# Learning Curves for Gaussian Processes via Numerical Cubature Integration

Simo Särkkä

Department of Biomedical Engineering and Computational Science
Aalto University, Finland
simo.sarkka@tkk.fi

**Abstract.** This paper is concerned with estimation of learning curves for Gaussian process regression with multidimensional numerical integration. We propose an approach where the recursion equations for the generalization error are approximately solved using numerical cubature integration methods. The advantage of the approach is that the eigenfunction expansion of the covariance function does not need to be known. The accuracy of the proposed method is compared to eigenfunction expansion based approximations to the learning curve.

**Keywords:** Gaussian process regression, learning curve, numerical cubature

## 1 Introduction

Gaussian process (GP) regression [1, 2] refers to a Bayesian machine learning approach, where instead of using a fixed form parametric model such as a MLP neural network [3] one postulates a Gaussian process prior over the model functions. Learning in Gaussian process regression means computing the posterior Gaussian process, which is conditioned to observed measurements. The prediction of unobserved values amounts to computing predictive distributions and their statistics.

This paper is concerned with approximate computation of learning curves for Gaussian process regression. By learning curve we mean the average generalization error $\epsilon(n)$ as function of the number of training samples $n$. A common way to compute approximations to the learning curves is to express the approximate average learning curve or its bounds in terms of the eigenvalues of the covariance function [4–8]. Upper and lower bounds for one-dimensional covariance functions, in terms of spectral densities and eigenvalues have been presented in [9]. One possible approach is to express the lower bound for the learning curve in terms of the equivalent kernel [10], which leads to similar results as the classical error bounds for Gaussian processes (see, e.g., [11, 12]). Statistical physics based approximations to GP learning curves have been considered in [13, 14].

In this paper we shall follow the ideas presented in [5, 8], but instead of using the eigenfunction expansion, we approximate the integrals over the training and test inputs with multidimensional numerical integration. The advantage of the

approach is that the learning curve can be evaluated without the knowledge of the eigenfunctions and eigenvalues of the covariance function. In the numerical integration methods, we shall specifically consider application of multidimensional generalizations of Gauss-Hermite quadratures, that is, Gauss-Hermite cubatures for computation of the multidimensional integrals. The usage of such numerical cubature rules has also recently gained attention in context of nonlinear Kalman filtering and smoothing [15–18].

## 2 Recursion for Learning Curve

Consider the following Gaussian process regression model:

$$\begin{aligned} f(\mathbf{x}) &\sim \mathcal{GP}(0, C(\mathbf{x}, \mathbf{x}')) \\ y_k &= f(\mathbf{x}_k) + r_k, \end{aligned} \tag{1}$$

where $y_k$, $k = 1, 2, \ldots, n$ are the measurements, $r_k \sim \mathrm{N}(0, s^2)$ is the IID measurement error sequence, and the input is $\mathbf{x} \in \mathbb{R}^d$. That is, the unknown function $f(\mathbf{x})$ is modeled as a zero mean Gaussian process with the given covariance function $C(\mathbf{x}, \mathbf{x}')$. Here we shall assume that both the function $f(\mathbf{x})$ and the measurements $y_k$ are scalar valued, but the extension to vector case is straightforward. We shall also assume that the prior Gaussian process has zero mean for notational convenience.

Given $n$ measurements $\mathbf{y} = (y_1, \ldots, y_n)$ at input positions $\mathbf{x}_{1:n} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ the posterior mean and covariance functions of $f$ are given as [1, 2]:

$$\begin{aligned} m^{(n)}(\mathbf{x}) &= C(\mathbf{x}, \mathbf{x}_{1:n}) \left[C(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) + s^2\, \mathbf{I}\right]^{-1} \mathbf{y} \\ C^{(n)}(\mathbf{x}, \mathbf{x}') &= C(\mathbf{x}, \mathbf{x}') - C(\mathbf{x}, \mathbf{x}_{1:n}) \left[C(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) + s^2\, \mathbf{I}\right]^{-1} C^T(\mathbf{x}', \mathbf{x}_{1:n}). \end{aligned} \tag{2}$$

