Stochastic Approximation Using Neural Processes

Name Jimmy Woo
Date May 14, 2023

Abstract

Gaussian processes (GP) are widely used as surrogate models to simulation optimization tasks where the analytical solutions are unavailable. It has previously been shown that GPs can be used in conjunction with stochastic approximation (SA) to improve the convergence rate while maintaining theoretical guarantees of local convergence using the *stochastic approximation with Gaussian process regression* (SAwGP) algorithm [13]. In this paper, we present the *stochastic approximation with neural process model* (SAwNP) algorithm, where we demonstrate the use of pre-trained neural processes [5][7] in place of the GP in the SAwGP framework, and empirically show that the convergence rate can be further improved.

1 Introduction

Stochastic approximation (SA) is one of the most widely used methods in simulation optimization, due to its applicability to a wide range of problems and its asymptotic convergence rates being independent of the dimensionality of the data under certain conditions [3]. Using the Kiefer-Wolfowitz (KW) method [6] to compute finite difference gradient estimates, the optimization problem can be solved with asymptotic convergence rate of $O\left(n^{-1/3}\right)$. However, the number of required simulation replications grows linearly with the dimensions of the input data, and thus obtaining these symmetric differences for a high-dimensional vector in a naive way can be costly. One potential method to mitigate this issue is use surrogate-based optimization, where the optimum values obtained by approximating the objective function using surrogates. Gaussian processes (GP) are a popular choice of surrogates, due to the fact that they can provide a probabilistic estimate of the behaviour of the objective function and provide a measure of uncertainty. Yan et al. [13] presents the *stochastic approximation with Gaussian process regression* (SAwGP) algorithm, where the KW method is used to optimize the objective function, and samples accumulated from SA is used to fit a GP and accelerate the convergence to the optimum state. It has been empirically shown on various functions of known analytical form that SAwGP can outperform naive SA and converge to a more optimal state faster. However, GPs require computing the inverse of the covariance matrix, and thus require $O(n^3)$ computations to fit a GP with n samples. Similar to GPs, neural processes (NP) represent a distribution over functions rather than a single function, and provide uncertainty measurements of predictions. However, the runtime of a single forward pass of an NP model is only $O(n+m)$, where n and m are the number of context and target points respectively. Therefore, in cases where fast online optimization is required, pre-trained NPs can be a more efficient option. Kim et al. [7] demonstrate that neural processes have a tendency to underfit, and present attentive neural processes (ANP) that uses multi-head self-attention[12] as its aggregator function.

In this paper, we propose the SAwNP algorithm, that uses pre-trained ANPs in place of GPs to guide the SA convergence. Similar to the SAwGP evaluation, we focus on the KW type SA where the unbiased gradient estimator is unavailable. For convenience purposes ANPs will be referred to as NPs, to refer to the *neural process family* [1] of models.

The remainder of this paper is organized as follows: Section 2 briefly introduces the background information on SA, GPs, NPs, and SAwGP, Section 3 describes the SAwNP algorithm, Section 4 provides the empirical results,

and Section 5 provides conclusions, limitations of the proposed algorithm, and suggested future work.

2 Background

2.1 Stochastic Approximation

In SA, given some performance measure of interest $f(\mathbf{x})$, the objective is to find the optimal parameters $\mathbf{x}^* \in \mathbb{R}^d$ such that

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Theta} f(\mathbf{x})$$

where $\mathbf{x}^*$ corresponds to a unique local optimum, and Θ corresponds to the possible state space for a minimization problem. SA is a recursive algorithm that converges to the optimum value using the following update equation

$$\mathbf{x}_{n+1} = \Pi_\Theta\left(\mathbf{x}_n - a_n \hat{\nabla}_{\mathbf{x}_n} f(\mathbf{x}_n)\right)$$

where $\mathbf{x}_i$ are the i^{th} input values, $\Pi_\Theta(\mathbf{x}_i)$ the projection of $\mathbf{x}_i$ in the feasible region, a_i is a sequence such that $a_i \to 0$, $\sum_{i=1}^{\infty} = \infty$, and $\hat{\nabla}_{\mathbf{x}_n} f(\mathbf{x}_n)$ is the estimate of the gradient of the performance measure $f(\mathbf{x}_n)$ with respect to its input $\mathbf{x}_n$. The symmetric finite difference gradient estimate in KW type SA can be computed as

