Visualizing a neural network that develops quantum perturbation theory


As Published
http://dx.doi.org/10.1103/PhysRevA.98.010701

Publisher
American Physical Society

Version
Final published version

Accessed
Fri Dec 07 22:13:04 EST 2018

Citable Link
http://hdl.handle.net/1721.1/117207

Terms of Use
Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.
Visualizing a neural network that develops quantum perturbation theory

Yadong Wu,1 Pengfei Zhang,1 Huitao Shen,2 and Hui Zhai1,3

1Institute for Advanced Study, Tsinghua University, Beijing 100084, China
2Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA
3Collaborative Innovation Center of Quantum Matter, Beijing 100084, China

(Received 14 March 2018; published 30 July 2018)

Motivated by the question whether the empirical fitting of data by neural networks can yield the same structure of physical laws, we apply neural networks to a quantum-mechanical two-body scattering problem with short-range potentials—a problem that by itself plays an important role in many branches of physics. After training, the neural network can accurately predict s-wave scattering length, which governs the low-energy scattering physics. By visualizing the neural network, we show that it develops perturbation theory order by order when the potential depth increases, without solving the Schrödinger equation or obtaining the wave function explicitly. The result provides an important benchmark to the machine-assisted physics research or even automated machine learning physics laws.

DOI: 10.1103/PhysRevA.98.010701

Introduction. Human physicists have made great achievements in discovering laws of physics during the last several centuries. Based on experimental observations, they interpret data and build theories using logical inference and rigorous mathematical reasoning. Artificial intelligence, which has recently been extensively applied to physics research [1–37], interprets data empirically by, for example, fitting on statistical models.

While physicists hope that the artificial intelligence could help discover new physics in unexplored areas, there are certainly challenges to overcome due to the different nature of “knowledge” among human physicists and artificial intelligence. First, the great complexity of machine learning models, such as deep neural networks with more than millions of fitting parameters, makes it difficult for humans to understand whether machines have successfully captured the underlying laws of physics. Second, despite the great success of artificial intelligence in many disciplines outside physics [38], its weakness [39,40] leads people to believe that it lacks the ability of sophisticated reasoning [41]. Therefore, it is important to ask the question whether these two approaches of distilling knowledge from data will yield the same answer when facing the same problem. The positive answer will mark an important step forward to the machine-assisted physics research or even automated machine learning physics laws.

Motivated by this question, we let the machine supervisedly learn a quantum-mechanical scattering problem. We show that the machine could not only correctly produce the scattering length for a given interaction potential, but also give a human tractable decision-making process. More concretely, the machine will develop perturbation theory order by order with increasing the potential strength.

Two-body scattering problem. The scattering problem plays an important role in many branches of physics, including atomic and molecular physics, nuclear physics, and particle physics [42]. In this work, we focus on the two-body scattering problems in three dimensions with spherical potentials.

Consider the Schrödinger equation in the relative frame

$$\left( -\frac{\hbar^2}{2\tilde{m}} \nabla^2 + V(r) \right) \Psi = E \Psi,$$

where $\tilde{m}$ is the reduced mass of two particles and $V(r)$ is the interaction potential. The wave function can be expanded through partial wave decomposition [43] as

$$\Psi(r) = \sum_{l=0}^{+\infty} \frac{\chi_l(r)}{kr} \mathcal{P}_l(\cos \theta),$$

where $k$ is the wave vector and $l$ labels different angular momentum. $\chi_l$ is the radial wave function and $\mathcal{P}_l(\cos \theta)$ represents the angular part. For s-wave scattering, one can show that the asymptotic behavior of the radial wave function $\chi_{s,l=0} \propto \sin(kr + \delta_s)$. We can therefore determine the important quantity of the s-wave scattering length $a_s$ as $\tan \delta_s = -ka_s + o(k^2)$ [43].

In this way, we establish a mapping between $V(r)$ and $a_s$, which is to be learned by the neural network. In the following, we first train neural networks with $V(r)$ as the input and $a_s$ as the output. Here $a_s$ is obtained by numerically solving the Schrödinger equation (see Appendices). In practice, $a_s$ can be extracted from experimental data of low-energy collision. After training, the neural network can predict $a_s$ directly from $V(r)$.

Perturbation theory. Before presenting the results of neural networks, let us first review a textbook method for computing the scattering length. One can compute the scattering amplitude $T$ analytically through perturbative Born expansion [43]. In fact, the perturbation theory is by far the most well-established approach to compute observables in interacting quantum mechanics or quantum field theory. Particularly in our two-body scattering problem, $T$ is given by $T = V + VG_0V + VG_0VG_0V + \cdots$, where $G_0$ is the Green’s function for free Hamiltonian with $V(r) = 0$. Keeping only the s-wave component and taking the low-energy limit, we obtain $a_s = 2\pi^2 \langle 0 | T | 0 \rangle$, where $| 0 \rangle$ denotes the zero-momentum state.
different number of neurons discretized at \( N_0 \) different points. (c) The error \( \epsilon \) of the neural network after training as a function of maximum scattering length \( \tilde{a}_s \), with different number of neurons \( N_1 \) in the first hidden layer. See main text for the definition of \( \epsilon \) and \( \tilde{a}_s \).

