Machine-Learned Wavefunctions

Github Code

Amro Imam*
Center for Quantum Information Physics, New York University

The probability distribution of a quantum mechanical system is of great utility for modelling and understanding the effects particles undergo under various types of potentials. Modelling the wavefunction ψ of a system is therefore of significant importance. This project presents an alternative method for finding the wavefuction, without solving the Schrodinger equation. We make use of machine learning models to find optimal parameters and hyperparameters for different trial modelling functions to represent what would be experimental observations of a quantum mechanical system. Polynomial and radial basis functions were used in this implementation, and were extended to systems of two dimensions. Successful replications of probability distributions under the potentials of the infinite well for one and two dimensions, as well as the harmonic oscillator in one dimension were achieved using this method with a negligible mean square error loss.

Introduction

The wavefunction of a quantum mechanical can be simply found if the Schrodinger equation (1) is solved.

$$\frac{-\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} + V\psi = E\psi \tag{1}$$

However, quite often, solving this second-order differential equation is not trivial. An alternative method is therefore necessary to be able to model an observed probability distribution of **any** quantum mechanical system. Machine learning offers a very powerful (and often not very well understood) arsenal of tools that have gained a lot of traction over the past two decades for its applications in classification, sentiment analysis, computer vision, etc. In this project, however, we are interested in more of a rudimentary method; regression. The purpose of regression, and its nature, are as follows: We have a particular relationship between multiple variables and an output that depends on these variable in a particular way. Regression is an attempt to find the best possible representation/description of this relationship. This process entails multiple steps:

1. Choose modelling function (linear, polynomial, exponential, etc.)
2. Find optimal hyperparameter (polynomial degree, variance)
3. Find optimal model parameters

In classical machine learning, which is what we implement here, it is necessary to first choose a family of functions which is believed to be a good candidate to describe the relationship between the variables of interest (position and probability density for our purpose). This, however, is not a necessity for a machine learning model that makes use of neural networks, and the reason for that can be better understood by first understanding how we assess the quality of our predictions. This is done using what is called a loss function. This is a measure of how far, or close the predictions that we have made using our model are to the true values in the distribution. For most purposes a mean square error loss function $L(y, f(x,\theta))$ (2) is reasonable:

$$\begin{aligned} L(y_i, f(x_i,\theta)) &= \frac{1}{2}(yi - f(x_i,\theta))^2 \\ R &= \frac{1}{N}\sum_{i=1}^{N} L(x_i,\theta) \end{aligned} \tag{2}$$

where y_i is the corresponding output (label) to our data vector x_i. And $f(x_i,\theta)$ is the model function we arbitrarily chose that depends on x_i and the model parameters θ. R is the risk, and is simply the average of the loss incurred using our model. Different values of our hyperparameter, will provide different risk values. The aim is then to optimize our choice to achieve the lowest risk. Finally we find the optimal parameters θ^*. The following derivation [1] provides us with a simple equation (3) for calculating the optimal parameters of a polynomial model by minimizing the risk:

$$\begin{aligned} \nabla_\theta\left(\frac{1}{2N}\|\mathbf{y}-\mathbf{X}\theta\|^2\right) &= 0 \\ \frac{1}{2N}\nabla_\theta\left((\mathbf{y}-\mathbf{X}\theta)^T(\mathbf{y}-\mathbf{X}\theta)\right) &= 0 \\ \frac{1}{2N}\nabla_\theta\left(\mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\theta + \theta^T\mathbf{X}^T\mathbf{X}\theta\right) &= 0 \\ \frac{1}{2N}\left(-2\mathbf{y}^T\mathbf{X} + 2\theta^T\mathbf{X}^T\mathbf{X}\right) &= 0 \\ \mathbf{X}^T\mathbf{X}\theta &= \mathbf{X}^T\mathbf{y} \\ \theta^* &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{3}$$

It is worth noting that $\mathbf{y}$ and $\mathbf{X}$ are vector representations of the labels and data respectively.