$$\hat{\nabla}_{\mathbf{x}_n} f(\mathbf{x}_n) = \left(\frac{\partial \hat{f}(\mathbf{x}_n)}{\partial x_n^{(1)}}, \cdots, \frac{\partial \hat{f}(\mathbf{x}_n)}{\partial x_n^{(d)}}\right)^T$$
$$\frac{\partial \hat{f}(\mathbf{x}_n)}{\partial x_n^{(i)}} = \frac{Y(\mathbf{x}_n + c_n\mathbf{e}_i) - Y(\mathbf{x}_n - c_n\mathbf{e}_i)}{2c_n}$$

where $\frac{\partial \hat{f}(\mathbf{x}_n)}{\partial x_n^{(i)}}$ is the estimate of the gradient of the performance measure with respect to the i^{th} dimension of $\mathbf{x}_n$, and c_i is a sequence such that $c_i \to 0$, $\sum_{i=1}^{\infty} a_i c_i < \infty$, $\sum_{i=1}^{\infty} a_n^2 c_n^{-2} < \infty$, and $\mathbf{e}_i$ is the d-dimensional vector of zeros, where only the i^{th} element is 1. Note that common random number (CRN) was used for each replication of the simulations for the symmetric finite difference gradient estimation for the implementation.

2.2 Gaussian Processes Regression

GPs represent a probability distribution over functions, where each function is a possible realization of a stochastic process. GPs are fully characterized by a mean function $\mu : \mathbb{X} \mapsto \mathbb{R}$, and a covariance fuction $K : \mathbb{X} \times \mathbb{X} \mapsto \mathbb{X}$. Note that all GPs implemented for the numerical experiments used the squared exponential kernel, defined as [2]

$$K(\mathbf{x}, \mathbf{x'}) = \tau^2 exp\left(-\alpha||\mathbf{x} - \mathbf{x}'||^2\right)$$

where τ and α are constants. For given sets $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\{y_1, \ldots, y_n\}$ with some noise $\epsilon \sim \mathtt{N}(0, \sigma^2)$, the prior $f \sim \mathtt{GP}(\mu, K)$ has a multivariate normal distribution with mean vector $\boldsymbol{\mu} = (\mu(\mathbf{x}_1, \ldots, \mu(\mathbf{x}_n))$ and a covariance matrix $\mathbf{K} = K(\mathbf{x}_i, \mathbf{x}_j)_{i,j=1}^n$. The posterior distribution of the Gaussian process regression (GPR) model can be characterized by

$$\mu_n(\mathbf{x}) = \mu(\mathbf{x}) + \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \Sigma)^{-1} (\mathbf{y} - \boldsymbol{\mu})$$
$$K_n(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \Sigma)^{-1} \mathbf{k}(\mathbf{x}')$$

where $\mathbf{k}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n))$, $\mathbf{y} = (y_1, \ldots, y_n)^T$, and Σ corresponds to the n by n diagonal covariance matrix corresponding to the noise[11]. In the case of surrogate modelling, the response surface estimate $\hat{f}(\mathbf{x})$ is simply the posterior mean function $\mu_n(\mathbf{x})$.

2.3 Stochastic Approximation with Gaussian Process Regression

SAwGP is an iterative algorithm that uses KW type SA to generate a set of samples to fit a GPR model. The initial step is similar to SA, but the main difference is that all traversed states and corresponding samples during symmetric finite differences computations are being stored in memory. After η iterations of SA, the stored state/sample pairs can be used to fit a GPR model. Using the fitted GPR model, the current best solution can be approximated. A point is sampled around a uniform ball with distance δ around the current best solution, and is added to the set of collected samples. At this point SA can start again using the current best solution, and this process can be repeated for a desired number of iterations. The local convergence properties of SAwGP have been shown to hold true [13], but in practice, the performance of SAwGP is highly dependent on the quality of the fitted GPR model, and the solver to compute the current best with each fitted GPR model. For the numerical experiments, limited-memory BFGS-B (L-BFGS-B) algorithm [14] was used for the bounded, non-constrained optimizations, and sequential quadratic programming (SQP) algorithm [9] was used for the constrained optimizations.

