Pittsburgh Interdisciplinary Mathematics Review
Volume 1, December 2024, Pages 59–70
https://doi.org/10.5195/pimr.2024.39

Shallow Neural Networks and Laplace's Equation on the Half-Space with Dirichlet Boundary Data

Malhar Vaishampayan

(Communicated by Leonardo Finzi)

Abstract

In this paper we investigate the ability of Shallow Neural Networks i.e. neural networks with one hidden layer, to solve Laplace's equation on the half space. We are interested in answering the question if it is possible to fit the boundary value using a neural network then is it possible to learn the solution to the PDE in the entire region using the same network? Our analysis is done primarily in Barron Spaces, which are function spaces designed to include neural networks with a single hidden layer and infinite width. Our results indicate in general the solution is not in the Barron space even if the boundary values are. However, the solution can be approximated to $\sim \varepsilon^2$ accuracy with functions of a low Barron norm. We implement a Physics Informed Neural Network with a custom loss function to demonstrate some of the theoretical results shown before.

Keywords: Elliptic partial differential equations, Laplace's equation, Shallow Neural Networks, Barron spaces, Physics Informed Neural Networks

1. Introduction

In recent years, Neural Networks have gained huge popularity in scientific computing. One such area is numerical solutions of Partial Differential Equations (PDEs). Oftentimes the PDEs we encounter in practical applications are too complicated to solve analytically and we have to rely on numerical approximations of solutions. Frequently, in scientific computing all we have are some observations and the physical law the phenomenon satisfies in the form of a PDE. Thus, the boundary values also have to be learnt from the data and are not known exactly. To this end we are interested in answering the question *"If the boundary values can be learnt from the data and the physical laws are known, can the solution also be learnt by the same neural network?"*.
In this article, we investigate the ability of Shallow Neural Networks (i.e. Neural Networks with one hidden layer) to represent the solution of Poisson's equation $-\Delta u = f$ (where f is a given function) or the special case of Laplace's equation $-\Delta u = 0$ on the half-space.
In electrostatics, the electric potential satisfies Poisson's equation (f is the charge density) and in the modelling of diffusion, u models the steady state concentration if f is

the 'source' (production and consumption) of the diffusing quantity. More generally, the Laplace operator is the unique homogeneous and isotropic second order differential operator (i.e. the only operator for which $(\Delta u)(x_0 + Ox) = \Delta\big(u(x_0 + Ox)\big)$ for all $x_0 \in \mathbb{R}^n$ and all orthogonal matrices O). It therefore appears in all physical models for phenomena are invariant under translations and rotations, for instance in the spatial variables for the heat equation (diffusion) or the wave equation (vibration). It is also the prototypical example for the class of *second order elliptic partial differential equations*. We can therefore consider it as an import baseline.
In particular, we use the Rectified Linear Unit (ReLU) and its powers both as the activation function of the neural network and as the boundary value. ReLU is a very popular choice for the activation of neural networks, and we are primarily interested in the case where it is easy to fit the boundary values, asking: Is it also easy to simultaneously solve the PDE?

2. Barron Spaces and Shallow Neural Networks

Barron Spaces are a function class which was introduced independently several times in recent years [1, 4, 6], primarily for ReLU activation, based on the work of Andrew Barron in 1993 [2]. Barron spaces comprise classes of functions that can be approximated well by ReLu networks with a single hidden layer. A neural network with a single hidden layer of width n and activation function $\sigma : \mathbb{R} \to \mathbb{R}$ can be written as

$$f_n(x) = \frac{1}{n} \sum_{i=1}^{n} a_i \, \sigma(w_i^T x + b_i)$$

where the parameters $a_i \in \mathbb{R}$, $w_i \in \mathbb{R}^d$ are the weights of the network and b_i its biases. For this article we use $\mathrm{ReLU}^\alpha(x) = (\max\{0, x\})^\alpha$ as the activation function

$$f_n(x) = \frac{1}{n} \sum_{i=1}^{n} a_i \, \mathrm{ReLU}^\alpha(w_i^T x + b_i).$$