Methods

The purpose of this method is to utilize raw experimental data to model the probability distribution. However, for the sake of proof of concept, it is sufficient to perform the same exercise on data generated from our analytical solutions to the Schrodinger equation. We can further randomly sample from this generated distribution to better emulate true experimental observations. For the purpose of demonstration, we will discuss three different potentials. From there we can start playing around with our data to find the necessary parameters for our model. In 2 of our examples we assume a polynomial modelling function (4), and in the last one, a radial basis function (RBF) [2] (5).

$$f(x;\theta) = \sum_{p=1}^{P} \theta_p x^p + \theta_0 \tag{4}$$

$$f(\mathbf{x};\theta) = \sum_{k=1}^{N} \theta_k \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{x}_k\right\|^2\right) \tag{5}$$

The task at hand now it to first find the optimal hyperparameter d^* (optimal polynomial degree) and then model-parameter vector θ^* using the d-degree polynomial-transformed data vector $\mathbf{X}$ and output vector $\mathbf{y}$. The same process can be extended to another hyperparameter such as σ for the RBF with the replacement of step 2 by the step of computing the kernel matrix as described by Equation (5). A description of the algorithm used is found below:

Algorithm 1 Training Algorithm
Input: $\mathbf{X}, \mathbf{y}$
Output: d^*, θ^*
Initialisation: d-range = (0,18), a=1, n=2
LOOP Process
1: **for** $d = 0$ to 18 **do**
2: Poly transform $\mathbf{X}$ to d-degree
3: Calculate $\theta_d = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$
4: Calculate $R_d = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{y} - f(\mathbf{X}, \theta_d))^2$
5: Store R_d in R_{All}
6: **if** R_d is $min(R_{All})$ **then**
7: $d^* = d$
8: $\theta^* = \theta_d$
9: **end if**
10: **end for**
11: **return** d^*, θ^*

The algorithm describes a process where, in step 1, we begin looping through an arbitrarily defined range for our hyperparameter, polynomial degree for the polynomial modelling, and σ for the RBF. Step 2 then entails transformation of the input data in a way to befitting to the number of parameters in each of our constructed models. For the polynomial case, the number of parameters is equal to the optimal hyperparameter. For the RBF case, the number of parameters is N, the number of your input data vectors. In step 3, we very conveniently utilize the expression for the calculation of the model parameters, and in step 4, using those parameters to predict, and then compare our prediction with the labels of the data by averaging the risk/loss using mean square error. Prediction using the current model parameters can be made using $\hat{\mathbf{y}} = \mathbf{X}\theta^*$. If our risk happens to be the lowest value we have encountered so far, we can then conclude that the optimal hyperparameter was found and from it, retrieve the optimal model parameters.

Results

The Infinite Well - 1 Dimension

Consider the wavefunction solution to the Schrodinger equation for the 1-D infinite well:

$$\psi(x) = \sqrt{\frac{2}{a}} sin\left(\frac{2\pi x}{a}\right) \quad n = 1, 2, 3, ... \tag{6}$$

The corresponding probability distribution can be seen in Figure 1.

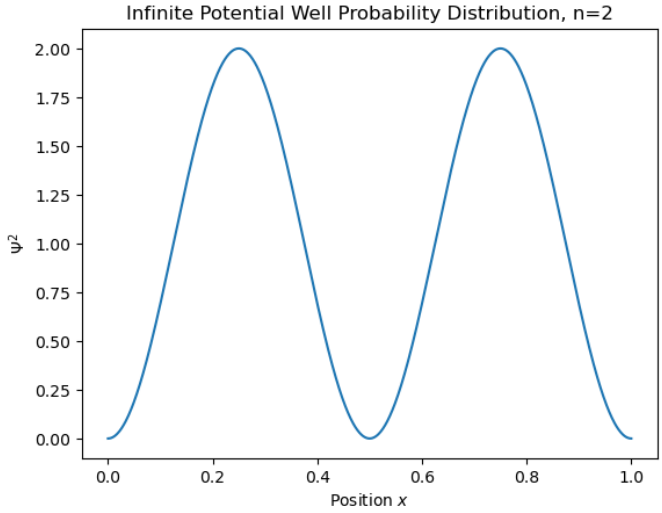

FIG. 1. The probability distribution of the 1-dimensional infinite potential well.

