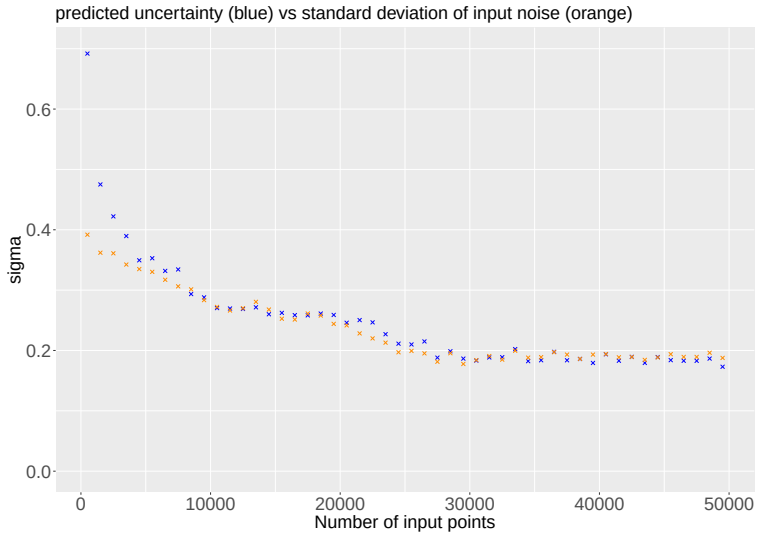

FIGURE 3. Example of the performance of our version of the DLGP on the GAMBIT data. The blue crosses are the predicted uncertainty aggregated over 1000 data points. The orange crosses are the aggregated σ_ϵ values. We observe that σ_* converges to σ_ϵ remains in its vicinity afterwards. For this DLGP, the following parameters where chosen: Matérn kernel with $\nu = \frac{3}{2}$, $\bar{N} = 100$, splitting along first principal component while using the median as center. A linear overlap shape with 1% overlap was used.

4.3 Sensitivity Analysis

A grid-search is performed to find the best combination of hyperparameters. We explore 96 different GP tree configurations and perform tests both with the data from the GAMBIT Collaboration (2019) and the SARCOS data used in Lederer et al (2020). Due to the computational expense, we use only the 50000 first data points in these initial tests. For the most promising GP tree configurations we then carry out tests using the full data sets.
When describing the performance of the tree, we refer to an improvement if a parameter improves on RMSE and Δ_σ. We observe that the choice of covariance function impacts the performance the most. The Matérn kernel function with $\nu = \frac{3}{2}$ clearly outperforms the other kernels. In agreement with the SARCOS study, we observe that for the GAMBIT data a larger maximum number of points per leaf $\bar{N}$ improves the result. Furthermore, increasing the overlap between the sibling leaves yields improvements with

regards to the RMSE, in contrast to the SARCOS results of Lederer et al (2020). However, σ_* tends to be larger and less consistent for larger overlaps as Δ_σ is significantly larger in this case. The overlapping shape has a consistently insignificant contribution to the performance. There seems to be a connection between using the median to define the center of the splitting dimension and splitting along the first principal component: Using the median instead of mean decreases the performance if all other parameters are equal, and the same is true for splitting along the first principal component instead of the dimension with maximum variance. However, the aforementioned combination performs equally well as mean and maximum variance as splitting criterion.

5 Conclusion

In this paper, an application of dividing local Gaussian processes (Lederer et al, 2020) is elaborated. In the presented theoretical physics use case, the focus lies more on high quality than fast predictions. We showed that it is worth considering improvements to the original algorithm. In particular we showed that working on the covariance function can have a large impact on the results. We believe that this online learning algorithm can also be applied in other fields where fast and continuous updating is relevant, such as finance or chemistry.

References

GAMBIT Collaboration: Athron, P., Balázs, C. et al (2019). *Combined collider constraints on neutralinos and charginos*. Eur. Phys. J. C 79, 395

Lederer, A., Conejo, A.J.O., Maier, K., et al (2020). *Real-time regression with dividing local Gaussian processes. arXiv preprint*, arXiv:2006.09446.

Carl Edward Rasmussen and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning*. The MIT Press, 2006. ISBN 0-262-18253-X

Olivier Roustant, David Ginsbourger, Yves Deville (2012). *DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization*. Journal of Statistical Software, 51(1), 1–55