For the purposes of estimating the learning curves, we shall assume that the input positions in the training set $\mathbf{x}_k$ are random, and form an IID process $\mathbf{x}_1, \ldots, \mathbf{x}_n$ such that $\mathbf{x}_k \sim p(\mathbf{x})$. If we assume that the test inputs have the distribution $\mathbf{x} \sim p^*(\mathbf{x})$, we obtain the following well known expression for the average generalization error of the Gaussian process:

$$\epsilon(n) = \left\langle C(\mathbf{x}, \mathbf{x}) - C(\mathbf{x}, \mathbf{x}_{1:n}) \left[C(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) + s^2\, \mathbf{I}\right]^{-1} C^T(\mathbf{x}, \mathbf{x}_{1:n}) \right\rangle, \tag{3}$$

where the expectation is taken over both the training and test input positions $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p(\cdot)$ and $\mathbf{x} \sim p^*(\cdot)$, respectively. Note that the error is no longer function of the measurements $y_1, \ldots, y_n$, nor the input positions.

This Gaussian process regression solution (2) can also be equivalently written in the following recursive form:

– **Initialization:** At initial step we have

$$\begin{aligned} m^{(0)}(\mathbf{x}) &= 0 \\ C^{(0)}(\mathbf{x}, \mathbf{x}') &= C(\mathbf{x}, \mathbf{x}'). \end{aligned} \tag{4}$$

– **Update:** At each measurement we perform the following update step:

$$\begin{aligned} m^{(k+1)}(\mathbf{x}) &= m^{(k)}(\mathbf{x}) + \frac{C^{(k)}(\mathbf{x}, \mathbf{x}_k)}{C^{(k)}(\mathbf{x}_k, \mathbf{x}_k) + s^2} \left(y_k - m^{(k)}(\mathbf{x})\right) \\ C^{(k+1)}(\mathbf{x}, \mathbf{x}') &= C^{(k)}(\mathbf{x}, \mathbf{x}') - \frac{C^{(k)}(\mathbf{x}, \mathbf{x}_k)\, C^{(k)}(\mathbf{x}', \mathbf{x}_k)}{C^{(k)}(\mathbf{x}_k, \mathbf{x}_k) + s^2}. \end{aligned} \tag{5}$$

The result at step $k = n$ will then be exactly the same as given by the equations (2). This recursion can be seen as a special case of the update step of infinite-dimensional distributed parameter Kalman filter (see, e.g., [19, 20]) with a trivial dynamic model.

Using these recursions equations, we can now write down the formal recursion formula for the covariance function, which is averaged over $n$ training inputs as follows:

$$\hat{C}^{(k+1)}(\mathbf{x}, \mathbf{x}') = C^{(k)}(\mathbf{x}, \mathbf{x}') - \int_{\mathbb{R}^d} \frac{C^{(k)}(\mathbf{x}, \mathbf{x}_k)\, C^{(k)}(\mathbf{x}', \mathbf{x}_k)}{C^{(k)}(\mathbf{x}_k, \mathbf{x}_k) + s^2}\, p(\mathbf{x}_k)\, d\mathbf{x}_k. \tag{6}$$

In this article, we shall follow [8] and ignore the dependence from the inputs before the previous step and approximate this as

$$\hat{C}^{(k+1)}(\mathbf{x}, \mathbf{x}') = \hat{C}^{(k)}(\mathbf{x}, \mathbf{x}') - \int_{\mathbb{R}^d} \frac{\hat{C}^{(k)}(\mathbf{x}, \mathbf{x}_k)\, \hat{C}^{(k)}(\mathbf{x}', \mathbf{x}_k)}{\hat{C}^{(k)}(\mathbf{x}_k, \mathbf{x}_k) + s^2}\, p(\mathbf{x}_k)\, d\mathbf{x}_k. \tag{7}$$

The approximation to the average generalization error is then given as

$$\epsilon(n) = \int_{\mathbb{R}^d} \hat{C}^{(n)}(\mathbf{x}, \mathbf{x})\, p^*(\mathbf{x})\, d\mathbf{x}. \tag{8}$$

