Independent Study Summary Next Steps

Matthieu Meeus

May 2020

1 Introduction

This document will briefly summarize the work done during the independent study of Spring 2020. Under supervision of Prof. Sondak and with support from Prof. Protopapas and Dr. Mattheakis, several interesting ideas have been implemented and evaluated to improve the performance of neural networks solving ordinary and partial differential equations (ODEs and PDEs). First, an overview will be listed of what has been tried and corresponding results will be briefly discussed. Next, the next steps and future work will be listed, which will serve as starting point when this research will be picked up again during the summer or at the start of Fall 2020. Finally, a brief conclusion and acknowledgements are added.

2 Summary of work done

The following list gives an overview of the work done during Spring 2020:

- From a mathematical perspective, we first solved for the analytical solution of several PDEs through the common method of separation of variables. Often in the context of the heat equation, both 2D and 3D equations in polar and cylindrical coordinates have been solved with various boundary conditions and corresponding basic functions in the analytical solution. After some trials, a very solid understanding of separation of variables has been built. This has led to a great intuition of how the analytical solutions of real-world problems are composed of linear combinations of non-linear functions. This inspires us to build neural networks to solve the same equations and to incorporate contextual knowledge inside their architecture to do so more efficiently.
- Next, it was useful to write out the mathematical expression of the output of a neural network in terms of the weights, activation functions and input parameters. We learned that implementing more layers leads to a serial application of the activation function (for instance a sine of a sum of sines), while more nodes increases the length of the linear combinations within one sine. Both options lead to an equal increase in weights.

- In order to get familiar with Pytorch in general and with neural networks to solve PDEs in particular, we built a network to solve a simple ODE from scratch. This made it very clear how exactly the set-up works, including the boundary/initial value operator that enforces the satisfaction of the requirements and the differential equation that is directly used as loss function.

- Getting familiar with the set-up raised the attention to the generation of the training data. Up until then, uniform sampling and equidistant methods with and without noise were used. From a mathematical perspective, it could make sense to use Chebyshev interpolation instead, as this method is proven to minimize the interpolation error for polynomial fitting. Although being an interesting idea, it turned out that the performance was actually slightly worse than the other methods.

- Next, it was understood that the neural network solution were pretty bad outside the training domain. As this is especially unfortunate for periodic functions, we attempted to enforce periodicity in the solutions by using a so-called 'masternode'. Here, the input was first activated by a sine before being further used as input to the overall network. As such, periodic solution would naturally arise, with the frequency as trainable parameter. Note that the periodicity was also included in the ansatz operator. Different conclusions could be made based on the results. First, the loss function inside the training domain was often lower. Next, the network solution was always periodic but only rarely it was able to predict the correct frequency. The performance was extremely sensitive to the value of the trained frequency.

- After solving very easy ODEs, we decided to challenge our network and apply it to strongly non-linear, but still periodic ODEs: the Van der Pol and Duffing equation. In both cases, the NN solution were very far off and it was clear that it is very hard to learn solutions of strongly non-linear equations. It also became clear that the performance of the masternode network was significantly worse than the traditional network, despite it enforcing periodic solution.

- With this information in mind, we decided to move to PDEs, from now on only using the 'neurodiff' package built by Feiyu. A simple 2D Laplace equation was solved to get familiar with the code. Quickly, it became clear how much effort and thought was put into the code and how useful this would become for the purposes for this project.

- We could now move onto a very practical problem: the 2D Helmholtz equation in polar coordinates. First the exact analytical solution was computed, which included combinations of Bessel functions and sines/cosines. Next, a polar coordinate version of the code was built and used to solve the equation, successfully.

- We noticed that enforcing the boundary condition for r was easy, but that enforcing periodicity for θ was hard. The best approach seemed include a penalization in the loss function.
- Next, it was confirmed that using sine as activation consistently outperforms the use of Tanh.
- With all the intuition built up until now, we could finally start playing around with heterogeneous layers inside the network! For the Helmholtz equation specifically, a network was built that uses Bessel functions and sines as activation for the input parameters r and θ respectively. With a tensor outer product operation directly following the initial activation, we believe to have built a true 'separation of variable' neural network. Until this point, the performance is not nearly near to what we could expect.
- Lastly, a heterogeneous layer of sines and cosines was built in order to mimic the Fourier Series/Fourier Transform. While this sounds as a very interesting idea, an elaborate comparison with the real Fourier Transform still needs to be set up.

3 Next steps

With all these interesting experiments, a profound intuition has been built for the problem at hand, which only makes it more exciting to discuss future steps. These would include:

- First, the heterogeneous, 'separation of variable' network should actually work. The true analytical solution is directly present in the output of the neural network, but it does not seem to learn the correct weights to find it. This could be because of a mistake in the Pytorch implementation, because of the intrinsic difficulties in using Bessel functions as activation or making it too complicated with the tensor outer product and flattening of the output. This needs to be explored in detail and we would expect a fully working SOV network that potentially has very interesting expansions.
- Next, it is also crucial to further explore the Fourier network and make a very solid comparison with the true Fourier Transorm. If this would work, very interesting next steps could arise.

We further note that, given the amount of fun experiments we already tried and the numerous ideas for sidetracks that came up along the way, the possibilities for next steps are endless.

4 Conclusion and acknowledgements

First, I'd like to express my gratitude towards Prof Sondak, Dr. Mattheakis and Prof. Protopapas for their feedback, support and very interesting input during the semester. It was truly fascinating to witness and participate in brainstorming sessions with you on this matter. I learned a lot about the technicalities of the problem, but also got a very valuable exposure to what individual research looks like. I'll bear this experience closely in mind when I make any decisions on my future. Next, I would also like to thank Feiyu for his amazing package. Its functionalities were obviously great for this project, but I'd like to point out the clear documentation and great structure that made it very easy to use and adapt the code to problems at hand. It is clear that a lot of interesting things have been touched, but that even more exciting work lies ahead. Many, many thanks for everything and I look forward to continuing this project in the near future.