Pablo Lahmann - 19-947-035
AI in the Sciences and Engineering
April 12, 2024

Project 1: Thermal Storage Design with PINNs

DUE: April 12th, 23:59 CET.

Question 1. PINNs for solving PDEs

For this task I received the best results when implementing one Neural Network for computing the values for T_f and T_s simultaneously. Particularly, I used a fully connected architecture with 4 hidden layers, and 20 hidden nodes in each layer. This architecture gave the best loss results, compared to using different amounts of layers or hidden nodes.
Similar to previous tutorials, my approach consisted of several different steps. First, I prepare the points, on which later the loss is computed. This happens by sampling the domain points relevant for the interior and boundary regions with help of the Sobol Sequence. Secondly, computing the predictions (including their derivatives, if necessary) on the previously chosen points in the domain. In a next step, I compute the residuals using the predicted values. These residuals are based on the conditions given by the PDEs and boundary conditions. The mean of the squared residual values of each residual type, then results in the loss which I use for optimizing the Neural Network. These losses are grouped into two categories: interior loss and boundary loss. During my training I achieved very good results by weighting the boundary loss with a factor of $\lambda = 10$. Since it is clearly crucial that the boundary conditions are fulfilled correctly.
Lastly, for optimizing the Neural Network I received best results when using the LBFGS optimizer, like in the tutorials. With not too much runtime I thus achieved a good result.

Question 2. PDE-Constrained Inverse Problem

For this task I received the best results when implementing one Neural Network for computing the values for T_f and T_s simultaneously.
First, I prepare the points, on which later the loss is computed. This happens by sampling the domain points relevant for the interior and boundary regions with help of the Sobol Sequence, as well as reading the provided data (for supervised training) from the file 'DataSolution.txt'. Secondly, computing the predictions on the previously computed points in the domain. An important step to apply the boundary conditions correctly, is to identify in each domain point of the boundaries, whether they are situated in a charging, discharging, or idle phase. For this I implemented a new function called 'time is phase'. In a next step, I compute the residuals using the predicted values. These residuals are based on the conditions given by the PDEs and boundary conditions. Note that here, it was necessary, to also create an additional function for the fluid velocity, with values depending on the current phase of the process. The mean of the squared residual values of each residual type, then results in the loss which I use for optimizing the Neural Network. These losses are grouped into three categories: interior loss, boundary loss, and supervised loss. Where the supervised loss corresponds to the loss wrt. the provided data for the values of T_f. During training it seemed particularly important to weigh the supervised loss very heavily, since it is clearly crucial to learn the values for T_s correctly. To help with this, I adapted the code to resample random supervised data for training during each epoch.
Unfortunately, I did not manage to achieve satisfactory results after training. I have done extensive examination of my code, and am convinced of its correctness. Thus, I suspect that the optimization process (e.g. using ADAM) requires epochs in the range of thousands, to converge to a good result. Unfortunately, I was not able to do this for task 2.

Question 3. Applied Regression

Data Understanding: When analyzing the data we notice for some feature clipping, and skewed histograms (which can be corrected by applying log+1). With correlation analysis we can exclude some redundant features. Some features have Nan values, which can be filled using linear regression. Further, by looking at features, like latitude, longitude and ocean proximity, we can clearly see that location strongly influences the target variable. This could be used in very creative ways to come up with more helpful features.

Model Implementation: I trained a fully connected Neural Network (ReLU activation) with different values for number of hidden layers, and hidden nodes. As formulated in the task description, I focused on exploring how the RMSE changes with different architectures and hyperparamaters. I did not modify the available features (since we are not asked to do this).
It turns out that as soon as an architecture with sufficiently many variables (such as a NN with 4 hidden layers with 20 hidden nodes each), the training and test loss will not differ significantly. It will stay around as RMSE value of 115000. Note that even drastic increases in the number of variables (NN such as 10 hidden layers with 100 hidden nodes in each layer), will not significantly improve the RMSE.

Comparison with Reference: Note that the best model in the kaggle notebook achieved a RMSE of around 44000, which is significantly better than the above described NN. But note that a significant difference is given by the extensive analysis of the data which is performed, to create additional helpful features, for prediction. This was not done in the above described model, which clearly shows how necessary this step of creating new features is. Further, models were used, that are quite different to Neural Networks. This also reminds us of the fact that simply using a NN is not always to best way to make predictions.
In general, even for the best models, it turns out to not be too easy to predict housing prices in California using this data. The best achieved RMSE is still significantly large.

Question 4. Robustness of PINNs and Transferability

We apply the hint from the task description, and thus notice the following bound:

$$\begin{aligned}
\|u_\theta(x) - u_\delta(x)\|_{C^1(B_{1/2})} &= \|u_\theta(x) - u(x) + u(x) - u_\delta(x)\|_{C^1(B_{1/2})} \\
&\leq \|u_\theta(x) - u(x)\|_{C^1(B_{1/2})} + \|u(x) - u_\delta(x)\|_{C^1(B_{1/2})} \\
&\leq \epsilon + C_{\epsilon'}(\|u(x) - u_\delta(x)\|_{L^\infty(B_1)} + \|f(x) - f_\delta(x)\|_{L^\infty(B_1)}) \\
&\leq \epsilon + C_{\epsilon'}(\|u(x) - u_\delta(x)\|_{L^\infty(B_1)} + \|\delta\varphi\|_{L^\infty(B_1)})
\end{aligned}$$

This gives us a good upper bound.
From this upper bound we can deduce that the robustness of a PINN depends on the perturbation of the data. The less perturbation we have, the better a bound we receive.