Here and throughout the rest of this work, we take \( m = \hbar = 1 \) for simplicity. Combining these equations, we get $a_s = a_s^{(1)} + a_s^{(2)} + \cdots$, with

\[
a_s^{(1)} = 2\pi^2 \langle 0 | V | 0 \rangle = \int V(r) r^2 dr
\]

as the first-order Born approximation and

\[
a_s^{(2)} = 2\pi^2 \langle 0 | V G_0 V | 0 \rangle = -\frac{1}{2} \int V(r) V(r') K(r, r') dr dr'
\]

as the second-order Born approximation. Here

\[
K(r, r') = rr'(r + r' - |r - r'|).
\]

The Born expansion is based on the potential strength, and is a very good approximation for weak potentials.

**Neural network.** In order to keep the size of the input data finite, we consider short-range attraction potentials \( V(r) \), i.e., \( V(r) < 0 \) for \( r < R \) and \( V(r) = 0 \) for \( r \geq R \). For simplicity, we restrict the depth of \( V(r) \) to be before the first resonance so that \( a_s < 0 \). We first generate the potential randomly (see Appendices) and then discretize it uniformly along \( r = 0 \) to \( r = R \) into \( N_0 \) pieces, as shown in Fig. 1(b). The input data is thus an \( N_0 \)-dimensional vector

\[
V = \left[ V \left( \frac{R}{N_0} \right), V \left( \frac{2R}{N_0} \right), \ldots, V(R) \right]^T.
\]

Before the first resonance, the larger the potential depth, the larger the absolute value of scattering lengths in general.

We use a fully connected neural network as our machine learning model, whose structure is schematically shown in Fig. 1(a). It is composed of three fully connected layers. The effect of the first (hidden) layer can be mathematically summarized as \( Y = f(WV + B) \), where \( V \) is the \( N_0 \)-dimensional input vector defined previously, \( Y \) is an \( N_1 \)-dimensional output vector, \( W \) is an \( N_1 \times N_0 \) weight matrix, and \( B \) is an \( N_1 \)-dimensional bias vector. \( f \) is the activation function that is applied elementwise to the vector and is chosen to be rectified linear units \( f(x) = \max(0, x) \). A similar hidden layer taking \( Y \) as its input comes after the first hidden layer. Its weight matrix is of size \( N_2 \times N_1 \), and its output is linearly mapped to the final single value output, which is interpreted as the scattering length \( a_s \). In our work, we set \( N_0 = 64 \), \( N_2 = 32 \) fixed and change \( N_1 \) of the neural network.

We train the neural network by minimizing the mean squared error on \( 3 \times 10^4 \) randomly generated \( \{V, a_s^T(V)\} \) pairs in units of \( 1/R^2 \), \( R \), where \( a_s^T(V) \) is the scattering length obtained from solving the Schrödinger equation given discretized potential \( V \). In the training data set, \( a_s^T(V) \) is uniformly distributed in the range of \( (\tilde{a}_s, 0) \). We set \( L2 \) regularization strength to be 0.1 to prove weight matrix visualization. After training, the neural network should be able to make a prediction of the scattering length, denoted as \( a_s^T(V) \), directly from the input potential \( V \). This prediction may deviate from the true scattering length \( a_s^T(V) \). We characterize the performance of the trained neural network by calculating the averaged error \( \epsilon \) defined as

\[
\epsilon = \frac{1}{N_t} \sum_i \left| \frac{a_s^T(V) - a_s^T(V)}{a_s^T(V)} \right|,
\]

on a test set of similar \( \{V, a_s^T(V)\} \) pairs. Here \( V' \) denotes the \( l \)th potential in the test set and \( N_t = 10^3 \) is the size of the testing set. As shown in Fig. 1(c), the network can always predict the scattering length \( a_s \) with high accuracy for different \( \tilde{a}_s \) and with different number of neurons in the first layer (denoted by \( N_1 \)).

To understand how the neural network computes the scattering length, it is very informative to visualize the weight matrix \( W \) in the first hidden layer that maps the input data \( V \) to the first intermediate vector \( Y \). We plot \( W \) in Fig. 2 for
two neural networks trained on a very small magnitude of \( \tilde{a}_s/R = -0.1 \) and on a relatively large magnitude of \( \tilde{a}_s/R = -0.5 \), respectively. For both networks, most of their matrix elements automatically vanish and are thus not responsible for producing the scattering length. In the following, we analyze the remaining matrix elements for these two scenarios.