## 3 Eigenfunction Expansion Approximation of Recursion

As done in [8], we can use the eigenfunction expansion method for solving the approximate average generalization error as follows. By Mercer's theorem the input averaged kernel $\hat{C}^{(k)}(\mathbf{x}, \mathbf{x}')$ has the eigenfunction expansion

$$\hat{C}^{(k)}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i^{(k)}\, \phi_i(\mathbf{x})\, \phi_i(\mathbf{x}'), \tag{9}$$

where $\phi_i(\mathbf{x})$ and $\lambda_i^{(k)}$ are the orthonormal set of eigenfunctions and eigenvalues of the kernel such that

$$\lambda_i^{(k)}\, \phi_i(\mathbf{x}) = \int_{\mathbb{R}^d} \hat{C}^{(k)}(\mathbf{x}, \mathbf{x}')\, \phi_i(\mathbf{x}')\, p(\mathbf{x}')\, d\mathbf{x}'. \tag{10}$$

Substituting the series into the recursion (7) now gives

$$\begin{aligned} \hat{C}^{(k+1)}(\mathbf{x}, \mathbf{x}') &= \sum_i \lambda_i^{(k)}\, \phi_i(\mathbf{x})\, \phi_i(\mathbf{x}') \\ &\quad - \int_{\mathbb{R}^d} \left\{ \frac{\left[\sum_i \lambda_i^{(k)} \phi_i(\mathbf{x})\, \phi_i(\mathbf{x}_k)\right] \left[\sum_j \lambda_j^{(k)} \phi_j(\mathbf{x}')\, \phi_j(\mathbf{x}_k)\right]}{\left[\sum_i \lambda_i^{(k)}\, \phi_i(\mathbf{x}_k)\, \phi_i(\mathbf{x}_k)\right] + s^2} \right\} p(\mathbf{x}_k)\, d\mathbf{x}_k. \end{aligned} \tag{11}$$

If we approximate the latter integral by taking expectations separately in denominator and numerator, then by the orthonormality properties of the eigenfunctions this reduces to:

$$\hat{C}^{(k+1)}(\mathbf{x}, \mathbf{x}') = \sum_i \left( \lambda_i^{(k)} - \frac{\left[\lambda_i^{(k)}\right]^2}{\sum_j \lambda_j^{(k)} + s^2} \right) \phi_i(\mathbf{x})\, \phi_i(\mathbf{x}'), \tag{12}$$

which implies that $\hat{C}^{(k+1)}(\mathbf{x}, \mathbf{x}')$ also has an eigenfunction expansions in terms of the same eigenfunctions. If we denote the coefficients as $\lambda_i^{(k+1)}$, then the approximate recursion equation for the coefficients is given as

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} - \frac{\left[\lambda_i^{(k)}\right]^2}{\sum_j \lambda_j^{(k)} + s^2} \tag{13}$$

If we have $p^*(\mathbf{x}) = p(\mathbf{x})$, then the approximation (8) to the average generalization error now reduces to [8]

$$\epsilon_D(n) = \sum_i \left( \lambda_i^{(n)} - \frac{\left[\lambda_i^{(n)}\right]^2}{\sum_j \lambda_j^{(n)} + s^2} \right). \tag{14}$$

We could then proceed to use further approximations by considering $n$ as continuous, which would lead to UC and LC approximations [8]:

- The upper continuous (UC) approximation has the form

$$\epsilon_{UC}(n) = s^2 \sum_i \frac{\lambda_i}{n'\,\lambda_i + s^2}, \tag{15}$$

  where $\lambda_i$ are the eigenvalues of the prior covariance function, and the effective number of training examples $n'$ is the solution to the self-consistency equation

$$n' + \sum_i \ln\left(1 + s^{-2}\, n' \lambda_i\right) = n. \tag{16}$$

- The lower continuous (LC) approximation is the solution to the self-consistency equation

$$\epsilon_{LC}(n) = s^2 \sum_i \frac{\lambda_i}{n'\,\lambda_i + s^2}, \tag{17}$$

  where $n' = s^2\, n/[s^2 + \epsilon_{LC}(n)]$.