Barron spaces are function classes designed to include the infinite width limits of neural networks with a single hidden layer whose coefficients remain bounded. Let π be a probability measure on $\mathbb{R} \times \mathbb{R}^d \times \mathbb{R}$. We denote $f_{\pi,\alpha} : \mathbb{R}^d \to \mathbb{R}$

$$f_{\pi,\alpha}(x) = \mathbb{E}_{(a,w,b)\sim\pi}\Big[a\, \sigma_\alpha(w^T x + b)\Big] = \mathbb{E}_{(a,w,b)\sim\pi}\Big[a \,\mathrm{ReLU}^\alpha(w^T x + b)\Big].$$

The map $\pi \mapsto f_{\pi,\alpha}$ is not unique – in fact, any measure π which is invariant under the reflection map $(a, w, b) \mapsto (-a, w, b)$ represents the function $f_{\pi,\alpha} \equiv 0$. We therefore make the following definition: Let $D \subseteq \mathbb{R}^d$ be a subset. For a continuous function $f : D \to \mathbb{R}$, we define

$$\|f\|_{\mathcal{B}_\alpha(D)} = \inf\left\{\mathbb{E}_{(a,w,b)\sim\pi}\left[|a|\Big(|w|+|b|\Big)^\alpha\right] : f(x) = f_{\pi,\alpha}(x) \text{ for all } x \in D\right\}$$

and

$$\mathcal{B}_\alpha(D) = \{f \in C^0(D) : \|f\|_{\mathcal{B}_\alpha(D)} < \infty\}.$$

We call $\mathcal{B}_\alpha$ the α-Barron space. This follows previous works which have considered Barron spaces for powers of ReLU, such as [3, 5].

3. Laplace's Equation with ReLU^α Boundary Values

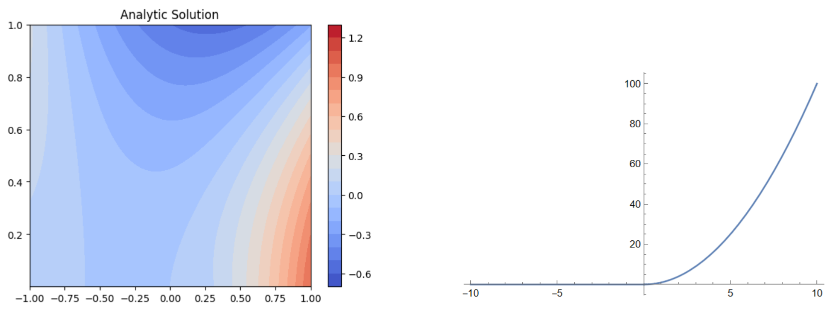
We consider the following equation:

$$\begin{cases} -\Delta u &= 0 \qquad\qquad \text{in } \mathbb{R}^2_+, \\ \quad u &= \mathrm{ReLU}^\alpha(x) \quad \text{on } \partial\mathbb{R}^2_+ \end{cases} \tag{1}$$

where $\mathbb{R}^2_+ = \mathbb{R} \times (0,\infty)$ and $\partial\mathbb{R}^2_+ = \mathbb{R} \times \{0\}$. For this article we will focus on the case $\alpha = 2$. In a future article we will consider the case where $\alpha \in \mathbb{N}$. In the same article we will also consider the case $\alpha \in (0,1) \notin \mathbb{N}$, which is entirely different from when α is a positive integer.

Theorem 1. *A solution to* (1) *is* $u = u_\alpha$ *where:*

$$u_\alpha(x,y) = \left(\frac{1}{\pi}\arctan\left(\frac{x}{y}\right) + \frac{1}{2}\right)(x^2 - y^2) - \frac{xy}{\pi}\log\left(x^2 + y^2\right) \textit{ if } \alpha = 2.$$



(a) $\alpha = 2$ Contour (b) $\alpha = 2$ Boundary

Figure 1: We visualize the solution of the PDE as contour plots and their boundary values.

