Documentation Notes for 'EPINN_Poisson.ipynb'

Written by Hai Siong Tan
✉haisiong.tan@gryphonai.com.sg

November 2025

This document provides more detailed explanations for the notebook **'EPINN_Poisson.ipynb'** in this repository. This project implements an application of *Evidential Physics-Informed Neural Networks* (E-PINN) to a 1D poisson equation with a Gaussian source, as presented in Section 3 of [1]. As a controlled case study, the 1D Poisson equation examined was

$$\frac{d^2u}{dx^2} + e^{-\frac{(x-x_0)^2}{2\sigma_f^2}} = 0 \tag{1}$$

where x_0, σ_f^2 are parameters to be learned. The main goal of the notebook was to study if E-PINN can be used effectively to recover these parameters from a synthetic dataset constructed based on actual solutions to (1).

In the following, we provide some explanatory notes for the main sections of the notebook. Each section is intended to be read alongside the corresponding one in the notebook that can be straightforwardly run in Colab. The first 'Preliminaries' section includes a list of helper functions.

1 Dataset

This section generates the synthetic training data which is basically a set of solutions to (1) with some random noise added to simulate empirical observations. The domain was taken to be $0 < x < 1$, with the boundary condition $x = 0$ at both ends. We add some additional datapoints (**'x_bx_left, x_bx_right'**) to increase the weight of the target at these two boundary points.[1] The helper function **'num_sol_poisson (x_0, σ_f^2, N)'** generates a numerical solution to (1) in the form of a N-component array of points in $x \in [0, 1]$. For our work in [1], we took $x_0 = 1/3, \sigma_f^2 = 0.02$ for the training dataset.

2 1st training phase

In this section, we train the model based purely on data without alluding to the ODE description. The model architecture is a 2-layer perceptron model with 16 neurons in each layer. The loss function **'evidential_data_loss'** [2] is defined as the negative logarithm of the probability density in eqn. (6) of [1], i.e.

$$\begin{aligned} \mathcal{L} &= -\log[P(\mathcal{D}|\mathcal{M}(w)], \\ P(\mathcal{D}|\mathcal{M}(w) &= \frac{\Gamma\left(\alpha+\frac{1}{2}\right)}{\Gamma(\alpha)\sqrt{2\pi\beta(1+\nu)/\nu}}\left(1+\frac{(y-\gamma)^2}{2\beta(1+\nu)/\nu}\right)^{-(\alpha+\frac{1}{2})}. \end{aligned} \tag{2}$$

This is implemented by the helper function **'evidential_data_loss'**. The model has four outputs corresponding to the target u. In the line **'L_p_alpha, L_p_beta, L_p_nu, L_pred =**

[1]This is sometimes known as imposing the boundary conditions *softly*. Another approach would be use an ansatz for $u(x)$ such that by construction $u(0) = u(1) = 0$, e.g. $u(x) = x(1-x)\tilde{u}$ and let the dependent variable be $\tilde{u}$ instead.

nn_I(x_train)', the four quantities in the LHS refer to $\alpha, \beta, \nu, \gamma$ in eqn. (2) above. The first three quantities are related to the predictive uncertainty while the final one is the mean value.

For reference, our repository contains a trained model (at the end of the first phase) **'ini_epinn_poisson .pth'** in the 'models' folder.

3 Intermediate phase

In this phase, we first use the data-fitted model (obtained from the first phase) to construct prior density functions for the unknown parameters x_0, σ_f^2.

We first set the range of admissible values for each parameters as expressed in **'x0_vals', 'var_vals'**. We then construct the density function $f(\Omega)$ as described in eqn.9 of [1] by calculating the deviation between each numerical solution (defined within the domain of (x_0, σ_f^2)) and the data-fitted model. The array **'prob_grid'** refers to this function $f(\Omega)$.

Assuming the prior density for the parameters to be the product of two independent Gaussians, we determine the means and variances of these Gaussians as follows.

- The modes of $f(\Omega)$ are taken to be the means of the Gaussian priors for x_0, σ_f^2, as implemented by the lines **'x0_prior_mean = x0_vals[row] var_prior_mean = var_vals[col]'** and preceding ones.
- The interquartile range of $f(\Omega)$ was used to set the variances of the Gaussian priors for x_0, σ_f^2, as implemented by the lines **'x0_std = (Q3 - Q1)/1.349, var_std = (Q3 - Q1)/1.349'** and surrounding ones.
- In the plot of the parameter prior function $\pi(\Omega)$, we also overlay it on the density function $f(\Omega)$ (shaded in gray). The latter is more complicated and the prior we constructed is a simplified representation.