## 4 Numerical Cubature Approximation of Recursion

Cubature integration refers to methods for approximate computation of integrals of the form

$$\mathrm{E}[g(\mathbf{x})] = \int_{\mathbb{R}^d} g(\mathbf{x})\, p(\mathbf{x})\, d\mathbf{x}, \tag{18}$$

where $p(\mathbf{x})$ is some fixed weight function. In particular, cubature integration methods here primarily refer to multidimensional generalizations of Gaussian quadratures, that is, to approximations of the form

$$\mathrm{E}[g(\mathbf{x})] \approx \sum_i W^{(i)}\, g(\mathbf{x}^{(i)}), \tag{19}$$

where the weights $W^{(i)}$ and the evaluation points $\mathbf{x}^{(i)}$ are (known) functionals of the weight function $p(\mathbf{x})$. In particular, when $p(\mathbf{x})$ is a multidimensional Gaussian distribution, we can use multidimensional Gauss-Hermite cubatures or more efficient spherical cubature rules (see, e.g., [21, 16, 17]). However, because here we need quite high order rules and construction of such efficient higher order spherical rules is quite complicated task, here we have used simpler Cartesian product based Gauss-Hermite cubature rules.

We can now use a multidimensional cubature approximation to the integral in Equation (7) which leads to the following:

$$\hat{C}^{(k+1)}(\mathbf{x}, \mathbf{x}') = \hat{C}^{(k)}(\mathbf{x}, \mathbf{x}') - \sum_i W^{(i)}\, \frac{\hat{C}^{(k)}(\mathbf{x}, \mathbf{x}^{(i)})\, \hat{C}^{(k)}(\mathbf{x}', \mathbf{x}^{(i)})}{\hat{C}^{(k)}(\mathbf{x}^{(i)}, \mathbf{x}^{(i)}) + s^2}, \tag{20}$$

where the weights $W^{(i)}$ and sigma points $\mathbf{x}^{(i)}$ correspond to integration over the training set distribution $p(\mathbf{x})$. For arbitrary $\mathbf{x}$ and $\mathbf{x}'$ we thus may now run the recursion (7), apply the above approximation on each step and get an approximation to $\hat{C}^{(n)}(\mathbf{x}, \mathbf{x}')$. Analogously, we can now form approximation to the average generalization error in Equation (8) as follows:

$$\int_{\mathbb{R}^d} \hat{C}^{(n)}(\mathbf{x}, \mathbf{x})\, p^*(\mathbf{x})\, d\mathbf{x} \approx \sum_j W^{*(j)}\, \hat{C}^{(n)}(\mathbf{x}^{*(j)}, \mathbf{x}^{*(j)}), \tag{21}$$

where the weights $W^{*(i)}$ and sigma points $\mathbf{x}^{*(i)}$ correspond to integration over the test set distribution $p^*(\mathbf{x})$. The computation of the latter integral can now be done by evaluating the former quadrature based approximation (20) at the quadrature points of the latter integral, that is, at $\mathbf{x} = \mathbf{x}' = \mathbf{x}^{*(j)}$. Note that this procedure might underestimate the generalization error slightly, because the sigma points for the train and test sets are in the same positions. It would be possible to use different sigma points for train and test sets, but then the computation would be slightly more complicated.

We can now compute simple approximation to the learning curve by assuming that $p^*(\mathbf{x}) = p(\mathbf{x})$ and by using the same cubature rule for train and test sets. This leads to a single set of sigma points $\mathbf{x}^{(i)} = \mathbf{x}^{*(i)}$ and weights $W^{(i)} = W^{*(i)}$. The algorithm can be implemented as follows:

– Initialize the elements of matrix $\mathbf{P}^{(0)}$ as follows:

$$P_{ii'}^{(0)} = C(\mathbf{x}^{(i)}, \mathbf{x}^{(i')}). \tag{22}$$

– for $n = 1, \ldots, N$ do

$$\mathbf{P}^{(n)} = \mathbf{P}^{(n-1)} - \sum_i W^{(i)} \frac{\mathbf{P}_{*i}^{(n-1)} \mathbf{P}_{i*}^{(n-1)}}{P_{ii}^{(n-1)} + s^2}, \tag{23}$$

where $\mathbf{P}_{*i}$ denotes the $i$th column of $\mathbf{P}$ and $\mathbf{P}_{i*}$ denotes the $i$th row.

