Solving the Schrödinger eigenvalue problem using deep learning

Trung Do
Mentor: Prof. YuLong Lu
Department of Mathematics, University of Massachusetts Amherst

Abstract

We study the effectiveness of using feedforward neural network to solve ordinary differential equation, particularly the one-dimensional Schrödinger equation. We compare the accuracy of the neural network solutions with the solutions from another analytical method, the Finite Difference method. Based on the empirical result, we found the optimal network architecture that results in solutions with high accuracy compare to solutions of the analytical method. We conclude the study with some future research ideas.

1 Introduction and Motivation

The Schrödinger equation is one of the fundamental equations in quantum mechanics. Solving the time-independent Schrödinger equation gives the stationary-state wave function of a quantum mechanical system. The one-dimensional, time-independent Schrödinger equation is defined as the following:

$$\left[-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\right]\psi(x) = E\psi(x) \tag{1}$$

where m is the mass of the particle, $\hbar$ is the reduced Planck constant, E is the real energy eigenvalue of the wave function $\psi(x)$, and $V(x)$ is the potential function. The boundary condition for x is $[-L, L]$ such that:

$$\psi(-L) = \psi(L) = 0$$

Physicists are interested in solving the Schrödinger equation and have solved the equation for some special and interesting forms of the potential function $V(x)$ using the analytical method. However, in general settings, i.e., for the general form of the potential function $V(x)$, one often cannot solve the Schrödinger equation analytically, but rather make use of some approximation methods. Now, with the vast availability of computational power and the recent development in deep learning, neural networks can efficiently solve differential equations

with several advantages over traditional numerical methods. The neural network solutions are differentiable, analytical, and robust against the "curse of dimensionality" in solving differential eigenvalue problems in higher dimensions [1], [2]. In this study, we explore the application of a feedforward neural network in solving the time-independent Schrödinger equation.

2 Methodology

Without loss of generality, let $\hbar = m = 1$. The equation (1) becomes:

$$\frac{-\psi(x)''}{2} + V(x)\psi(x) = E\psi(x) \tag{2}$$

We wish to find the smallest eigenvalue E_{ground} and the corresponding eigenfunction $\psi(x)$, with the boundary condition:

$$\psi(-L) = \psi(L) = 0$$

We propose two methods to solve this eigenvalue problem: the Finite Difference (FD) method and the Neural Network (NN) method, to solve the one-dimensional Schrödinger eigenvalue problem. We want to evaluate the accuracy of the Neural Network method solution by comparing it with the analytical solution from the Finite Difference method.

2.1 The Finite Difference Method

We want to solve the system of differential equation (2) on the interval $[-L, L]$. Using numerical approach, we discretize the interval $[-L, L]$ into N points such that $x_1 = -L$ and $x_N = L$. We have a condition for the value of ψ evaluated on the boundaries: $\psi(x_1) = \psi(x_N) = 0$. Thus, we only have to evaluate the value of ψ on the remaining $N-2$ points on the interval $[-L, L]$, or equivalently, evaluate $\psi(x_i)$ for $i = 2, 3, ..., N-1$. Let

$$h = \frac{2L}{N-1}$$

be the equidistance between x_i and x_{i+1} for $i = 1, 2, ..., N-1$. We use the Taylor's theorem to expand the term:

$$\begin{aligned}
\psi(x_i + h) &= \sum_{n=0}^{\infty} \frac{\psi^{(n)}(x_i)}{n!}(x_i + h - x_i)^n \\
&= \psi(x_i) + \frac{h\psi'(x_i)}{1!} + \frac{h^2\psi''(x_i)}{2!} + R_n(x) \\
&= \psi(x_i) + h\psi'(x_i) + \frac{h^2}{2}\psi''(x_i) + R_n(x)
\end{aligned}$$

where $R_n(x)$ is the remainder term of higher order. The Taylor's theorem can also be expanded as:

$$\begin{aligned}\psi(x_i - h) &= \sum_{n=0}^{\infty} \frac{\psi^{(n)}(x_i)}{n!}(x_i - h - x_i)^n \\ &= \psi(x_i) - \frac{h\psi'(x_i)}{1!} + \frac{h^2\psi''(x_i)}{2!} + R_n(x) \\ &= \psi(x_i) - h\psi'(x_i) + \frac{h^2}{2}\psi''(x_i) + R_n(x)\end{aligned}$$