We can then sample from this distribution randomly to create a synthetic set of experimental observations as an approximation, as shown in Figure 2.

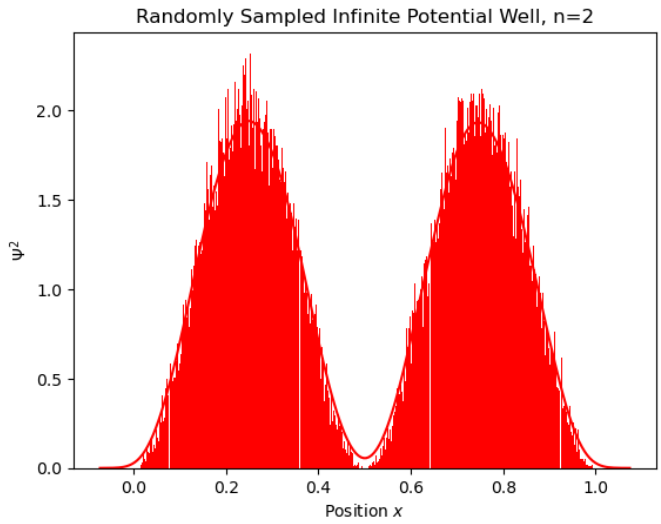

FIG. 2. A uniform, random sample from the analytical probability distribution of the 1-D infinite well.

The results of running the algorithm for hyperparameter optimization are shown in Figure 3. A polynomial of degree 10 gave us the best accuracy, i.e. the lowest risk/loss for our constructed model.

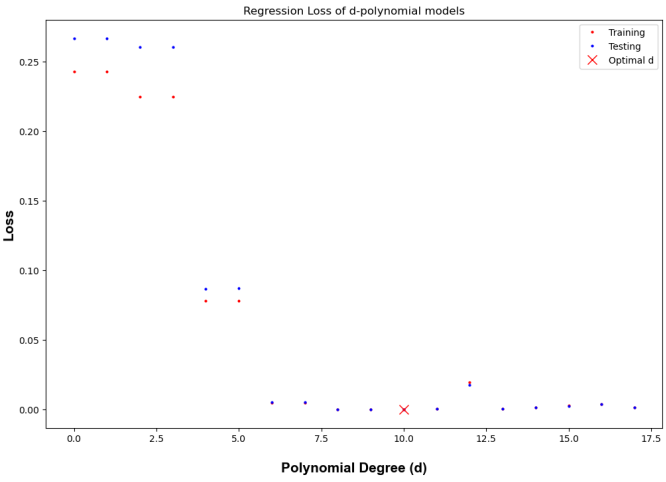

FIG. 3. Polynomial degree optimization for the 1-D infinite well. Note the interchangeable usage of the terms risk and loss.

Plotting our predictions against the original distribution, we find a great match as seen in Figure 4.

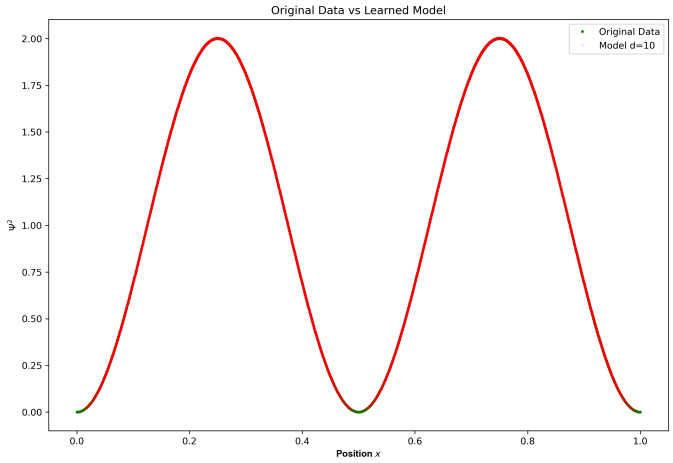

FIG. 4. An overlay of the predicted vs original distribution for the 1-D infinite well.

