Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

AI in the Sciences and Engineering **2024**

Student: Carla Judith López Zurita

Project 1 Due date: Friday 12 April 2024, 23:59 (midnight)

The main objective of the project is to apply the concepts learned in class by implementing our own machine learning algorithms. The tasks are related to solving differential equations using a Physics Informed Neural Network (PINN), as well as solving an inverse problem. The project also includes a regression problem and an optional task to test the robustness of a given PINN. My attempt at solving each of the mentioned tasks will be described in detail below.

1. Task 1: PINNs for solving PDEs

The first task consists in solving the system of equations given by the heat equation for a fluid and a solid in a thermal storage. The equations are as follows:

$$\frac{\partial T_f}{\partial t} + U_f \frac{\partial T_f}{\partial x} = \alpha_f \frac{\partial^2 T_f}{\partial x^2} - h_f(T_f - T_s) \quad \text{for} \quad x \in [0,1], \quad t \in [0,1], \tag{1}$$

$$\frac{\partial T_s}{\partial t} = \alpha_s \frac{\partial^2 T_s}{\partial x^2} + h_s(T_f - T_s) \quad \text{for} \quad x \in [0,1], \quad t \in [0,1], \tag{2}$$

with appropriate boundary conditions and constant values defined in the project description. To solve this problem, a two-outputs neural network $(t, x) \rightarrow (T_f^\theta, T_s^\theta)$, with tunable parameters θ, was trained. The implementation was based on the Tutorial 2 presented in class. It consists on a very straight-forward implementation of a feed-forward neural network using pytorch. The neural network uses SiLU and Linear activation functions and has with adjustable number of layers and neurons, for which I chose 2 and 100 respectively. The numbers reveal a dense but not very deep network. The neural network was trained using the LBFGS optimizer with ten thousand iterations done over a single epoch. The optimizer parameters include a learning rate of 0.5, a history size of 150, with a strong Wolfe condition. The loss function is a custom function that consists of the sum of the mean squared error for the PDEs and the boundary conditions. Both Dirichlet and Von Neumann boundary conditions were used, with the latter requiring a derivative. This and all other derivatives required by the system of equations were calculated using the autograd function. The results of the loss function when training are shown in Figure 1. The plot shows the loss function decreasing steadily and smoothly, reaching a minimum value of 1×10^{-04}. Figure 2 shows the visualization of the results of the PINN for the heat equation. The plot shows the approximate solution for the fluid T_f and solid phases T_s at the end of the simulation. The results show a smooth and continuous solution that seems to be a good approximation of the real system.

2. Task 2: PDE-Constrained Inverse Problem

In the second task, the goal is to solve an inverse problem for the following equation,

$$\frac{\partial T_f}{\partial t}(x,t) + U_f(t) \frac{\partial T_f}{\partial x}(x,t) = \alpha \frac{\partial^2 T_f}{\partial x^2}(x,t) - h_f(T_f(x,t) - T_s(x,t)) \quad \text{for} \quad x \in [0,1], \quad t \in [0,8], \tag{3}$$

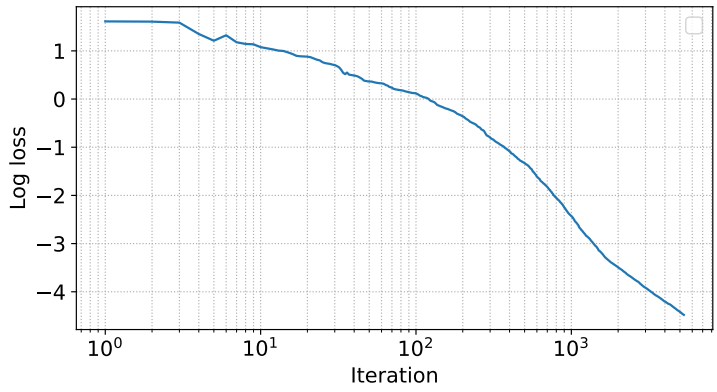

Figure 1: Loss function during training.