Proof. We can begin by verifying the harmonicity of the solution:

$$\begin{aligned}
\partial_x u_2(x,y) &= \frac{1}{\pi}\left(x\pi - y + 2x\arctan\left(\frac{x}{y}\right) - y\log\left(x^2+y^2\right)\right)\\
\partial_{xx} u_2(x,y) &= 1 + \frac{2}{\pi}\arctan\left(\frac{x}{y}\right)\\
\partial_y u_2(x,y) &= \frac{-1}{\pi}\left(x + y\pi + 2y\arctan\left(\frac{x}{y}\right)\right) + x\log\left(x^2+y^2\right)\\
\partial_{yy} u_2(x,y) &= -1 - \frac{2}{\pi}\arctan\left(\frac{x}{y}\right).
\end{aligned}$$

Hence we have,

$$\Delta u_2(x,y) = \partial_{xx}u_2 + \partial_{yy}u_2 = 1 + \frac{2}{\pi}\arctan\left(\frac{x}{y}\right) - 1 - \frac{2}{\pi}\arctan\left(\frac{x}{y}\right) = 0.$$

Next, we can verify the boundary condition:

$$\begin{aligned}
\lim_{y\to 0^+} u_2(x,y) &= \left(\frac{1}{\pi}\frac{\pi}{2}\mathrm{sign}(x) + \frac{1}{2}\right)(x^2 - 0) - \frac{1}{\pi}\log\left(x^2+0\right)x(0)\\
&= \frac{x^2 + x^2\mathrm{sign}(x)}{2} = \mathrm{ReLU}^2(x)
\end{aligned}$$

Thus, the harmonicity and boundary condition are verified and u_2 is indeed a solution. □

3.1. Non-Uniqueness

It is important to note that our solution is not unique. In fact, the solution to the equation:

$$\begin{cases} -\Delta u &= f \quad \text{in } \mathbb{R}^2_+ \\ \quad u &= g \quad \text{on } \partial\mathbb{R}^2_+ \end{cases} \tag{2}$$

is never unique for any f, g. If u is a solution to (2) then so is $u+h$ where h is a harmonic function (i.e. $\Delta h = 0$) on $\mathbb{R}^2_+$ with zero boundary values on $\partial\mathbb{R}^2_+$. For instance

$$h(x,y) = \lambda_1 y + \lambda_2 xy + \lambda_3\left(\frac{y^3}{6} - \frac{x^2 y}{2}\right) + \lambda_4 e^x \sin(y) + \lambda_5 e^{-2x}\sin(2y)$$

For any coefficients $\lambda_i \in \mathbb{R}$ there is an infinite (and in fact infinite-dimensional) space of solutions. The non-uniqueness of solution in this problem comes due to lack of a boundary condition 'at infinity'.
Since we are considering the upper half of the plane, the region extends to infinity and we don't get a second boundary. In physics, Laplace's equation is used to model electric fields where a problem could arise if we can only observe a small part of the region we are considering.

3.2. Non-Barron Solution

We can ask: If the boundary data of the PDE is Barron, is the solution also Barron? So, if we can easily learn the boundary condition, can we also learn the solution? This is a now classical approach in the theory of PDEs: We phrase PDEs in terms of linear 'operators' between function spaces and use techniques from functional analysis, an infinite-dimensional generalization of linear algebra. Knowing whether we can find the solution in the space we want to use is of crucial importance.