The Harmonic Oscillator - 1 Dimension

We next discuss a harmonic oscillator system in the second excited state ($n = 2$), in which a particle is trapped between position -1 and 1. An interesting question to ask is whether the probability distribution we see in Figure 5, represented by (7), can be represented differently purely based on our observation of the distribution.

$$\psi^2_{n=2} = \left\| \left(\frac{mw}{h\pi}\right)^{1/4} \cdot \left(\frac{1}{2}\right)^{1/2} \cdot e^{\left(\frac{-mwx^2}{2h}\right)} \right\|^2 \tag{7}$$

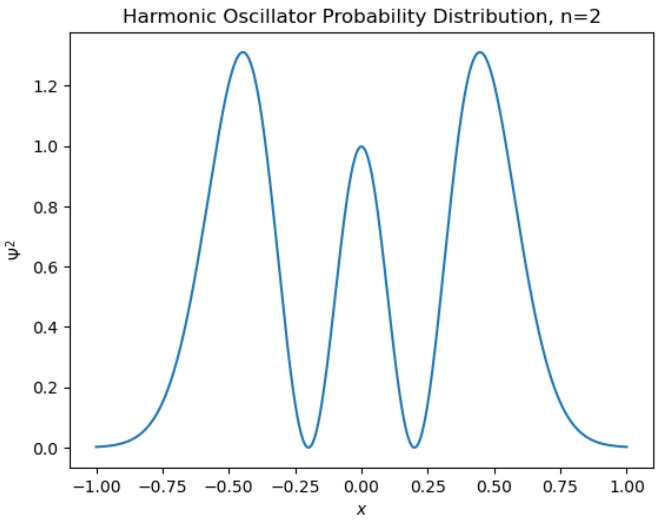

FIG. 5. The probability distribution of the 1-D harmonic oscillator potential.

In fashion identical to what we did with the 1-D infinite well, we repeat the sampling process (Figure 6),

hyperparameter optimization (Figure 7) and then fitting using our optimized model (Figure 8).

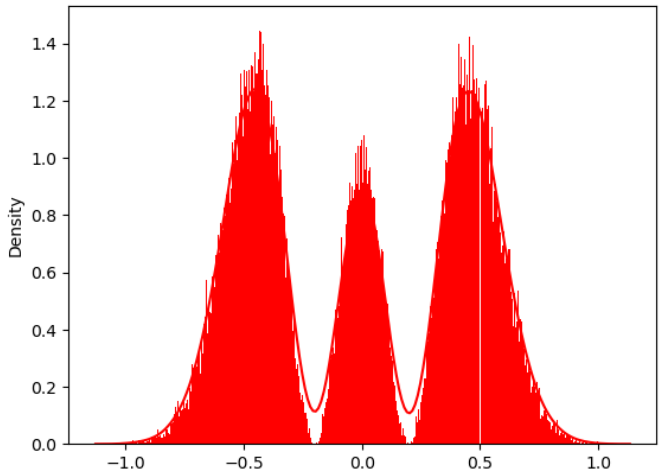

FIG. 6. A uniform, random sample from the analytical probability distribution of the 1-D harmonic oscillator potential.

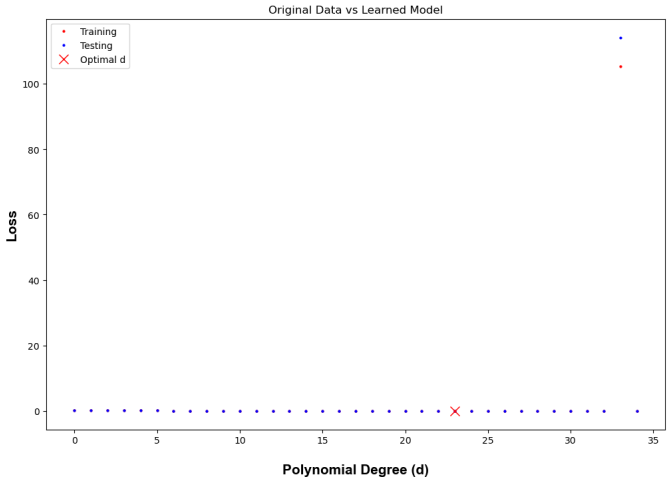

FIG. 7. Polynomial degree optimization for the 1-D harmonic oscillator potential.