2.4 Neural Process Family

2.4.1 Neural Processes

Neural processes, parameterized by θ, learns a family of conditional distributions that maps input $x \in \mathbb{R}^d$ to output $y \in \mathbf{R}$ given some arbitrary number of observed context points $C = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$ (analogous to the training data in GPR). The deterministic version of NP, also known as conditional neural processes (CNP)[4], models this likelihood as

$$p_\theta(\mathbf{y}_T|\mathbf{x}_T; C) = \prod_{i=1}^{T} p_\theta\left(y^{(i)}|x^{(i)}; R\right)$$

where $x^{(i)}$ and $y^{(i)}$ correspond to the i^{th} input and output values of the target, and R is the aggregator that combines $\mathbf{x}_c$ and $\mathbf{y}_c$ values with permutation invariance with respect to the context set. The latent counterpart of CNP, also known as neural processes [5], introduces a global latent variable $\mathbf{z}$ that aggregates all of the context information, with the goal of modelling different realizations of stochastic processes. The likelihood $p_\theta(\mathbf{y}_T|\mathbf{x}_T; C)$ can now be expressed as

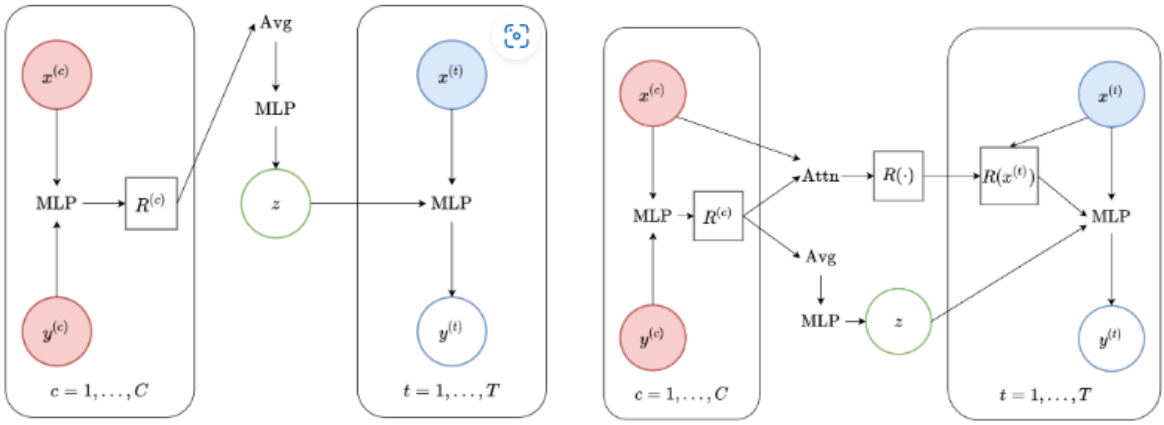
$$p_\theta(\mathbf{y}_T|\mathbf{x}_T; C) = \int p_\theta(\mathbf{y}_T|\mathbf{x}_T; \mathbf{z}) p_\theta(\mathbf{z}|\mathbf{x}_T; C) d\mathbf{z}$$

$$p_\theta(\mathbf{y}_T|\mathbf{x}_T; C) = \int p_\theta(\mathbf{z}|\mathbf{x}_T; C) \prod_{i=1}^{T} p_\theta\left(y^{(i)}|x^{(i)}; \mathbf{z}\right) d\mathbf{z}$$

where $p_\theta(\mathbf{z}|\mathbf{x}_T; C)$ and $p_\theta(\mathbf{y}_T|\mathbf{x}_T; \mathbf{z})$ are modelled by multilayer perceptron (MLP) encoder and decoder respectively. Refer to Figure 1 for a graphical representation of the NP model[1]. Using the reparametrization trick introduced with variational autoencoders (VAE) [8], these MLPs can be trained based on the evidence lower-bound (ELBO):

$$\log p_\theta(\mathbf{y}_T|\mathbf{x}_T; C) \geq \mathbb{E}_{z\sim p_\theta(\mathbf{z}|D)}\left[\log p_\theta(\mathbf{y}_T|\mathbf{x}_T, \mathbf{z}) - \mathrm{KL}\left(p_\theta\left(\mathbf{z}|D\right)||p_\theta\left(\mathbf{z}|C\right)\right)\right]$$

where D is the union of the context set and the target set. NPs achieve permutation invariance of the context set by using simple aggregator functions such as the mean function. In addition, for context and target sets with sizes n and m respectively, a forward pass of an NP scales $O(n+m)$, which is significantly better than GPs that scale $O((n+m)^3)$. One drawback to using NPs is that the permutation invariance comes at a cost of ignoring context consistency, and thus NPs tend to suffer from underfitting[7].



