Active Learning Based Error Sampling for High-dimensional Nonlinear PDEs

Wenhan Gao[1,2](presentor), Chunmei Wang[2], Haizhao Yang[3]

1 Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY

2 Department of Mathematics and Statistics, Texas Tech University, Lubbock, TX

3 Department of Mathematics, Purdue University, West Lafayette, IN

Motivation

Most deep-learning-based solvers are computationally expensive; the convergence is usually slow and not guaranteed. In this paper, the main aim in is to address this issue by active-learning-based adaptive sampling techniques to sample the most informative training examples.

The Residual Model

Consider the following general form of PDE:

$$\begin{cases} \mathcal{D}u(\boldsymbol{x}) = f(\boldsymbol{x}) & \text{in } \Omega \\ \mathcal{B}u(\boldsymbol{x}) = g(\boldsymbol{x}) & \text{on } \partial\Omega \end{cases}$$

PDEs are solved via deep neural networks by directly minimizing the degree to which the network approximation violates the PDE and boundary/initial conditions:

$$\begin{aligned} \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) &:= \arg\min_{\boldsymbol{\theta}} \|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})\|_2^2 + \lambda\|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})\|_2^2 \\ &= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\in\Omega}\left[|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})|^2\right] + \lambda\mathbb{E}_{\boldsymbol{x}\in\partial\Omega}\left[|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})|^2\right] \\ &\approx \arg\min_{\boldsymbol{\theta}} \frac{1}{N_1}\sum_{i=1}^{N_1}|\mathcal{D}\phi(\boldsymbol{x}_i;\boldsymbol{\theta}) - f(\boldsymbol{x}_i)|^2 + \frac{\lambda}{N_2}\sum_{j=1}^{N_2}|\mathcal{B}\phi(\boldsymbol{x}_j;\boldsymbol{\theta}) - g(\boldsymbol{x}_j)|^2 \end{aligned}$$

, where λ is a positive hyper-parameter that weights the boundary loss, the last step is a Monte-Carlo approximation, and $x_i \in \Omega$, $x_j \in \partial\Omega$ are N_1 and N_2 allocation points, respectively.

Error Sampling

It is natural to wonder how to sample these points and whether each point is of the same importance for the model to minimize the empirical loss function. Therefore, error sampling is proposed in this work to preferentially choose allocation points with larger absolute residual errors. Intuitively, one can think of high residual error at a point as a proxy of the model being wrong to a greater extent at this point.

The fundamental methodology of error sampling is to choose from a biased importance distribution $q(x) \propto \mathcal{R}^p_{abs}(x)$that attaches higher priority to important volumes/regions of the domain:

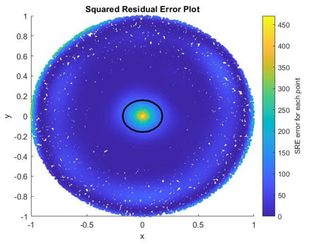
$$q(\boldsymbol{x}) = \frac{\mathcal{R}^p_{abs}(\boldsymbol{x})}{NC}$$

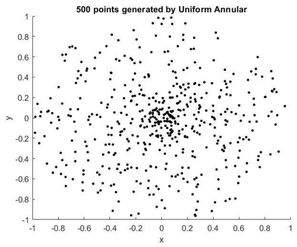
, where $NC = \int_\Omega \mathcal{R}^p_{abs}(x)dx$ is the normalizing constant that is unknown, p is a non-negative constant hyper-parameter that controls the effect of error sampling.

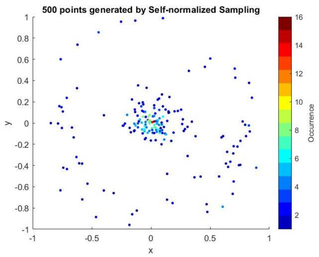
Sampling Techniques

It is impossible to directly sample points from $q(x)$. A few algorithms to simulate observations from $q(x)$ are presented in the paper. Also, a variant of error sampling that still has all the advantages of error sampling is also introduced in the paper; this variant might be a better fit for some PDEs.

Illustration of Sampling







Approach	Number of Points inside Ellipse
No Error Sampling	138
Error Sampling	348

Numerical Results

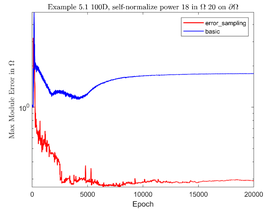
In this work, error sampling is tested on two PDEs examples, elliptic and parabolic equations. Testing includes various dimensions. First, elliptic equation(5.1):

$$\begin{aligned} -\nabla\cdot\left((1+\frac{1}{2}|\boldsymbol{x}|^2)\nabla u\right) + (\nabla u)^2 &= f(\boldsymbol{x}), \quad \text{in } \Omega := \{\boldsymbol{x} : |\boldsymbol{x}| < 1\} \\ u &= g(\boldsymbol{x}), \quad \text{on } \partial\Omega, \end{aligned}$$

, where $g(x) = 0$ and $f(x)$ is specified appropriately so that the exact solution is: $u(\boldsymbol{x}) = sin(\frac{\pi}{2}(1-|\boldsymbol{x}|)^{2.5})$

Results:

Dimension		Error Sampling	Basic Model
10 D	ℓ_2 error	9.121454e-03	2.482386e-02
	max modulus error	3.369123e-02	1.239435e-01
	Running Time in Seconds	8844.328335	7528.474081
20 D	ℓ_2 error	3.193670e-02	7.046980e-02
	max modulus error	1.083703e-01	2.838619e-01
	Running Time in Seconds	12554.326195	10129.247696
100 D	ℓ_2 error	1.142189e-01	5.174087e-01
	max modulus error	2.919715e-01	1.755006e+00
	Running Time in Seconds	52927.455038	40356.392521



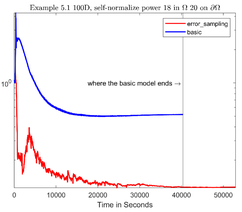
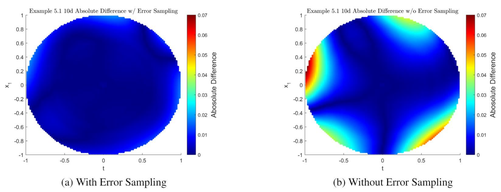
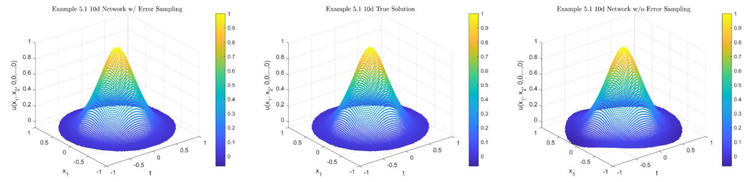


Figure 5.3: 5.1 10D $(x_1, x_2, 0, 0, ..., 0)$-surface Absolute Difference $|u - \phi|$



(a) With Error Sampling (b) Without Error Sampling

Figure 5.2: 5.1 10D $(x_1, x_2, 0, 0, ..., 0)$-surface of network solutions and the true solution



(a) With Error Sampling (b) The True Solution (c) W/O Error Sampling

Another PDE example, parabolic equation(5.2):

$$\begin{aligned} \partial_t u(x,t) - \nabla_x\cdot\left((1+\frac{1}{2}|x|)\nabla_x u(x,t)\right) &= f(x,t), \quad \text{in } \Omega := \omega\times\mathbb{T} \\ u(x,t) &= g(x,t), \quad \text{on } \partial\Omega = \partial\omega\times\mathbb{T} \\ u(x,0) &= h(x), \quad \text{in } \omega, \end{aligned}$$

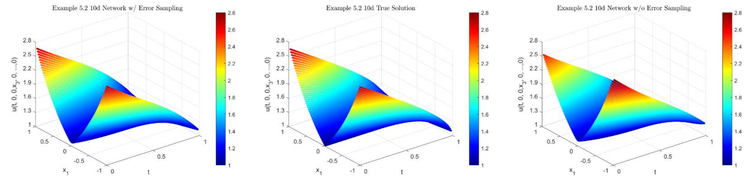
, where $\omega := \{x : |x| < 1\}$, $\mathbb{T} = (0,1)$, $g(x) = e^{|x|\sqrt{1-t}}$, and $h(x) = exp(|x|)$.

$f(x)$ is specified appropriately so that the exact solution is:

$$u(x,t) = e^{|x|\sqrt{1-t}}$$

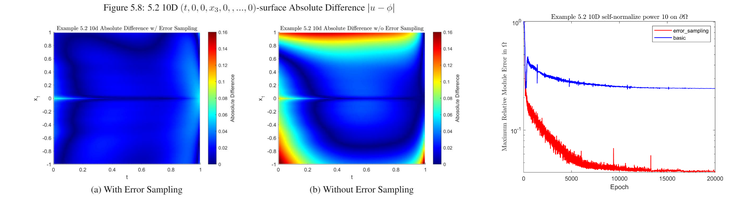
Results:

Figure 5.7: 5.2 10D $(t, 0, 0, x_3, 0, , ..., 0)$-surface of network solutions and the true solution



(a) With Error Sampling (b) The True Solution (c) W/O Error Sampling

Figure 5.8: 5.2 10D $(t, 0, 0, x_3, 0, ..., 0)$-surface Absolute Difference $|u - \phi|$

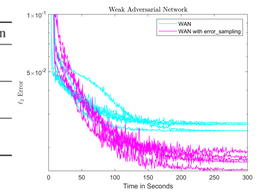


(a) With Error Sampling (b) Without Error Sampling

Compatibility

Error Sampling is also compatible with some recent frame-works, namely WAN, DRM, DGM[1][2][3].

Framework	Mean	Standard Deviation	Minimum Value	Coefficient of Variation
DGM	0.0333	0.0064	0.0244	19.2%
DGM*	0.0280	0.0079	0.0155	29.2%
DRM	0.0273	0.0097	0.0132	35.5%
DRM*	0.0255	0.0093	0.0122	36.5%
WAN	0.0329	0.0062	0.0245	18.8%
WAN*	0.0282	0.0075	0.0153	26.6%



*** denotes being equipped with error sampling**

References

[1] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. Journal of Computational Physics

[2] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. Journal of computational physics, 375:1339–1364, 2018. ISSN 0021-9991.

[3] Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 6(1), 2018. ISSN 2194-6701.

Acknowledgement: This work is supported by **NSF under grant DMS-2050133.**