**First-order Born approximation.** Let us first focus on the case of shallow potentials. For convenience, we represent \( W \) using \( N_0 \)-dimensional row vectors, known as “neurons”: \( W = (\xi_1, \xi_2, \ldots, \xi_{N_0})^T \). The effect of the first (hidden) layer can then be represented as \( Y_{(1)} = f(\mathbf{J} \mathbf{V} + \mathbf{B}) \). Each neuron in this layer is denoted as \( \xi_n \). From Fig. 2(a), one can see that nearly all nonvanishing \( \xi_n \) behave similarly. A typical \( \xi_n \) is plotted in Fig. 2(c).

Importantly, we find that typical \( \xi_n \) can be quite accurately described by \( \sim j^2 \). As shown in Fig. 2(c), \( \xi_n \) and normalized \( \xi \propto (1, 4, \ldots, N_0^2) \) correspond very well. This means that the \( \alpha \)th neuron in the first hidden layer performs the calculation

\[
Y_{\alpha} = \sum_j \xi_{\alpha,j} V_j \propto \sum_j j^2 V \left( \frac{JR}{N_0} \right).
\]

This is precisely the discrete version of the first-order Born approximation shown in Eq. (3). If one uses a similar neural network to study scattering volume or supervolume for \( p \)-wave or \( d \)-wave scattering, one obtains the same result for the shallow potential as \( \sim j^2 \) or \( \sim j^6 \), respectively, which are both consistent with the first-order Born approximation (see Appendices).

**Second-order Born approximation.** As the potential depth increases, more features begin to emerge in \( W \). There are many neurons whose behavior cannot be fitted by \( j^2 \) discussed in the previous section, shown in Figs. 2(b) and 2(d). It is natural to suspect that this new neuron pattern comes from the second-order Born approximation. To verify this, we first notice that the key observations so far can be summarized as follows:

(i) When the potential is shallow, the neural network only captures the first-order perturbation as it is already accurate enough to reach a certain level of accuracy. (ii) As the potential becomes deeper, the neural network gradually develops the structure to capture at least the second-order perturbation. It is quite remarkable that, by empirical fitting, the neural network develops the same kind of perturbation theory that was developed by human physicists. At least in this example, we understand how the neural network works and provides a positive answer to the question raised in the Introduction, namely, how human physicists can yield the same answer when facing the same problem.

**An empirical formula.** Last but not least, we bring out a different aspect: If one can tolerate a certain amount of error, the neural network can also map an empirical formula that works quite well even beyond the perturbative regime, although the formula lacks rigorous mathematical reasoning.

Here the formula is inspired by the fact that even if we reduce the first hidden layer to have only one neuron, the neural network can still predict scattering lengths quite accurately with errors only increasing by a small amount [Fig. 1(c)]. In Fig. 4(a), we compare \( W \) with \( \xi \) and \( \tilde{\xi} \). After proper scaling, they actually behave similarly. This implies that they

\[
\frac{\tilde{\xi}^T \mathbf{V}}{\Lambda_1 - \Lambda_2} \sim \frac{\xi^T \mathbf{V}}{\Lambda_1 - \Lambda_2}.
\]

FIG. 3. Neural network with the same architecture as that in Fig. 2, but trained with \( a_{(2)}^s \) as the output instead of \( a_{(1)} \). (a) Visualization of the matrix \( W \). (b) A typical vector \( \xi_{r=32} \) as a function of \( r/R = i/N_0 \) for \( \tilde{a}_s/R = -0.5 \). It is compared with the eigenvector of \( K \) matrix with the largest eigenvalue \( \sim 4 \) shown by the solid line. The inset of (b) is the eigenvectors associated with first two largest eigenvalues. For \( N_0 = 64 \), the blue solid curve shows \( \xi_1 \) with the largest eigenvalue \( \Lambda_1 = 29.5 \), and the orange dashed curve shows \( \xi_2 \) with the second largest eigenvalue \( \Lambda_2 = 1.8 \).
works in this simple problem can shed light on understanding more sophisticated problems better.

Acknowledgments. We thank Ning Sun, Ce Wang, and Jinmin Yi for helpful discussions. This work is supported by MOST under Grant No. 2016YFA0301600 and NSFC Grant No. 11734010. H.S. thanks IASTU for hosting his visit to Beijing, where the key parts of this work were done.

APPENDIX A: METHOD FOR GENERATING TRAINING DATA

In this Appendix we demonstrate in detail the method for generating our training data \( \{ \mathbf{V}, a_s(\mathbf{V}) \} \).