We can show that our solution is not in $\mathcal{B}_2(\mathbb{R}^2_+)$. First we begin by showing that if a function of 2 variables is in $\mathcal{B}_2(\mathbb{R}^2_+)$ then its second derivatives are bounded. Recall, a $\mathcal{B}_2$ functions can be written as:

$$f(x,y) = \mathbb{E}_{(a,w,b)\sim\pi}[a\sigma(w^T x + b)]$$

We consider the mixed derivative as a representative example.

$$\begin{aligned}
\frac{\partial f}{\partial x \partial y} &= \mathbb{E}_{(a,w,b)\sim\pi}[a w_1 w_2 \sigma''(w^T x + b)] \\
&\leq 2\,\mathbb{E}_{(a,w,b)\sim\pi}[|a|\cdot|w_1|\cdot|w_2|] \\
&\leq 2\,\mathbb{E}_{(a,w,b)\sim\pi}[|a|\cdot|w|^2] \\
&= 2\,\|f(x,y)\|_{\mathcal{B}_2}
\end{aligned}$$

Thus if our function is in $\mathcal{B}_2(\mathbb{R}^2_+)$, i.e. if the Barron norm of the function is bounded, then the second derivative must be bounded. Now we can show the mixed second derivative of our solution is not bounded making it not Barron.

$$\begin{aligned}
\partial_x u_2(x,y) &= \frac{1}{\pi}\left(x\pi - y + 2x\arctan\left(\frac{x}{y}\right) - y\log(x^2+y^2)\right) \\
\partial_y\partial_x u_2(x,y) &= \frac{-1}{\pi}\left(1 + \frac{4x^3}{y^3}\frac{y^2}{x^2+y^2} + \log(x^2+y^2) - \frac{2y^2}{x^2+y^2}\right) \\
\partial_y\partial_x u_2(0,y) &= -\frac{1}{\pi}(1 + 0 + 2\log(y) - 2).
\end{aligned}$$

At the origin $y = 0$, which makes the log term blow up making the mixed second derivative unbounded. Thus, our solution is not Barron. However, we can approximate our solution by making the following change:

$$u_{2,\varepsilon} = \left(\frac{1}{\pi}\arctan\left(\frac{x}{y}\right) + \frac{1}{2}\right)(x^2 - y^2) - \frac{xy}{\pi}\log(x^2 + y^2 + \varepsilon^2)$$

for some $0 < \varepsilon < 1$.

With the ε correction the log term no longer blows up at the origin. Also note that $\Delta u_{2,\varepsilon} \to 0$ as $\varepsilon \to 0$ pointwise. In a future article we will establish the exact Barron norm of the approximation, which grows only logarithmically in ε, i.e. so slowly to *almost* be considered merely a constant. Now, we can verify that $u_{2,\varepsilon}$ still satisfies the boundary condition:

$$\lim_{y\to 0^+} u_{2,\varepsilon} = \left(\frac{1}{\pi}\frac{\pi}{2}\operatorname{sign}(x) + \frac{1}{2}\right)x^2 - \frac{1}{\pi}\log(x^2+\varepsilon^2)(0) = \mathrm{ReLU}^2(x)$$

Next, we can verify this is indeed a good approximation by computing the L^∞ norm of the difference on the upper half of the disk of radius R centered at the origin. Recall, the L^∞ norm of a continuous function is defined as $\|f\|_{L^\infty(D)} = \sup\{|f(x)| : x \in D\}$. So,

$$\begin{aligned}\|u_2 - u_{2,\varepsilon}\|_{L^\infty(B_R^+(0))} &= \left\|\frac{xy}{\pi}(\log(x^2+y^2+\varepsilon^2) - \log(x^2+y^2))\right\|_{L^\infty(B_R^+(0))} \\ &= \left\|\frac{xy}{\pi}\log\left(1+\frac{\varepsilon^2}{x^2+y^2}\right)\right\|_{L^\infty(B_R^+(0))}.\end{aligned}$$

Since we are considering a disk it is easier to switch to polar co-ordinates:

$$\begin{aligned}\|u_2 - u_{2,\varepsilon}\|_{L^\infty(B_R^+(0))} &= \left\|\frac{r^2\sin(\phi)\cos(\phi)}{\pi}\log\left(1+\frac{\varepsilon^2}{r^2}\right)\right\|_{L^\infty(B_R^+(0))} \\ &= \max_{0\le r\le R}\max_{0\le\phi\le\pi}\left|\frac{r^2\sin(2\phi)}{2\pi}\log\left(1+\frac{\varepsilon^2}{r^2}\right)\right| \\ &\le \max_{0\le r\le R}\left(r^2\log\left(1+\frac{\varepsilon^2}{r^2}\right)\right)\end{aligned}$$