Assuming the term $R_n(x)$ is sufficiently small, combining $\psi(x_i + h)$ and $\psi(x_i - h)$ gets:

$$\psi(x_i + h) + \psi(x_i - h) \approx 2\psi(x_i) + h^2\psi''(x_i)$$

Taking ψ'' to one side:

$$\psi''(x_i) \approx \frac{\psi(x_i + h) + \psi(x_i - h) - 2\psi(x_i)}{h^2} \tag{3}$$

Replacing $x_i + h = x_{i+1}$ and $x_i - h = x_{i-1}$:

$$\psi''(x_i) \approx \frac{\psi(x_{i+1}) + \psi(x_{i-1}) - 2\psi(x_i)}{h^2} \tag{4}$$

For $i = 2, 3, ..., N - 1$, substituting (4) into equation (2), we get:

$$-\frac{\psi(x_{i+1}) + \psi(x_{i-1}) - 2\psi(x_i)}{2h^2} + V(x_i)\psi(x_i) = E\psi(x_i) \tag{5}$$

Rearrange the above equation, we get:

$$-\frac{1}{2h^2}\psi(x_{i-1}) + \frac{1}{h^2}\psi(x_i) - \frac{1}{2h^2}\psi(x_{i+1}) + V(x_i)\psi(x_i) = E\psi(x_i) \tag{6}$$

Solving for a particular case, $i = 2$:

$$-\frac{1}{2h^2}\psi(x_1) + \frac{1}{h^2}\psi(x_2) - \frac{1}{2h^2}\psi(x_3) + V(x_2)\psi(x_2) = E\psi(x_2) \tag{7}$$

Because $\psi(x_1) = 0$, the equation above becomes:

$$\frac{1}{h^2}\psi(x_2) - \frac{1}{2h^2}\psi(x_3) + V(x_2)\psi(x_2) = E\psi(x_2) \tag{8}$$

For $i = 3$, equation (2) can be represented in the form of (6) as:

$$-\frac{1}{2h^2}\psi(x_2) + \frac{1}{h^2}\psi(x_3) - \frac{1}{2h^2}\psi(x_4) + V(x_3)\psi(x_3) = E\psi(x_3) \tag{9}$$

Continuing the process for every $i = 4, 5, ..., N-1$, we discover that the general form of equation (2) evaluated at every discrete value x_i, $i = 2, 3, ..., N-1$ can be represented as a system of linear equations:

$$\frac{-1}{2h^2}\boldsymbol{P}\boldsymbol{\psi} + \boldsymbol{V}\boldsymbol{\psi} = E\boldsymbol{\psi} \tag{10}$$

where,

$$\boldsymbol{P} = \begin{pmatrix} -2 & 1 & 0 & \dots & 0 & 0 \\ 1 & -2 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 & -2 \end{pmatrix},$$

$$\boldsymbol{V} = \begin{pmatrix} V(x_2) & 0 & \dots & 0 \\ 0 & V(x_3) & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & V(x_{N-1}) \end{pmatrix},$$

$$\boldsymbol{\psi} = \begin{pmatrix} \psi(x_2) \\ \psi(x_3) \\ \vdots \\ \psi(x_{N-1}) \end{pmatrix}$$

We solve the eigenvalue problem for this system of equations using the numpy library in Python. For now, we care about the ground state eigenvalue E_{ground} and the corresponding eigenfunction ψ_{ground}. The eigenfunction ψ_{ground} is normalized by the constant $\frac{1}{\sqrt{h}}$. The solutions are displayed in section 3 alongside with the solutions from the neural network method for comparison.

