Gaussian Process for Heat Equation Numerical Solution

Zurab Kiguradze

Electromagnetic Compatibility Laboratory, Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO, USA

E-mail: kiguradzz@mst.edu

The purpose of the present study is to develop a Machine Learning (ML) approach to the solution of the partial differential equations (PDEs). Due to this being our first attempt in this direction, in this note, we consider the simple heat equation.

In the domain $\Omega = (0,1) \times (0,T)$, $T = const > 0$, let us consider the initial-boundary value problem for the heat equation:

$$\begin{gathered} \frac{\partial U(x,t)}{\partial t} - a\,\frac{\partial^2 U(x,t)}{\partial x^2} = f(x,t), \quad (x,t) \in \Omega, \\ U(-1,t) = U(1,t) = 0, \quad t \in [0,T], \\ U(x,0) = U_0(x), \quad x \in [-1,1], \end{gathered} \tag{1}$$

where a is a positive constant and U_0 is a given function.

Our aim is to find the approximate solution $u(t,x)$ at $t > 0$ of problem (1). Introducing the uniform grid for the time variable $t_n = \tau \cdot n$, $\tau = T/N$ and applying Euler scheme, we get

$$u_n(x) = u_{n-1}(x) + a\tau\,\frac{d^2 u_{n-1}(x)}{dx^2} + \tau f_n(x), \quad n = 1, \ldots, N, \tag{2}$$

where N is a positive integer and $u_n(x) = u(t_n, x)$.

Although there are many methods for solving, even more, complex PDEs (see, for example, [2,3] and the references therein), our purpose, as we already mentioned, is to apply one of the well-known ML methods for solving problem (1). In particular, our goal is to design the Gaussian Process (GP) [6,9] for the heat equation to predict the solution [7,8].

The GP is an extension of Multivariate Gaussian Distribution. In turn, the multivariate Gaussian distribution is a generalization of the one-dimensional normal distribution to higher dimensions. For example, if there are inputs from two-dimensional space, then for any cross-section over the fixed one-dimensional input we get Gaussian distribution along each axis (see, Figure 1).

Probability density function (pdf) in two-dimensional space is given as follows:

$$pdf(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\, e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2} - \frac{(y-\mu_y)^2}{2\sigma_y^2}}.$$

In general, the pdf of the Multivariate Gaussian distribution in d dimensions is defined by the following formula:

$$pdf(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\, e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)},$$

where $\mu = (\mu_1, \mu_2, \ldots, \mu_d)$ is mean vector of $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and Σ^{-1} is the inverse of the $d \times d$ positively defined covariance matrix $\Sigma = cov[\mathbf{x}]$, which is constructed by one of the so-called covariance functions [6,9].

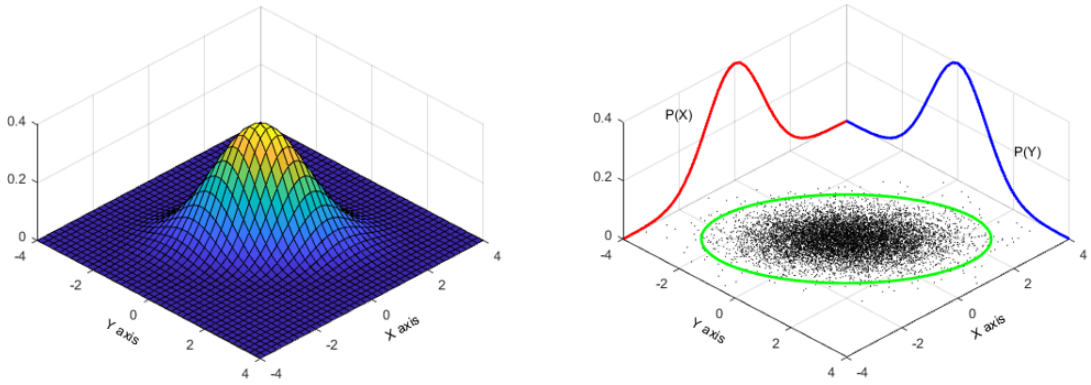

Figure 1. 2D multivariate Gaussian distribution and its cross-section projections.

GP presents one of the most important ML approaches based on a particularly effective method for placing a prior distribution over the space of functions [1, 6, 9]. GP can serve as an effective algorithm for function approximation. As an example let us consider samples from the GP, mean function, and some observation points where the values of the approximated function are known. Figures 2 depict 5 sample functions from the prior distribution over functions specified by a particular Gaussian with two (left) and four (right) observation points. Sample functions are plotted as dashed lines, the mean function is shown as a black solid line, observed points represented as red crosses, and the shaded region denotes uncertainty region. As it can be seen the uncertainty region is narrowing when the number of observed points is increasing. The equations for obtaining the mean function, which can be considered as the function approximation can be derived from the Sherman–Morrison–Woodbury formula [1, 5, 6, 9].