To find the maximum we compute the r-derivative and set it equal to zero.

$$\begin{aligned}\frac{d}{dr}\left(r^2\log\left(1+\frac{\varepsilon^2}{r^2}\right)\right) &= 2r\log\left(1+\frac{\varepsilon^2}{r^2}\right) - \frac{2\varepsilon^2}{r^2+\varepsilon^2}r = 0 \\ \Rightarrow \log\left(1+\frac{\varepsilon^2}{r^2}\right) &= \frac{\varepsilon^2}{r^2+\varepsilon^2}\end{aligned}$$

Note that if $r = 0$, then the difference is zero, hence it can't be the maximum, and if $r = R$, then $r^2\log(1+\varepsilon^2/r^2) \approx R^2 \cdot \varepsilon^2/R^2 \approx \varepsilon^2$ for small ε, since $\log'(1) = 1$. Let $z := \frac{r^2}{r^2+\varepsilon^2}$ and note that $0 < z < 1$. Then

$$\begin{aligned}\Rightarrow \log\left(\frac{1}{z}\right) &= 1 - z \\ \Rightarrow \log(z) &= z - 1\end{aligned}$$

We can solve this numerically to get $z \simeq 0.137$ and hence

$$z = \frac{r^2}{r^2+\varepsilon^2} = 0.137$$
$$\Rightarrow r = \varepsilon\sqrt{\frac{z}{1-z}} = \varepsilon\sqrt{\frac{0.137}{1-0.137}} \simeq 0.4\varepsilon.$$

Finally, we have:

$$\|u_2 - u_{2,\varepsilon}\|_{L^\infty(B_R^+(0))} = (0.4\varepsilon)^2 \log\left(1+\frac{\varepsilon^2}{(0.4\varepsilon)^2}\right) \leq \varepsilon^2.$$

Thus, we have shown our error scales $\sim \varepsilon^2$. Our main message here is that even though the solution is not in $\mathcal{B}_2$ it can still be well approximated by functions with a fairly small $\mathcal{B}_2$-norm. This has favorable implications for the approximability of the function u_2 by finite neural networks and on 'learning' the solution of the problem from a small set of datapoints.

4. Physics Informed Neural Networks

We try to learn the solution of the Laplace's Equation with ReLU$^\alpha$ boundary using Physics Informed Neural Networks(PINNs) on the region $(x,y) \in [-1,1]\times[0,1]$. Since our solution is non-unique we are also interested in knowing which solution the algorithm learns. The boundary condition is only given on the portion of the boundary where $y = 0$.
PINNs (first introduced in [7]) are a type of Neural Networks designed to solve supervised learning tasks while following any given physical laws described by PDEs. These networks utilize the recent developments in automatic differentiation to differentiate the neural network with respect to its input co-ordinates. For our task we consider a fully connected neural network with one hidden layer that tries to minimize the loss given by:

$$L_\beta(\theta) = \frac{1}{N}\sum_{i=1}^{N}\Big(\Delta u_\theta(x_i,y_i)\Big)^2 + \frac{\beta}{M}\sum_{j=1}^{M}\Big(u_\theta(x_j,0) - \mathrm{ReLU}^\alpha(x_j)\Big)^2 \tag{3}$$
$$= \text{Laplacian Loss} + \beta\,\text{Boundary Loss} \tag{4}$$

where θ represents the parameters $(a_i, w_i, b_i)_{i=1}^n$ of the network and u_θ is the network itself. Here N and M are the number of points we consider in the interior and the boundary respectively (generally $N > M$) and β is the weight given to the boundary condition. This parameter can be modified to ensure the network does not ignore the boundary condition or the PDE entirely.