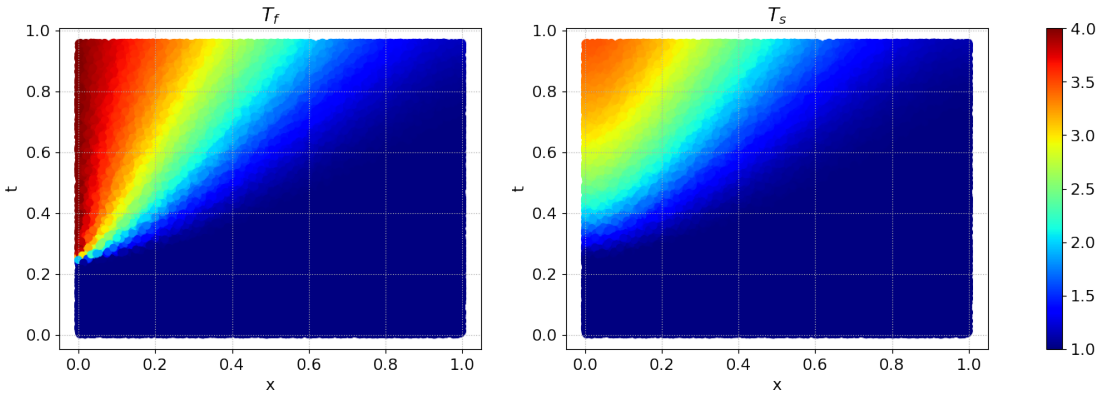

Figure 2: Approximate solution for the fluid T_f and solid temperatures T_s.

where the goal is to find the solid temperature T_s through all 8 phases of the simualtion. The equation is accompanied by a series of initial and boundary conditions (Dirichlet nd Von Neumann), as well as values for the time dependent parameter $U_f(t)$. These correspond to the different phases of the system, namely charging, idle and discharging. The problem is solved by training two different neural network with the same architecture, each one with a different output corresponding to the fluid and solid temperatures. The neural network follows a similar architecture as the one in Task 1, but has a Tanh activation function instead and uses Xavier initialization. The neural network is trained using the LBFGS optimizer with 10000 iterations over two epochs, with a learning rate of 0.3. The loss function is again a custom function that depends on the output of both neural networks, the PDEs and the initial and boundary conditions. The plot regarding the loss function during training is shown in Figure 3. The plot shows the loss function decreasing steadily and smoothly, but does not reach such a low minimum value as in Task 1, reaching instead a minimum value of -0.827. The training took around 30 minutes to complete on an M1 processor. The results of the inverse problem are shown in Figure 4. The plot show a good approximation of the solid temperature T_s through all 8 phases of the simulation. We can see the "discovered" physics in the plot, as the solid temperature T_s shows an interesting pattern that seems to be related to the fluid temperature T_f.

3. Task 3: Applied Regression

This task consists in solving a regression problem using a neural network. The problem is based on the California Housing dataset [2], and based on the Tutorial "California Housing analysis and preciction" by Larchenko [1] . The goal is to predict the median house value for California

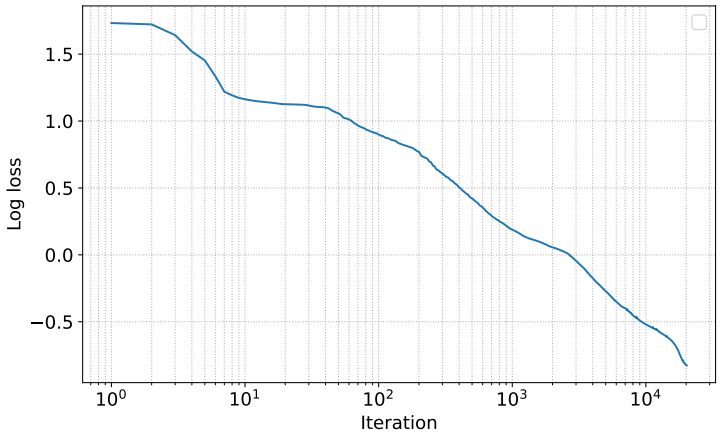

Figure 3: Loss function during training.