2.2 The Neural Network Method

We use the Variational Principle to approximate the energy of the ground state of a system [3]. The Variational Principle "guesses" an initial trial wave function $\psi(\alpha)$ with an variational parameter α and try to minimize the function E by adjusting parameter α of ψ:

$$E[\alpha] = \langle\psi|H|\psi\rangle = \int_{-L}^{L} \psi'(x)^2 + V(x)\psi(x)^2\, dx \tag{11}$$

with the normalization constraint:

$$\langle\psi|\psi\rangle = \int_{-L}^{L} \psi(x)^2\, dx = 1 \tag{12}$$

We can approximate the integral in equation (11) using the Riemann sum:

$$E[\alpha] = \int_{-L}^{L} \psi'(x)^2 + V(x)\psi(x)^2\, dx \approx h \sum_{i=1}^{N} \left[\psi'(x_i)^2 + V(x_i)\psi(x_i)^2\right] \tag{13}$$

where $h = x_i - x_{i-1}$ for $i \geq 2$, and N is the number of discrete points in the interval $[-L, L]$.

The lower the energy $E[\alpha]$ with the normalization constraint we can get by adjusting the parameter α of ψ, the more accurate our approximation to the actual ground state energy is. Thus, minimizing $\langle\psi|H|\psi\rangle$ to the lowest value gives an approximation of E_{ground}.

We propose a feedforward neural network to represent the wave function ψ. The network structure includes a single input x, one or two hidden layers with an activate function, and one single output. We define the loss function for the neural network solution as:

$$\arg\min_{\psi(x)} L_N(\psi(x)) = h \sum_{i=1}^{N} \left[\psi'(x_i)^2 + V(x_i)\psi(x_i)^2\right] \tag{14}$$

The lowest value of the loss function L_N found by the neural network will be the ground state energy E_{ground} corresponding to the ground state eigenfunction ψ_{ground}. The network's parameter, i.e., the set of weight layers, is the adjustable parameter α of ψ. Initially, the network parameter is initialized with random nonzero values. The feedforward neural network uses backpropagation to try to learn the wave function ψ that results in the lowest possible L_N value. To force the normalization constraint as in (12), we add a regularization term:

$$\lambda[h \sum_{i=1}^{N} \psi(x_i)^2 - 1]^2$$

where $h = \frac{2L}{N-1}$ and an arbitrarily large penalty term λ to the loss function L_N to ensure ψ satisfies the normalization constraint.

The final loss function becomes:

$$L_N(\psi(x)) = h \sum_{i=1}^{N} [(\psi'(x_i)^2 + V(x_i)\psi(x_i)^2] + \lambda[h \sum_{i=1}^{N} \psi(x_i)^2 - 1]^2 \tag{15}$$

3 Experiments and Results

We use both the Finite Difference and the Neural Network method to solve the Schrödinger equation (2) with the general form of potential function $V(x)$:

$$V(c_0, \alpha, M)(x) = c_0 + \sum_{i=1}^{M} \xi_i c_i \cos\left(\frac{i\pi x}{L}\right) \tag{16}$$

where

$$\xi_i \sim \mathcal{N}(0,\, 1),$$
$$c_i = (\frac{L}{i\pi})^{\alpha}$$

We choose the interval $[-10, 10]$, and discretize the interval into $N = 512$ points. The penalty term λ is set to 1000.

In the experiment, we initiate 6 different instances of potential function $V(x)$, *f1* to *f6*. The first three instances, *f1*, *f2*, and *f3*, have parameters $M = 5$ and $\alpha = 2$, the remaining instances, *f4*, *f5*, and *f6*, have parameters $M = 10$ and $\alpha = 2$.

We use the PyTorch machine learning framework to develop the neural network solution. The network uses the gradient-based training process with Adam optimizer with a learning rate of 8.10^{-3}. For the construction of the hidden layer, we try different numbers of layers, different width sizes for each layer, and different activation functions to find the optimal solution. The trained models and results are uploaded to my GitHub.