– The approximate learning curve is given as

$$\epsilon_C(n) = \sum_i W^{(i)} P_{ii}^{(n)}. \tag{24}$$

## 5 Numerical Comparison

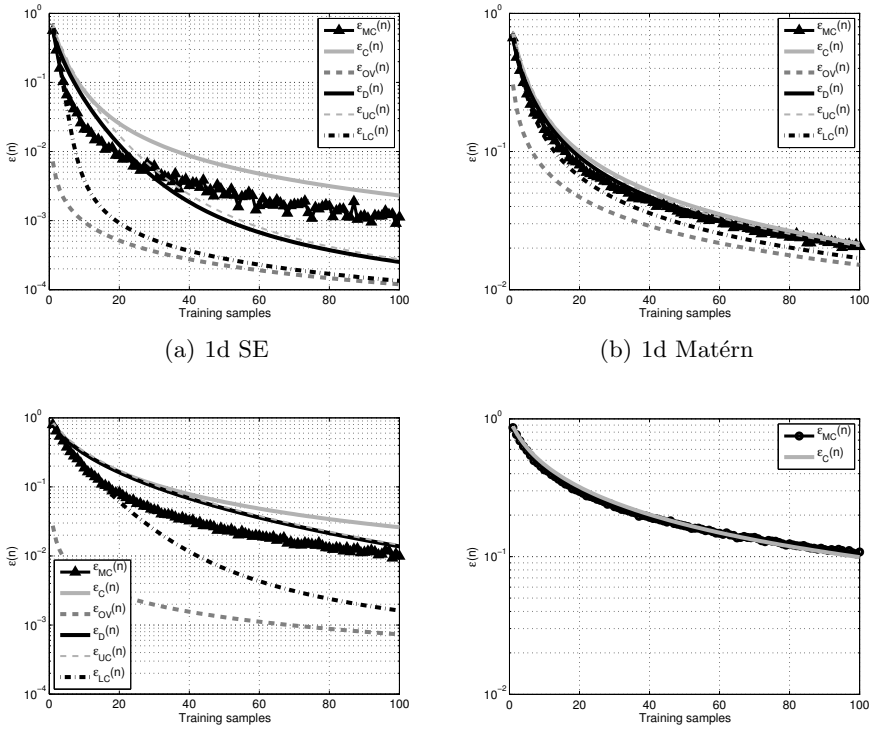
We tested the error bounds presented in this article using 1d and 2d squared exponential (SE) covariance functions $\exp(-|x - x'|^2/(2l^2))$ and with Matérn covariance function $(1+\sqrt{3}|x-x'|/l)\exp(-\sqrt{3}|x-x'|/l))$. For SE covariance we used the parameters values $l = 1$, $\sigma^2 = 10^{-3}$. The parameters for the Matérn covariance were selected to be $l = 1$, $\sigma^2 = 0.1$. The input and test sets were assumed to have a zero mean unit Gaussian distribution, for which the weights $W^{(i)}$ and evaluation points $x^{(i)}$ can be obtained by using existing methods.

In addition to the bounds $\epsilon_D(n)$ defined in Equation (14), $\epsilon_{UC}(n)$ in (15), $\epsilon_{LC}(n)$ in (17),and the proposed bound $\epsilon_C(n)$ in (24), we also compared to the following well known Opper-Vivarelli (OV) bound [5]:

$$\epsilon_{OV}(n) = s^2 \sum_i \frac{\lambda_i}{n\,\lambda_i + s^2}. \tag{25}$$

For the SE covariance functions we used the known closed form formulas for the eigenvalues, in the 1d Matérn case we computed the eigenvalues numerically. In the 2d Matérn case the eigenvalues were not available, because the eigenvalue problem became too big to be solved with the required numerical accuracy. We used 60th order Gauss-Hermite quadrature for the 1d $\epsilon_C(n)$ calculations and 20th order Gauss-Hermite product-rule cubature for the 2d $\epsilon_C(n)$ calculations. For all the cases, we also computed approximation to the 'true' generalization error curve $\epsilon_{MC}(n)$ using Monte Carlo method with 100 independent training sets for each training set size 1–100, and the generalization error was estimated with test sets of size 100, which were drawn independently for each MC sample.