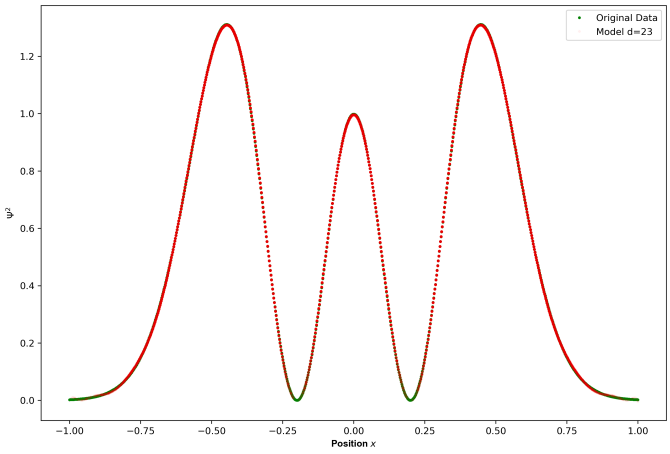

FIG. 8. An overlay of the predicted vs original distribution for the 1-D harmonic oscillator potential.

After having confirmed the accuracy of our fit using our prediction model, we can easily form an expression representing this probability distribution from our optimal parameters. Since we found that the polynomial degree 23 was optimal in this particular run, we can then create an expression comprised of 23 terms. An example of this is seen in Equation 8 below:

$$\psi^2_{n=2} = 0.996x^0 + 0.011x^1 + -61.64x^2 + -1.087x^3 + 1282.018x^4 29.311x^5 + -11016.962x^6 - 359.579x^7 + 54107.99x^8 + 2431.105x^9 + -171979.435x^{10} - 9996.17x^{11} + 371132.482x^{12} + 26332.093x^{13} - 549553.283x^{14} + -45412.089x^{15} + 549635.701x^{16} + 51031.095x^{17} + -354489.357x^{18} + -35997.078x^{19} + 132947.528x^{20} + 14471.787x^{21} - 22006.044x^{22} - 2529.398x^{23} \quad (8)$$

Quite the hefty expression compared to the much neater analytical solution in (7), but approximately identical nonetheless.

The Infinite Well - 2 Dimensions

We now extend this procedure to a multi-dimensional system. The changes needed to this method due to the increase in dimensions of the feature space, is, nothing. The only consideration needed to be made is whether a polynomial modelling function is still a viable option for our purposes. Therefore, this subsection first examines the results with the polynomial model (4), and then the radial basis function model (5).

In this example we study the situation where the principal quantum numbers are $n_x = 2$, and $n_y = 3$ as shown in Figure 9.

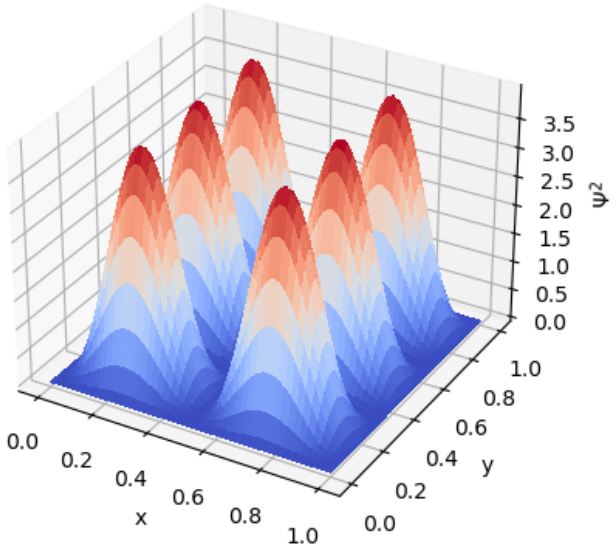

FIG. 9. The probability distribution of the 2-D , $n_x = 2$ and $n_y = 3$ infinite well potential.