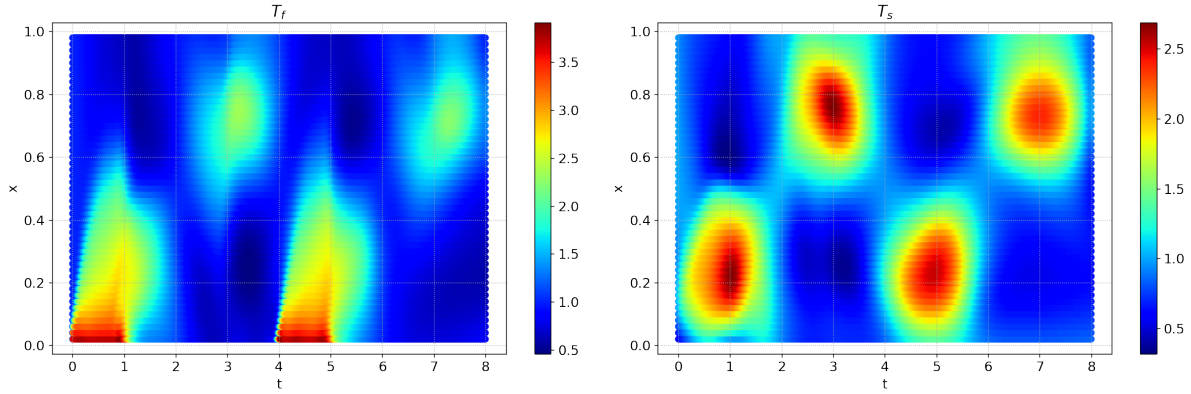

Figure 4: Results of the inverse problem for the system of equations, showing the fluid T_f and solid temperature T_s.

districts, given a set of features. I used the ones present on the dataset and derived by Larchenko. The dataset is divided into training and testing sets, with 75% of the data used for training and the remaining 25% for testing. The neural network used for this task has a single output and uses the ReLU activation function. The network has two hidden layers with 100 neurons each. The network is trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The loss function used is the mean squared error. The results of the loss function during training are shown in Figure 5. We can see that the loss function decreases steadily but starts to oscillate after a certain number of iterations. The minimum value reached The evaluation metrics for the regression problem are shown in Table 1. The metrics show that the model has a good performance, with a low mean squared error and a high R-squared value. The model seems to be a good approximation of the real system.

Table 1: Evaluation Metrics

Metric	**Normalized Values**	**Unnormalized Values**
Mean Squared Error (MSE)	0.0114	2,692,925,173.81
Root Mean Squared Error (RMSE)	0.107	51,893.4
Mean Absolute Error (MAE)	0.0705	34,170.62
R-squared (R2)	0.7954	0.7954

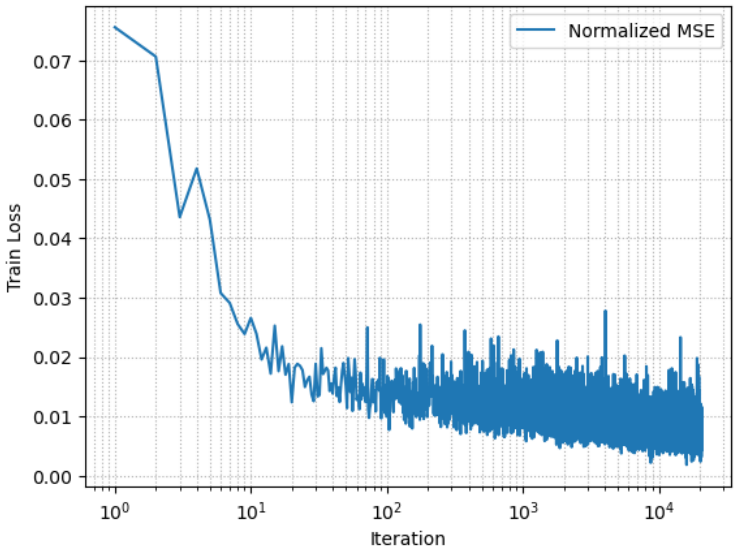

Figure 5: Loss function during training.

References

[1] Ilialar Larchenko. California housing analysis and preciction. `https://www.kaggle.com/code/ilialar/california-housing-analysis-and-preciction`, 2024. Accessed: April 08, 2024.

[2] Harry Wang. California housing data (1990): California housing price prediction. `https://www.kaggle.com/datasets/harrywang/housing`, 2024. Accessed: April 08, 2024.