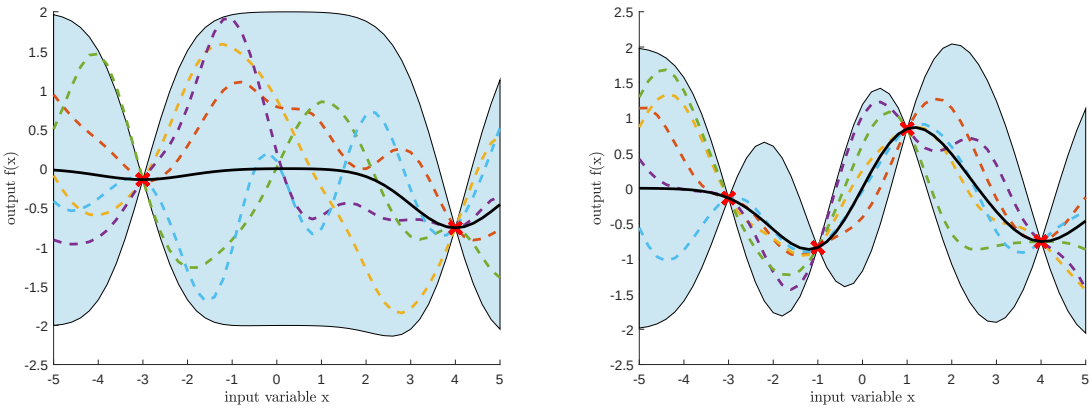

Figure 2. Fife samples from Gaussian posterior (dashed) and its mean (solid black) with the dataset of two and four points (red crosses). For colored figures, please refer to the online version.

Coming back to problem (1), let us set GP prior on u_{n-1} according to [6, 9]

$$u_{n-1}(x) \sim \mathcal{GP}(0, k_{n-1,n-1}(x, x', \theta)), \tag{3}$$

where $k_{n-1,n-1}$ is the kernel (covariance function) of the GP and θ represents the vector of the hyper-parameters of the covariance function [6–9].

Note that there are many different types of covariance functions. In this study, the neural

network covariance function is used [9]

$$k(x,x',\theta)=\frac{2}{\pi}\,\sin^{-1}\left(\frac{2(\sigma_0^2+\sigma^2xx')}{\sqrt{(1+2(\sigma_0^2+\sigma^2x^2))(1+2(\sigma_0^2+\sigma^2x'^2))}}\right), \tag{4}$$

where θ is two component hyper-parameter vector $\theta=(\sigma_0,\sigma)$.

The hyper-parameter θ can be trained by applying the initial (x_0,U_0), boundary (x_n^b,u_n^b) and already collected training data (x_{n-1},u_{n-1}) and Negative Log Marginal Likelihood resulting from [7,8]

$$\begin{bmatrix}u_n^b\\ u_{n-1}^b\end{bmatrix}\sim\mathcal{N}(0,K),$$

where

$$K=\begin{bmatrix}k_{n,n}(x_n^b,x_n^b) & k_{n,n-1}(x_n^b,x_{n-1})\\ & k_{n-1,n-1}(x_{n-1},x_{n-1})\end{bmatrix}.$$

To predict approximation at new point x_n^*, the following conditional distribution can be used

$$u_n(x_n^*)|\begin{bmatrix}u_n^b\\ u_{n-1}\end{bmatrix}\sim\mathcal{N}\left(q^TK^{-1}\begin{bmatrix}u_n^b\\ u_{n-1}\end{bmatrix},k_{n,n}(x_n^*,x_n^*)-q^TK^{-1}q\right),$$

where

$$q^T=\begin{bmatrix}k_{n,n}(x_n^*,x_n^b) & k_{n,n-1}(x_n^*,x_{n-1})\end{bmatrix}.$$