The second part of this intermediate phase lies in setting the prior for the loss weight of the PDE residual loss term in the code section titled **' Prior for the ODE residual's loss weight σ_R^2'**. This section follows the methodology described in Section 2.3 of [1]. The loss term for the PDE residual is

$$\frac{1}{2\sigma_R^2} \sum_{k=1}^{N_D} \mathcal{R}_k^2, \ \ \mathcal{R}^2 \equiv \left(\frac{d^2 u}{dx^2} - e^{-\frac{(x-x_0)^2}{2\sigma_f^2}} \right)^2, \tag{3}$$

where N_D is the number of training data points. In our formalism, σ_R^2 is a learnable parameter. We assume the prior for σ_R^2 to be an inverse gamma function

$$\pi(\sigma_R^2; \alpha_r, \beta_r) = \frac{\beta_r^{\alpha_r}}{\Gamma(\alpha_r)} \sigma_R^{-2(\alpha_r+1)} e^{-\frac{\beta_r}{\sigma_R^2}}, \tag{4}$$

where α_r, β_r are parameters to be determined as follows.

We would like to set the initial σ_R^2 such that the likelihood function $\sim \exp\left(-\frac{1}{2\sigma_R^2}\sum_{k=1}^{N_D} \mathcal{R}_k^2\right)$ is close to the parameter prior $\pi(\Omega)$ in the sense of the Kullback-Leibler divergence [3] between them. This is implemented in the code cell that ends with the line **'mean_var = optimal_beta.x[0]'**. We call the LHS **'mean_var'** as we would like to set this initial σ_R^2 to be the mean of the prior distribution (4) which is $\alpha_r/(\beta_r - 1)$. We also included a plot snippet for visually checking that this KL-divergence minimization is carried out correctly (i.e. there is a local minimum point in the graph). This corresponds to eqn.(14) of [1].

As the model is trained using the PDE constraint and data-fitting in the second phase, we would like σ_R^2 to evolve towards an ideal minimal value. A natural candidate for this minimal value is the one that yields a likelihood function close to a density function with variances characterized by the minimal resolution of the parameter domains. This is implemented in the code cell that ends with the line **min_var = optimal_beta.x[0]**. Setting this to be the mode of the prior density (4) allows us to solve for the parameters of (4) which is done in the code cell containing **'alpha_r = 1 + eps_s , beta_r = eps_s*mean_var'**. Since we are taking the negative logarithm of the prior for σ_R^2 in the loss function, gradient descent will induce a decrease of σ_R^2 from the mean to the mode. This means that the relative importance of the PDE residual loss term increases during model training.

To summarize, the intermediate phase involves fixing two prior density functions: one for the parameters $\pi(\Omega)$ and the other for the loss weight parameter of the PDE residual loss $\pi(\sigma_R^2)$. This method can be used whenever it is not clear what the prior density function for the parameters should be. In contexts where there may exist some reasonable choice for the prior $\pi(\Omega)$, then one doesn't need this default version. In a typical Physics-Informed-Neural-Networks setup, the loss weight σ_R^2 is treated as a hyperparameter that can be adjusted based on convergence of the loss terms and validation results. Our method removes this ambiguity by allowing σ_R^2 to evolve dynamically and from an initial point consistent with the statistical interpretation of the PDE residual loss term being related to the likelihood function for the parameters.

For reference, we have also included results obtained from the intermediate phase **'poisson_Intermediate_Phase_Dict.pth'** in the 'models' folder.

4 Second training phase

This training phase involves the full loss function as defined in eqn.17 of [1], in particular this includes the PDE residual loss term of standard PINN. For clarity, here are the definitions of the various loss terms expressed in the snippet that involves model training:

- **'physics_loss'**: the PDE residual loss term eqn. (3), with **'s'** denoting σ_R^2.
- **'physics_prior_loss'**: non-constant terms in $-\log\left(\pi(\sigma_R^2)\right)$, with $\pi(\sigma_R^2)$ defined in eqn. (4).
- **'data_loss'**: the evidential deep learning loss function defined in eqn. (2).
- **'neg_ll_x0_prior', 'neg_ll_var_prior'**: the Gaussian prior density function for x_0, σ_f^2. The means and variances are determined as described earlier in our discussion of the 'Intermediate phase'. The means are also taken as the initial values for the learnable parameters in this final training phase.

References

[1] Tan HS, Wang K, McBeth R. Evidential Physics-Informed Neural Networks for Scientific Discovery; 2025. Available from: `https://arxiv.org/abs/2509.14568`.

[2] Amini A, Schwarting W, Soleimany A, Rus D. Deep evidential regression. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20. Red Hook, NY, USA: Curran Associates Inc.; 2020. .

[3] Shlens J. Notes on Kullback-Leibler Divergence and Likelihood; 2014. Available from: https://arxiv.org/abs/1404.2000.