The learning curves computed using different approximations are shown in Figure 1. As can be seen in the figures, in the 1d and 2d SE cases the proposed approximation $\epsilon_C(n)$ overestimates the error, but is still much better than $\epsilon_{OV}(n)$ and its relative accuracy is close to the other methods. In the 1d and 2d Matérn cases the proposed approximation is very accurate. The overall performance of the proposed method is very good given that it does not need the eigenvalues of the covariance function at all.




(a) 1d SE (b) 1d Matérn

(c) 2d SE (d) 2d Matérn

**Fig. 1.** Learning curves for squared exponential (SE) and Matérn covariance functions with input dimensions 1 and 2.

## 6 Conclusion

In this article we have presented a new cubature integration based method for approximate computation of learning curves in Gaussian process regression. The advantage of the method is that it does not require availability of eigenvalues of the covariance function unlike most of the alternative methods. The accuracy of the method was numerically compared to previously proposed eigenfunction expansion based methods and the propoposed approach seems to give good approximations to learning curves especially in the case of Matérn covariance function.

**Acknowledgments.** The author is grateful to the Centre of Excellence in Computational Complex Systems Research of Academy of Finland for the financial support and also likes to thank Aki Vehtari for helpful comments on the manuscript.

## References


1. O'Hagan, A.: Curve fitting and optimal design for prediction (with discussion). Journal of the Royal Statistical Society B 40(1), 1–42 (1978)
2. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press (2006)
3. Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)
4. Opper, M.: Regression with Gaussian processes: Average case performance. In: Theoretical Aspects of Neural Computation. Springer-Verlag (1997)
5. Opper, M., Vivarelli, F.: General bounds on bayes errors for regression with gaussian processes. In: NIPS 11, pp. 302–308. The MIT Press (1999)
6. Sollich, P.: Approximate learning curves for Gaussian processes. In: Proceedings of ICANN'99, pp. 437–442. (1999)
7. Sollich, P.: Learning curves for Gaussian processes. In: NIPS 11. The MIT Press (1999)
8. Sollich, P., Halees, A.: Learning curves for Gaussian process regression: Approximations and bounds. Neural Computation 14, 1393–1428 (2002)
9. Williams, C.K.I., Vivarelli, F.: Upper and lower bounds on the learning curve for Gaussian processes. Machine Learning 40, 77–102 (2000)
10. Sollich, P., Williams, C.: Using the equivalent kernel to understand Gaussian process regression. In: NIPS 17, pp. 1313–1320. MIT Press (2005)
11. Van Trees, H.L.: Detection, Estimation, and Modulation Theory Part I. John Wiley & Sons, New York (1968)
12. Papoulis, A.: Probability, Random Variables, and Stochastic Processes. McGraw-Hill (1984)
13. Malzahn, D., Opper, M.: Learning curves for Gaussian process regression: A framework for good approximations. In: NIPS 13. The MIT Press (2001)
14. Malzahn, D., Opper, M.: A variational approach to learning curves. In: NIPS 14. The MIT Press (2002)
15. Ito, K., Xiong, K.: Gaussian filters for nonlinear filtering problems. IEEE Transactions on Automatic Control 45, 910–927 (2000)
16. Wu, Y., Hu, D., Wu, M., Hu, X.: A numerical-integration perspective on Gaussian filters. IEEE Transactions on Signal Processing 54, 2910–2921 (2006)
17. Arasaratnam, I., Haykin, S.: Cubature Kalman filters. IEEE Transactions on Automatic Control 54, 1254–1269 (2009)
18. Särkkä, S., Hartikainen, J.: On Gaussian optimal smoothing of non-linear state space models. IEEE Transactions on Automatic Control 55, 1938–1941 (2010)
19. Curtain, R.: A survey of infinite-dimensional filtering. SIAM Review 17(3), 395–411 (1975)
20. Ray, W.H., Lainiotis, D.G.: Distributed Parameter Systems. Dekker (1978)
21. Cools, R.: Constructing cubature formulae: The science behind the art. In: Acta Numerica. Volume 6, pp. 1–54. Cambridge University Press (1997)