Using a 2-D feature space, we once again optimize for the hyperparameter, 12 being the answer in this particular run.

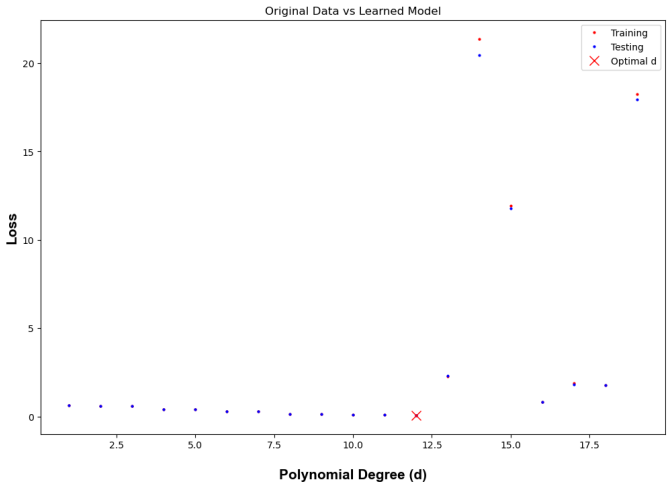

FIG. 10. Polynomial degree optimization for the 2-D infinite well.

From there, we create our predictions using $\hat{\mathbf{y}} = \mathbf{X}\theta^*$, where $\hat{\mathbf{y}}$ is the output vector of size $Nx1$, $\mathbf{X}$ is the data vector, now with 2 features (x and y coordinates), transformed to a polynomial of degree 10 and hence of size $Nx66$, and finally the optimal parameters vector θ^*, of size $2x66$. The number of terms is 66 because the transformation of a 2 features (x_1 and x_2) to 10 dimensions has $\sum_{i=1}^{10}(2i+1)$ terms:

$$x_1, x_2, {x_1}^2, x_1 \cdot x_2, {x_2}^2, {x_1}^2 \cdot x_2, ..., {x_2}^{10}$$

The `sklearn.preprocessing.PolynomialFeatures` package [3] was used to construct this transformed feature space.

Our prediction yields something seemingly similar but far from a good approximation. This can be seen from the leakage into the negative domain of ψ^2, as well as the non-uniformity of the peaks in Figure 11.

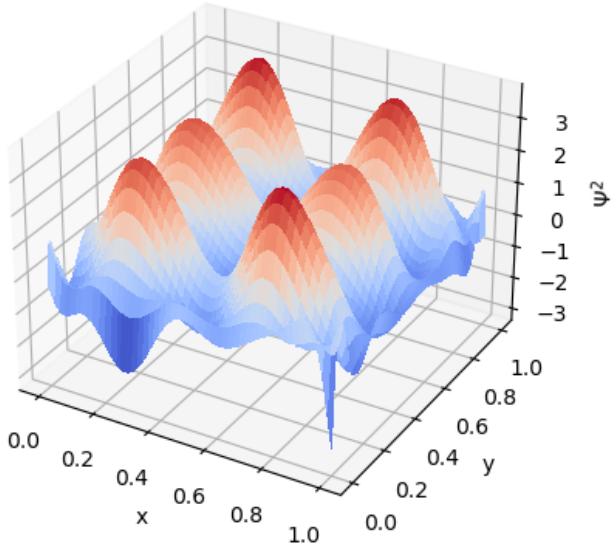

FIG. 11. Prediction of 2-D infinite well probability distribution using the parameter-optimized polynomial modelling function.