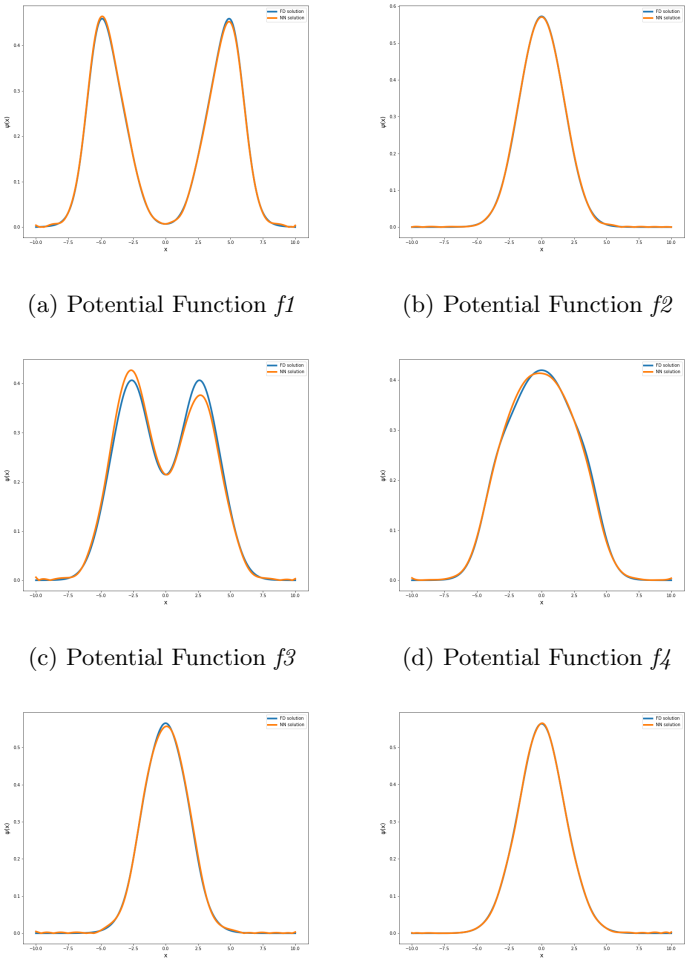
From the empirical result, we discovered that the network architecture resulting in the most accurate solutions to the FD solutions is a fully-connected feedforward neural network with **one input neuron**, **two hidden layers with the width of 60 neurons each**, and **one output neuron**. Each hidden layer uses the $sin()$ activation function, which helps the eigenfunctions converge to the right shape faster comparing to other more popular activation functions, such as $sigmoid()$ or $tanh()$. The accuracy of such network architecture is reflected in Table 1. The ground state eigenvalues found by the NN method are consistently closed to the ground state eigenvalues found by the FD method, with percentage errors less than $7{\times}10^{-2}$. The eigenfunction mean-squared errors are consistently low, under 10^{-4}.

Function instances	**Eigenvalue error (%)**	**Eigenfunction MSE**
f1	0.029	8.17e-06
f2	0.025	2.38e-06
f3	0.066	0.00016105
f4	0.067	2.72e-05
f5	0.077	1.17e-05
f6	0.045	1.84e-06

Table 1: Results of the NN method with best accuracy architecture

We run the batch gradient descent training process, with varied number of iterations:

1. For one-extremum eigenfunction, the loss value converges to the correct eigenvalue after 12,000-15,000 iterations.



(a) Potential Function *f1*

(b) Potential Function *f2*

(c) Potential Function *f3*

(d) Potential Function *f4*

(e) Potential Function *f5*

(f) Potential Function *f6*

Figure 1: Solutions using the Finite Difference and Neural Network methods (blue: FD solutions, orange: NN solutions)

2. For multiple-extremum eigenfunction, the training process might need to run 30,000-45,000 iterations to get a more accurate eigenfunction and the corresponding loss value result.

4 Conclusion and Future Research

In this study, we successfully develop a neural network solution for the one-dimensional eigenvalue differential equation. Our work shows the effectiveness of using neural networks to solve differential eigenvalue problems. Our work can help to study and solve a more generalized form of differential eigenvalue problem, the Sturm-Liouville problem. This study can be generalized to solve differential eigenvalue problems in higher dimensions.

References

[1] H. Jin, M. Mattheakis, and P. Protopapas. Physics-informed neural networks for quantum eigenvalue problems. *arXiv preprint arXiv:2203.00451*, 2022.

[2] I. Lagaris, A. Likas, and D. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[3] S. o. P. The University of Edinburgh. The variational principle. https://www2.ph.ed.ac.uk/~gja/qp/qp8.pdf.