4.1. Implementation

We implement a single layer neural network with 2000 parameters (i.e. a network which has a width of 500 and a depth of 2) which are optimized using the Stochastic Gradient Descent algorithm with a learning rate of 10^{-3} and a momentum of 0.99. The implementation is done in Python using the PyTorch library, and we use the PyTorch default parameter initialization. Since we are only looking at networks with a single layer, it is possible to easily compute the Laplacian of the network with respect to the inputs analytically without needing to use automatic differentiation. This is done primarily to speed up the training process as it is computationally expensive to use automatic differentiation. However, this is in general not possible for more complicated network architectures and we would have to rely on automatic differentiation.
We implement a custom loss function using the basic idea of (3) but with a few important changes. Instead of using a fixed value for β throughout the training we make it an adaptive parameter. We write the new loss function as:

$$\begin{aligned} L_{\mu,\theta} &= \frac{(1-\mu)}{N}\sum_{i=1}^{N}(\Delta u_\theta(x_i,y_i))^2 + \frac{\mu}{M}\sum_{j=1}^{M}\Big(u_\theta(x_j,0) - \mathrm{ReLU}^\alpha(x_j)\Big)^2 \\ &= (1-\mu)\text{Laplacian Loss} + \mu\text{Boundary Loss} \end{aligned}$$

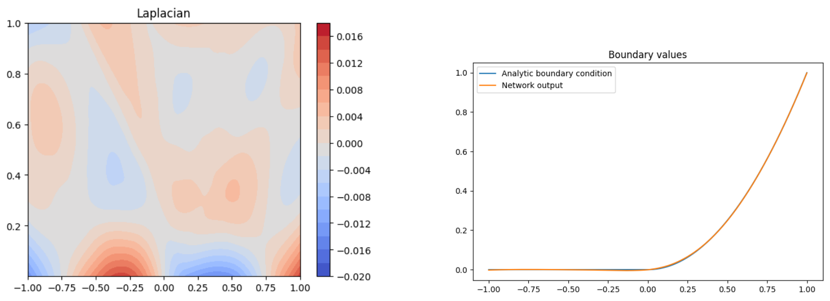
Here, instead of only adjusting the weight given to the boundary condition we have:

$$\begin{aligned} \mu &= \frac{\text{Boundary Loss}}{\text{Boundary Loss} + \text{Laplacian Loss}} \in (0,1) \\ 1-\mu &= \frac{\text{Laplacian Loss}}{\text{Boundary Loss} + \text{Laplacian Loss}} \in (0,1) \end{aligned}$$

This parameter μ is calculated and updated every step of the training process, making it adaptive. We use the losses from the previous time step for computational ease. It helps us to balance the importance given to the Laplacian loss and the boundary loss every step. If one of the two losses is much larger than the other, it will receive more attention in the following time step, so we prioritize to decrease both contributions to the loss simultaneously to comparable magnitude.
Figure 2 shows the results of the training. The Laplacian is near 0 at all points in the region and the boundary values are also learnt near perfectly.

4.2. Implicit Bias

When a problem has multiple solutions, the preference of an algorithm towards learning a particular solution or set of solutions is known as the implicit bias of the algorithm. Our results suggest that it is possible to solve equation (1) using PINNs, so we are interested in understanding which solution the network learns.



(a) $\alpha = 2$ PINN Simulation Laplacian Contour Plot

(b) $\alpha = 2$ PINN Simulation Boundary Plot

Figure 2: We visualize the Laplacian of our PINN numerical solution as a contour plot and the boundary values.

There is randomness inherent in the generation of the dataset, the initialization of the neural network weights, and the batch selection for stochastic gradient estimates. We therefore expect some variation between the solutions in different runs.