(a) (b)

Figure 1: (a) Computational graph of NP (b) Computational graph of ANP

2.5 Attentive Neural Processes

ANPs incorporate multi-head attention [12] into the NP architecture to mitigate the underfitting issue. For a given set of key-value pairs (k_i, v_i) and query q, the attention mechanism computes the similarities between the query and each of the keys to generate an aggregates these scores to form a single value corresponding to the query. Multi-head attention uses multiple instances of scaled dot-product attention (which computes similarities based on dot-products). Multi-head attention can be expressed as:

$$\text{Scaled Dot-Product Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}V\right)$$

$$\text{MultiHead}(Q, K, V) = \text{concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

$$\text{head}_i = \text{Scaled Dot-Product Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where W_i^Q, W_i^K, W_i^V, and W^O are parameter matrices. Refer to Figure 1 for a graphical representation of the ANP model[1]. ANP applies self-attention (i.e. keys identical to query) to the context points to learn a better representation of the inter-contextual interactions. Stacked self-attention mechanism in the architecture enables learning non-linear interactions. Cross-attention is used to replace the mean aggregation function of NPs, to better learn information on which context points are more relevant to generating a prediction for the current target. It has been empirically shown that ANPs can greatly improve the accuracy of predictions, and expand the range of functions that can be modelled compared to NPs. The increased performances comes at a cost of increased time complexity of $O(n(n+m))$.

3 Methodology

3.1 SAwNP Algorithm

The SAwNP algorithm differs from the SAwGP algorithm in two ways: SAwNP uses NPs instead of GPR models, and these NPs require training prior to the optimization process. The steps of the SAwNP algorithm are outlined in algorithm 1.

The training process of the NPs follow the training process originally outlined in Garnelo et al. [5]. For every epoch, randomly sample noisy estimates, sample the number of context points from a predefined uniform distribution, and use the remaining observations as target points. These steps are taken to ensure that the model learns different

realizations of a stochastic process. Refer to Table **??** in Appendix B for the hyperparameters of the NPs trained for each of the experiments.

Algorithm 1 SAwNP

Input: η, δ, sequences a_n and c_n, pre-trained `NeuralProcess` model, initial point $\mathbf{X}_{init}$, sample function Y

$n \leftarrow 1$, $r \leftarrow 1$, $\mathcal{X} \leftarrow \emptyset$, $\mathcal{Y} \leftarrow \emptyset$, $\mathbf{X}_1 \leftarrow \mathbf{X}_{init}$
for $i = 1, 2, \ldots$ **do**
  while $n \leq r * \eta$ **do**
    current_best $\leftarrow \mathbf{X}_n$
    $\mathbf{X}_{n+1} \leftarrow \texttt{SA\_iteration}(\mathbf{X}_n, a_n, c_n)$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{\mathbf{X}_n, \mathbf{X}_n + c_n\mathbf{e}_1, \ldots, \mathbf{X}_n + c_n\mathbf{e}_d, \mathbf{X}_n - c_n\mathbf{e}_1, \ldots, \mathbf{X}_n - c_n\mathbf{e}_d\}$
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{Y(\mathbf{X}_n), Y(\mathbf{X}_n + c_n\mathbf{e}_1), \ldots, Y(\mathbf{X}_n + c_n\mathbf{e}_d), Y(\mathbf{X}_n - c_n\mathbf{e}_1), \ldots, Y(\mathbf{X}_n - c_n\mathbf{e}_d)\}$
    $n \leftarrow n + 1$
    $\mathbf{x}_c \leftarrow \mathcal{X}$
    $\mathbf{y}_c \leftarrow \mathcal{Y}$
    $\mathbf{X}^* \leftarrow \underset{\mathbf{X} \in \mathbb{X}}{\arg\max}\, \texttt{NeuralProcess}(\mathbf{x}_c, \mathbf{y}, \mathbf{X})$
    $\mathbf{X}' \leftarrow \texttt{SampleUniformBall}(\mathbf{X}^*, \delta)$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{\mathbf{X}'\}$
    $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{Y(\mathbf{X}')\}$
    $\mathbf{X}_n \leftarrow \mathbf{X}^*$
  end while
end for

