GP Hyper-Parameter Fitting

Jared Carlson

February 21, 2020

1 Overview

Currently we are using the Squared exponential as our kernel in the Gaussian Processes interpolation, eq. 1. The choice of hyper parameter λ in this kernel determines the length scale over which grid points are highly correlated. In effect, lower λ's produce smoother solutions and higher λ's more oscillatory ones. The optimal choice of λ is dependent on the runtime values. We can therefore choose a global set of λs (one for each level of cell refinement) to use through out the simulation, calculate λ for each reconstruction, or a combination of the two. Each approach has its costs and benefits.

$$\phi(x_1, x_2) = \exp(-||x_1 - x_2||_2^2 \lambda^2) \tag{1}$$

2 Global Fitting

There are two common approaches to fitting hyper parameters in GP, cross validation or marginal likelihood. In both cases, we have the analytic form of the function and their derivative [1]. So far I have only explored the using the marginal likelihood approach. To do this I have used a simple root finding package in python to find the root of the Marginal Likelihood's derivative. This only works when things are well behaved, e.g. the gram matrix has a small conditions number. In general, it is rather an ill conditions non-convex optimization problem and more work needs to go into finding a stable method for optimizing for λ.

The benefits to global fitting is that only one value of λ^2 is needed for each level of grid refinement and can be calculated either once at the start, every time step, or at some interval. Because the computational cost is $\mathcal{O}(n^3)$ (some methods get it down to $\mathcal{O}(n^3)$), including all points in the domain is impractical and thus sampling from a characteristic

sub-domain preferred [2][3]. Because this method has only one degree of freedom per a refinement level, it is doubtful that using a characteristic subset of the domain will effect the outcome of λ.

The downside to this approach is that if the solution varies in smoothness at a given level of refinement then the accuracy will suffer. The use of GP-WENO may help to remedy this discrepancy. It may also be the case the AMR tends to refine the resolution of the grid until oscillation on a level of refinement are similar throughout the simulation, however this needs to be explored.

3 Local Fitting

3.1 RBF and GP Equivalence

I will show that with a choice of the $\bar{f}_* = 0$ and $\bar{f} = 0$ for the Gaussian Process formulation, it is equivalent to using radial basis functions[1]. As shown in [4], interpolated value at a cell interface is

$$\tilde{f}_* = T_*^T C^{-1} G \tag{2}$$

where

$$C_{kh} = \int K(x, y) dg_k(x) dg_h(y) \tag{3}$$

$$T_{*,k} = \int K(x, x_*) dg_k(x) \tag{4}$$

For the radial basis functions I use the formulation and notation in [5] section 3.2 . From that we have

$$\tilde{f}_* = \phi_* \alpha^T \tag{5}$$

Note that I am using $\tilde{f}_*$ instead of $s(x)$ as the point of interpolation as defined in [5]. From eq 3.9 from [5] we get

$$A\alpha^T = G \tag{6}$$

Again using G for our cell averaged values as opposed to their $f|_\Theta$. Thus we have

$$\tilde{f}_* = \phi_* A^{-1} G \tag{7}$$

[1] Assuming both are using the squared exponential function for their kernel/basis

I first show the equivalence A and C.

$$\begin{aligned} A &= \lambda_C^x \lambda_Z^\xi \phi(||x-\xi||) \\ &= \frac{1}{|C|}\int_C \frac{1}{|Z|}\int_Z \phi(||x-\xi||)d\xi dx \\ &= \int\int \phi(||x-y||)dg_k(y)dg_h(x) \\ &= \int\int K(x,y)dg_k(y)dg_h(x) \\ &= C \end{aligned}$$

Where $dg(\cdot)$ is as defined in [4]. For ϕ_* and T_*^T

$$\begin{aligned} \phi_{*,k} &= \lambda_C^\xi \phi(||x_*-\xi||) \\ &= \frac{1}{|C|}\int_C \phi(||x_*-\xi||)d\xi \\ &= \int \phi(||x_*-y||)dg_k(y) \\ &= \int K(x_*,x)dg_k(x) \\ &= T_{*,k} \end{aligned}$$

Thus $\tilde{f}_* = T_*^T C^{-1} G = \phi_* A^{-1} G$. Because these are equivalent, it means that the optimal choice for the RBF is the same for the GP kernel.