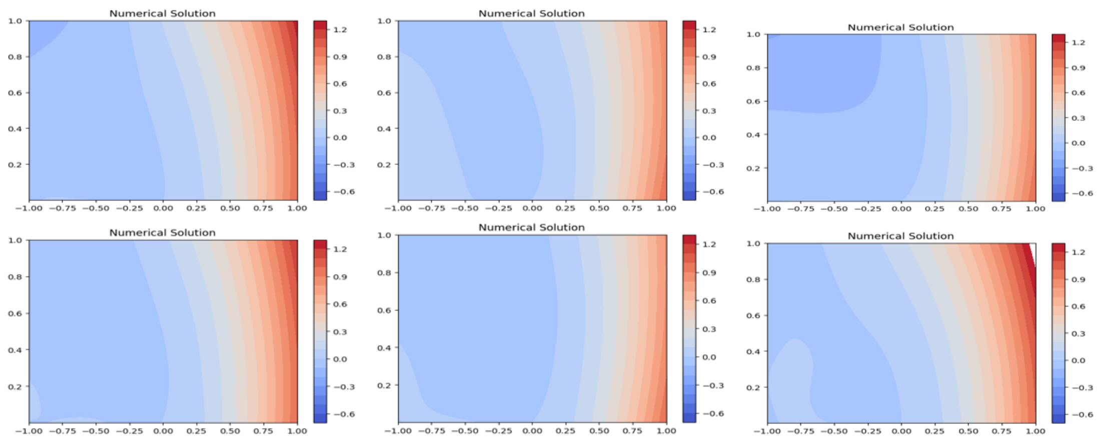


Figure 3: We visualize the countour plot of the solution learnt by PINNs for multiple runs.

From Figure 3 and Figure 1, it is clear that the solution learnt by a PINN is generally not the same as the one we presented in Theorem 1. The solution learnt by the PINN often seems to be monotonically increasing in the x direction and varies considerably less in the y direction than the explicit solution u_2.
With ReLU^2-activation, it is particularly easy to learn quadratic polynomials. To this end we conjecture that our learnt solution can be represented by our solution from 1 as:

$$\text{PINN Solution} \approx \text{Analytic Solution} + \lambda_1 y + \lambda_2 xy,$$

i.e. the two solutions mostly differ by a quadratic harmonic polynomial which vanishes on the boundary. The parameters $\lambda_1, \lambda_2 \in \mathbb{R}$ are the weights of linear harmonic corrector and the quadratic harmonic corrector respectively. To study this conjectured behavior in L^2, we compute $\lambda_1 y + \lambda_2 xy$ as the L^2-orthogonal projection of the difference between the analytic and numerical solutions. The optimal values are calculated as

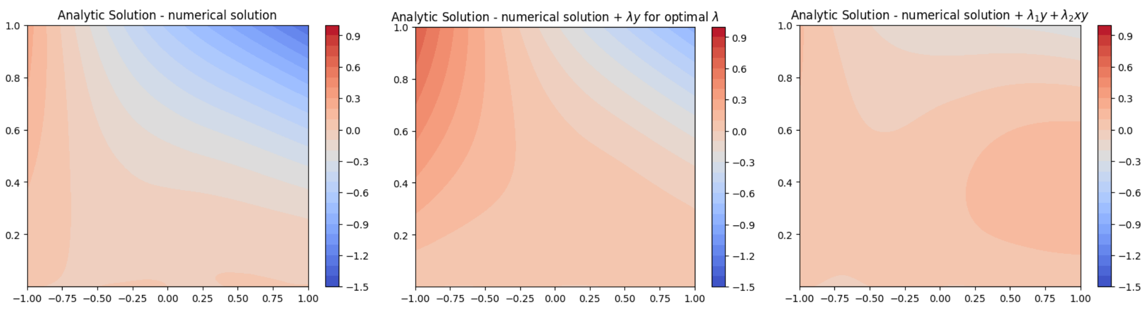
$$\lambda_1 = \frac{\langle u^{PINN} - u^{analytic}, y\rangle_{L^2((-1,1)\times(0,1))}}{\|y\|^2_{L^2((-1,1)\times(0,1))}}$$
$$\lambda_2 = \frac{\langle u^{PINN} - u^{analytic}, xy\rangle_{L^2((-1,1)\times(0,1))}}{\|xy\|^2_{L^2((-1,1)\times(0,1))}}$$

since the functions y, xy are L^2-orthogonal on the domain. We run the simulation 15 times to calculate the average values of λ_1 and λ_2 and their standard deviations. In the table below we present the results for 5 such runs.