3.2 Evaluation

To observe the convergence rates of SA, SAwGP and SAwNP, two experiments were performed: optimization of a quadratic objective function with dimensions d =1, 2, 3, and the optimization of stochastic activity network (SAN) presented in Nelson and Pei [10] (2021).

The quadratic objective function is as follows [13]:

$$h(\mathbf{x}) = -0.05||\mathbf{x} - \mathbf{2}||^2 + 10, \ \mathbf{x} \in \mathbb{X} := [-10, 10]^d$$

where $\mathbf{2}$ denotes a d dimensional vector of 2's. The optimal value of this objective function is found at $\boldsymbol{\theta} = \mathbf{2}$, with the optimal value of 10. In each of the 3 cases, a noise term sampled from the standard normal distribution is added. For the the algorithm parameters, $\eta = 20$ and $\delta = 1$ are used.

The SAN optimization task is performed with the following parameters: $\tau_j, c_j = 1$, $l_j = 0.5$ for $j = 1, 2, \ldots, 5$, and $b = 1$. For the the algorithm parameters, $\eta = 10$ and $\delta = 1$ are used.

4 Results

4.1 Quadratic Objective Function

Figure 2 clearly indicates that for all 3 cases, SAwNP performs superior compared to SAwGP and SA. This behaviour is to be expected since the pre-trained NP has already seen large volumes of the data during training, and is able to accurately model the objective function once enough context points are provided. In all 3 cases, SAwNP consistently converges faster to the known optimal value compared to the other two algorithms. It is evident that the NP is accurately determining the "next optimal step" at each step consistently, since the half

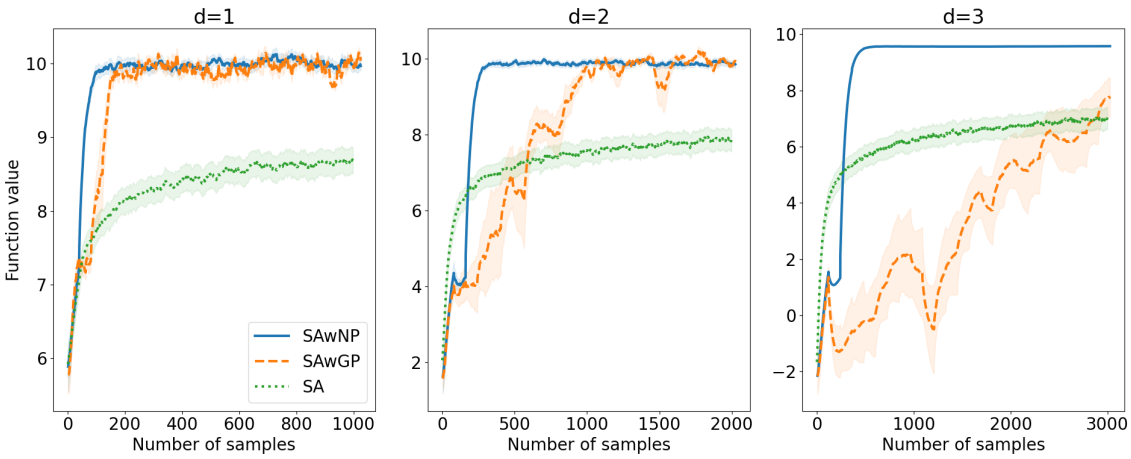


Figure 2: Convergence plot of the quadratic objective function averaged over 30 replications; shaded region represents 95% confidence interval

width of the 95% confidence interval is extremely narrow for all 3 cases. Furthermore, the performance of the algorithm seems to be consistent regardless of the increase in dimensions for this particular objective function.