3.2 Optimal Parameters

Work done by Hyoseon Yang has shown that the accuracy of the RBF interpolation when using the squared exponential can be increased by a correct choice of λ [6]. This is done through an analysis of the Taylor expansion of the interpolation. I have created a Mathematica code that calculates what λ^2 needs to be in order to achieve a desired degree of accuracy[2]. For N=2 and $u = u(x_{\text{interface}})$ we have

$$\lambda^2 = -\frac{u''(x_{1/2})}{6u(x_{1/2})} + \mathcal{O}(\Delta x^4) \tag{8}$$

and for N=4 we have

[2]https://gitlab.msu.edu/carls502/gaussian-processes-learning/blob/7fb41cd7abdb9471a03bacba80a402062de2ec41/HyperParameterOptimization/FindHyperparameter.nb

$$\lambda^2 = \frac{5u'' + \sqrt{5}\sqrt{5(u'')^2 - 3uu^{(4)}}}{30u} + \mathcal{O}(\Delta x^6) \tag{9}$$

.

Note that the number of points N is the cell average values of the N/2 cells to the left and right of the interface. The order of accuracy for u and its derivatives at the interface only need to be $\mathcal{O}(\Delta x^2)$ accurate when determining λ^2. When the desired order of accuracy is 2 less then the choice of lambda is irrelevant. If we want higher order approximations with the same number of points then we must find higher order approximations for interface values.

Thus we can use $\lambda = 1$ to get the approximations needed to calculate the optimal λ^2. For example, consider the case N=6. The approximation then becomes,

$$\hat{u} = u + 6\Delta x^6 \lambda^6 u + 3\Delta x^6 \lambda^4 u'' + \frac{3}{10}\Delta x^6 \lambda^2 u^{(4)} + \frac{1}{140}\Delta x^6 u^{(6)} + \mathcal{O}(\Delta x^8)$$

If we choose $\lambda = 1$ we get

$$\hat{u} = u + \mathcal{O}(\Delta x^6)$$

. These weights only need to be computed once at startup and are already needed for the WENO step. More work needs to be done here but it seems very promising.

The benefit of local fitting is that an optimal choice of λ can be chosen for each point, which can guarantee high accuracy. The downside is that the gram matrix must be recomputed and inverted, along with a low accuracy interpolation, for every point at every time step. A possible approximation to this would be to pick from a set of precomputed weight vectors (like is done for different refinement levels) that correspond to different ranges of λ. A lot of work is still needed for this method.

References

[1] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning, Cambridge, MA, USA: MIT Press, Jan. 2006.

[2] K. Wang, G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson, "Exact gaussian processes on a million data points," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, E. Fox, and R. Garnett, eds.), pp. 14648–14659, Curran Associates, Inc., 2019.

[3] H. Liu, J. Cai, Y. Ong, and Y. Wang, "Understanding and comparing scalable gaussian process regression for big data," *Knowledge-Based Systems*, 11 2018.

[4] A. Reyes, D. Lee, C. Graziani, and P. Tzeferacos, "A new class of high-order methods for fluid dynamics simulations using gaussian process modeling: One-dimensional case," *Journal of Scientific Computing*, vol. 76, no. 1, pp. 443–480, 2018.

[5] C. Bigoni and J. S. Hesthaven, "Adaptive weno methods based on radial basis function reconstruction," *Journal of Scientific Computing*, vol. 72, no. 3, pp. 986–1020, 2017.

[6] H. Y. J. Y. Youngsoo Ha, Chang Ho Kim, "Improving accuracy of the fifth-order weno scheme by using the exponential approximation space.".