Simulation	1	2	3	4	5
λ_1	0.4441	0.4451	0.5217	0.3163	0.3760
λ_2	0.5798	0.5410	0.7717	0.5046	0.4168
$\|u^{PINN} - u^{analytic}\|_{L^2}$	0.3322	0.3270	0.4074	0.2655	0.2718
$\|u^{PINN} - u^{analytic} - \lambda_1 y\|_{L^2}$	0.2107	0.2018	0.2738	0.1924	0.1632
$\|u^{PINN} - u^{analytic} - \lambda_1 y - \lambda_2 xy\|_{L^2}$	0.0824	0.0892	0.0912	0.0924	0.0847

Parameter	Mean	Std. Deviation
λ_1	0.3576	0.1164
λ_2	0.5784	0.0965
$\|u^{PINN} - u^{analytic}\|_{L^2}$	0.3000	0.0578
$\|u^{PINN} - u^{analytic} + \lambda_1 y\|_{L^2}$	0.2589	0.1612
$\|u^{PINN} - u^{analytic} - \lambda_1 y - \lambda_2 xy\|_{L^2}$	0.0891	0.0040

We can clearly see there is some decrease in the L^2 error when we use a linear corrector however there is a major decrease when using a quadratic corrector. The same can be seen through the contour plots in Figure 4.



(a) PINN - Analytical (b) PINN - Analytical with linear harmonic corrector (c) PINN - Analytical with quadratic harmonic corrector

Figure 4: We visualize the contour plot of the difference between the learnt solution and the analytic solution both with and without the correctors.

Unsurprisingly, the coefficient λ_2 is positive in all runs: Like the boundary condition, also the PINN solution is monotonically increasing in x. Also the parameter λ_1 appears to always be positive.

5. Conclusion

In this paper we investigated the ability of Shallow Neural Networks to solve the Laplace's equation with Dirichlet boundary data on the half-space. We were interested in knowing whether it is possible to use the same network to learn the boundary data and the solution. Our results seem to indicate this in general is not possible. However, we can approximate the boundary data and the solution to a great accuracy. Since the solution is not unique, we were also interested in understanding which solution the network learns. Our results seem to indicate when using a network with ReLU^2-activation, the learnt solution is well-approximated by the analytical solution plus a harmonic quadratic corrector. We do not analyze the remainder term here.

6. Acknowledgements

The author gratefully acknowledges the financial support from the Department of Mathematics at the University of Pittsburgh through a Painter Fellowship and from the Fredrick Honors College at the University of Pittsburgh through the Chancellor's Undergraduate Research Fellowship. The author is also thankful for the mentoring of Dr. Stephan Wojtowytsch under whose guidance this research was conducted.

References

[1] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

[2] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 1993.

[3] Tjeerd Jan Heeringa, Len Spek, Felix L Schwenninger, and Christoph Brune. Embeddings between Barron spaces with higher-order activation functions. *Applied and Computational Harmonic Analysis*, 73:101691, 2024.

[4] Chao Ma, Lei Wu, and Weinan E. A priori estimates of the population risk for two-layer neural networks. *arXiv preprint arXiv:1810.06397*, 2018.

[5] Tong Mao, Jonathan W Siegel, and Jinchao Xu. Approximation rates for shallow ReLU^k neural networks on Sobolev spaces via the Radon transform. *arXiv preprint arXiv:2408.10996*, 2024.

[6] Rahul Parhi and Robert D Nowak. Banach space representer theorems for neural networks and ridge splines. *Journal of Machine Learning Research*, 22(43):1–40, 2021.

[7] M Raissi, P Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019.

Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 12560
E-mail address: msv17@pitt.edu