4.2 Stochastic Activity Network Optimization

Figure 3 displays that neither SAwNP nor SAwGP performed better than the naive SA approach. SAwNP performed identical to SA for the SAN optimization task, due to the fact that after each iteration of the SA step, the trained NP model always estimated the last state visited by SA as the current best state, and thus provided no improvements. It is unlikely that the issue is related to the increased dimensionality, as the density of the data in this 5-dimensional data space is higher compared to the quadratic experiments. It is evident that this optimization task is relatively easier, as it only requires approximately 100 samples for SA to converge to the minimum value. Therefore, the fact that there are relatively fewer context points available at each inference step may attribute to the subpar performances of SAwNP and SAwGP. This is an indicator that SAwNP may be better suited for certain types of problems, and the choice should be carefully made based on the characteristics of the simulation task since there is a large overhead in generating simulation samples for the NP training process. One of the major benefits of using NPs instead of GPs is that the inference time is much faster, and thus SAwNP can be beneficial to use for cases where frequent online optimization is required.

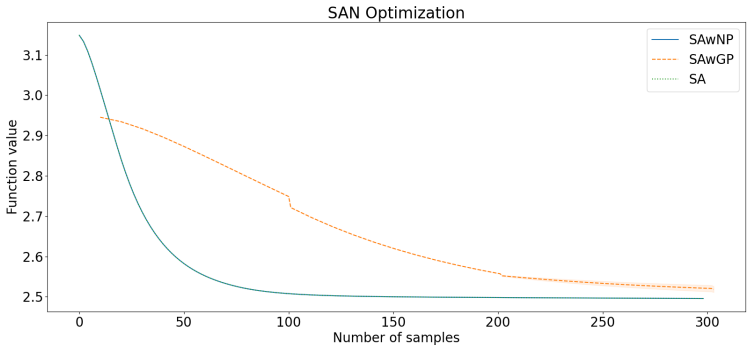


Figure 3: Convergence plot of the SAN optimization task averaged over 30 replications; shaded region represents 95% confidence interval

5 Conclusion

In this paper, we presentened SAwNP, which extends SAwGP by replacing the GPR models with NPs to further enhance the convergence rate. It has been empirically shown that SAwNP improves the convergence rate of the optimization task, but with a large overhead of pre-generating simulation samples to train the NP. There are few potential avenues for future research. A hybrid training scheme, or a fully online training scheme can be explored to minimize the simulation data required to train the NPs. Furthermore, SAwNP (and SAwGP) only use the estimated current best value for the optimization task. It may be beneficial to explore Bayesian optimization techniques to identify acquisition functions that can further improve the performances. Finally, further numerical experiments can be carried out to evaluate the performance of SAwNP on a wider range of simulation tasks of varying complexities to fully characterize the strengths and weaknesses of the algorithm.

References

[1] Yann Dubois, Jonathan Gordon, and Andrew YK Foong. Neural process family. `http://yanndubs.github.io/Neural-Process-Family/`, September 2020.

[2] Nicolas Durrande and Rodolphe Le Riche. Introduction to Gaussian Process Surrogate Models. Lecture, October 2017. URL `https://hal.science/cel-01618068`.

[3] Michael C. Fu. *Handbook of Simulation Optimization*. Springer Publishing Company, Incorporated, 2014. ISBN 1493913832.

[4] Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional neural processes. *CoRR*, abs/1807.01613, 2018. URL `http://arxiv.org/abs/1807.01613`.

[5] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes, 2018.

[6] J. Kiefer and J. Wolfowitz. Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466, 1952. doi: 10.1214/aoms/1177729392. URL `https://doi.org/10.1214/aoms/1177729392`.

[7] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes, 2019.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[9] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. URL `https://books.google.ca/books?id=4rKaGwAACAAJ`.

[10] Barry L. Nelson and Linda Pei. Foundations and methods of stochastic simulation. *International series in management science/operations research*, Jan 2021. doi: 10.1007/978-3-030-86194-0.

[11] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN 026218253X.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[13] Yingcui Yan, Haihui Shen, and Zhibin Jiang. Stochastic approximation with gaussian process regression. In *2021 Winter Simulation Conference (WSC)*, pages 1–12, 2021. doi: 10.1109/WSC52266.2021.9715329.

[14] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, dec 1997. ISSN 0098-3500. doi: 10.1145/279232.279236. URL `https://doi.org/10.1145/279232.279236`.