It is known that linear operations on GP give again GP and thus, taking into account the Euler scheme (2) together with GP prior assumption (3) allows to conclude that u_n and u_{n-1} are jointly Gaussian with the following GP [4,6–9]

$$\begin{bmatrix}u_n\\ u_{n-1}\end{bmatrix}\sim\mathcal{GP}\left(0,\begin{bmatrix}k_{n,n} & k_{n,n-1}\\ & k_{n-1,n-1}\end{bmatrix}\right),$$

where covariance functions are defined using the (4):

$$\begin{aligned}k_{n,n}&=k,\\ k_{n,n-1}&=k-a\tau\,\frac{d^2}{dx'^2}\,k-\tau f_n(x'),\\ k_{n-1,n-1}&=k-a\tau\,\frac{d^2}{dx'^2}\,k-\tau f_n(x')-a\tau\,\frac{d^2}{dx^2}\,k-\tau f_n(x)+a^2\tau^2\,\frac{d^2}{dx^2}\,\frac{d^2}{dx'^2}\,k-a\tau^2 f_n(x').\end{aligned}$$

For the test experiment we chouse the right-hand side of problem (1) in such a way that the exact solution is $U(x,t)=-\exp(-0.01\pi t)\sin(\pi x)$ with the initial solution $U_0(x)=-\sin(\pi x)$.

Figures 3 show a pretty good agreement between numerical and exact solutions for different time values.

In the end, let us note that our future work is aimed to apply the mentioned methodology to the PDEs with nonlinear diffusion coefficients as well as for the spatial multi-dimensional cases.

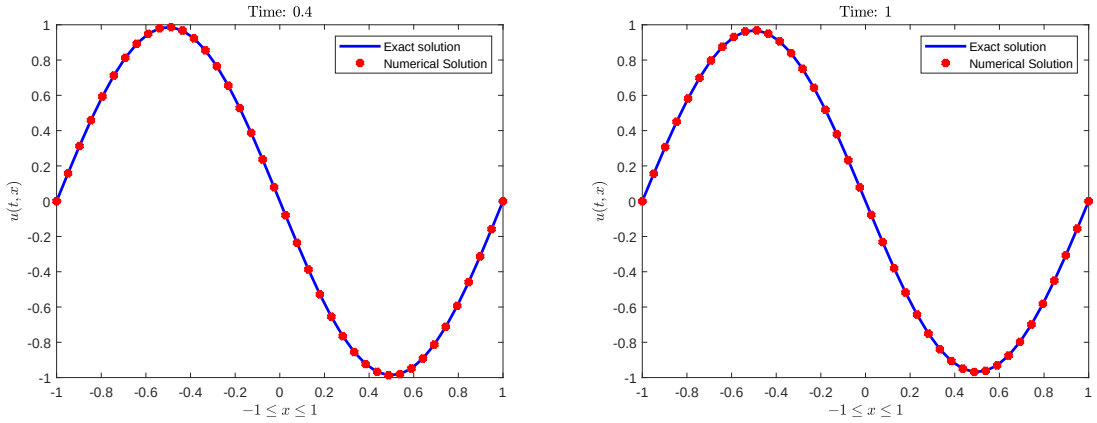

Figure 3. Exact and Numerical solutions at $t = 0.4$ and 1.

References

[1] E. Brochu, V. M. Cora and N. de Freitas, A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010; `https://arxiv.org/abs/1012.2599`.

[2] T. Jangveladze, Investigation and numerical solution of nonlinear partial differential and integro-differential models based on system of Maxwell equations. *Mem. Differ. Equ. Math. Phys.* **76** (2019), 1–118.

[3] T. Jangveladze, Z. Kiguradze and B. Neta, *Numerical Solutions of Three Classes of Nonlinear Parabolic Integro-Differential Equations*. Elsevier/Academic Press, Amsterdam, 2016.

[4] Z. Kiguradze, A Bayesian optimization approach for selecting the best parameters for weighted finite difference scheme corresponding to heat equation. *Abstracts of the International Workshop on the Qualitative Theory of Differential Equations – QUALITDE-2019, Tbilisi, Georgia, December 7-9*, pp. 108–111;
http://www.rmi.ge/eng/QUALITDE-2019/Kiguradze_Z_workshop_2019.pdf.

[5] Z. Kiguradze, J. He, B. Mutnury, A. Chada and J. Drewniak, Bayesian Optimization for Stack-up Design. *2019 IEEE International Symposium on Electromagnetic Compatibility, Signal Integrity and Power Integrity (EMC+SIPI)*, New Orleans, LA, (2019), 629–634.

[6] K. P. Murphy, *Machine Learning: a Probabilistic Perspective*. MIT press, 2012.

[7] M. Raissi, P. Perdikaris and G. Em. Karniadakis, Machine learning of linear differential equations using Gaussian processes. *J. Comput. Phys.* **348** (2017), 683–693.

[8] M. Raissi, P. Perdikaris and G. Em. Karniadakis, Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM J. Sci. Comput.* **40** (2018), no. 1, A172–A198.

[9] C. E. Rasmussen and Ch. K. I. Williams, *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2006.