Given that this modelling function was not a good choice, we trial the RBF. This first entails creating a kernel matrix from the N vectors in our input using (5). An attempt to do this manually was done as could be found in the code linked to this report, but was extremely inefficient and would take hours to run. The `sklearn.metrics.pairwise.rbf_kernel` [3] package was used to efficiently construct the kernel matrix.

Next, we optimize the hyperparameter σ, the standard deviation of the modelling bumps. The optimization result favours a smaller σ as seen in Figure 12.

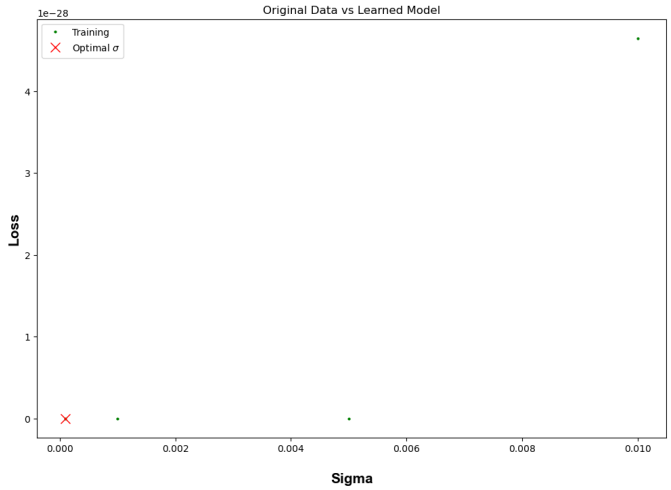

FIG. 12. Hyperparameter σ optimization for the 2-D infinite well.

The prediction using the RBF model (Figure 13) yields substantially better results than that of the polynomial (Figure 11), and almost identical to the original distribution (Figure 9).

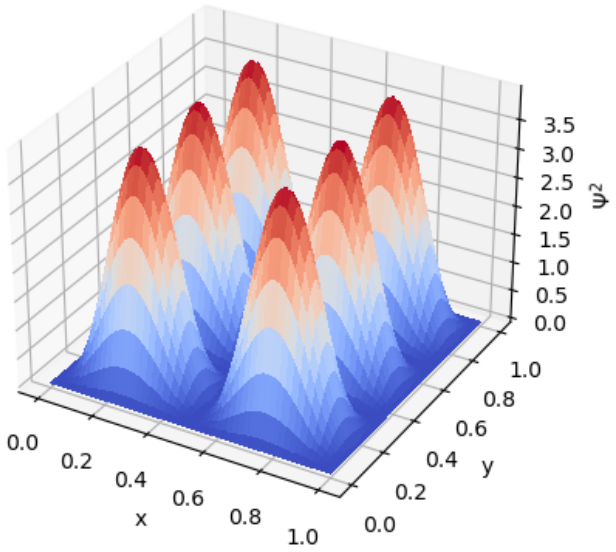

FIG. 13. Prediction of 2-D infinite well probability distribution using the parameter-optimized RBF modelling function.

Conclusion

We have discussed how to construct a viable, classical machine learning model to learn an expression for the probability distribution for quantum mechanics systems under the 1-Dimensional (1-D) infinite well, the 1-D harmonic oscillator and the 2-D infinite well. We found that a polynomial modelling function was sufficient to accurately replicate the distributions for the 1-D potentials. However, for the 2-D infinite well, the radial basis function was more fitting. We were also able to retrieve an expression for the probability density ψ^2 for the harmonic oscillator potential in the second excited state ($n = 2$) using the optimized model created.

Further work on this project would include an extension of the feature space to 3-D potentials. An attempt at utilizing deep learning can also be made as well for probability distributions that do not seem to match with conventional modelling functions. Access to laboratory data of different systems would also add further legitimacy to the concept.

* aes924@nyu.edu

[1] T. Jebara, Machine learning 4771.

[2] A Primer on Kernel Methods, in *Kernel Methods in Computational Biology* (The MIT Press, 2004) https://direct.mit.edu/book/chapter-pdf/232731/9780262256926_cab.pdf.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research **12**, 2825 (2011).