Generating potentials. To extract the universal method for solving a physical problem, the shape of the potentials in the training data should be diverse enough, so that the machine will not learn features specific to some potential shape. Therefore, we construct random potentials using random walks.

Formally, \( \tilde{V}(r) \), understood as a three-dimensional spherical potential, is constructed as

\[
\tilde{v}(r) = \begin{cases} 
V_i, & r_i \leq r \leq \min[r_{i+1}, R] \\
0, & r \geq R,
\end{cases}
\]

where \( V_i \) and \( r_i \) are generated by recurrence relations \( V_i = V_{i-1} + \delta V_i \) and \( r_i = r_{i-1} + \delta r_i \), with initial value \( V_0 \) uniformly distributed from \([-V_0/2, V_0/2] \) for some positive \( V_0 \) and \( r_0 = 0 \). \( \delta r_i \) and \( \delta V_i \) are all random variables; each is identically and independently distributed as \( \theta(\delta r_i)N(\delta r_i, 4R/N_0), \sqrt{2R/N_0} \) and \( N(\delta V_i, 0, V_0/10) \). Here \( N(x, \mu, \sigma) \) is the normal distribution with expectation \( \mu \) and standard deviation \( \sigma \). This procedure, on the one hand, introduces enough randomness, and on the other hand, gives a very smooth potential. Finally we get our purely attractive potential by \( V(r) = -|\tilde{V}(r)| \). A schematic of the generation of potential is shown in Fig. 5.

To feed the neural network with generated potentials, we discretize the potential \( V(r) \) uniformly from \( r = 0 \) to \( r = R \). As a result the input data are \( N_0 \)-dimensional vectors of

\[
\text{FIG. 5. Random square well potential generated by two kinds of Gaussian distributions.}
\]
the form

\[ V = \left[ V\left( \frac{R}{N_0} \right), V\left( \frac{2R}{N_0} \right), \ldots, V(R) \right]^T. \] (A2)

We take \( N_0 = 64 \) throughout this work.

*Generating scattering lengths.* After generating a potential, we calculate the corresponding scattering length by solving the Schrödinger equation using transfer matrix method [46] for the discretized potential \( V \) [using the original potential \( V(r) \) will introduce systematic error].

Within each square well \( r \in (r_{i-1}, r_i) \), the wave function is \( \phi_i = a_i j_l(k_i r) + b_i n_l(k_i r) \). For zero-energy scattering, we have \( k \sim \sqrt{-V} \) (\( m = \hbar = 1 \)). \( j_l, n_l \) are two spherical Bessel functions, and \( l \) labels the angular momentum and thus the partial wave. To avoid divergence at \( r = 0 \), \( \phi_0 = j_l(k_0 r) \). At the boundary of each square well \( r = r_i \), the wave function obeys continuity condition \( \phi(r_i^-) = \phi(r_i^+) \), \( \phi'(r_i^-) = \phi'(r_i^+) \). By matching the boundary condition, we can determine all \( \phi_i \) for \( r < R \).

Outside the short-range potential \( r \to R^+ \) the wave function is determined by scattering parameters:

\[
\begin{align*}
\text{s wave } : & \quad \psi_0 \propto 1 - \frac{a_s}{r} \equiv \phi_{0s}, \\
\text{p wave } : & \quad \psi_1 \propto \frac{x}{3v} - \frac{1}{x^2} \equiv \phi_{0p}, \\
\text{d wave } : & \quad \psi_2 \propto \frac{x^3}{15D} - \frac{3}{x^3} \equiv \phi_{0d}.
\end{align*}
\] (A3)

Finally, by matching the boundary condition at \( r = R \),

\[
\frac{\psi'_i}{\psi_i} = \frac{\phi'_s}{\phi_s} = \frac{\phi'_d}{\phi_d},
\] (A6)

we obtain the scattering length for different partial waves.

*Data postselection.* Before the first resonance, \( a_s < 0 \) becomes more negative when the potential becomes deeper. We generate potentials with different \( V_0 \), which determines the overall depth of generated potential. We then postselect preferred data to make our training set a uniform distribution \( a_s \in (\bar{a}_s, 0) \).

**APPENDIX B: TRAINING DETAILS**

We use Wolfram Mathematica [47] to train and evaluate neural networks. All the weights are randomly initialized to a normal distribution with Xavier method [48]. While optimizing parameters, we minimize the loss function

\[
L = \frac{1}{N} \sum \left[ a_s^x(V') - a_s^T(V') \right]^2
\] (B1)

using Adam algorithm with batch size 32. Here \( a_s^T(V) \) is the scattering length obtained using the method introduced in the last section. \( a_s^x(V) \) is the prediction made by the network and \( N \) is the size of the testing set. Considering there are thousands of parameters in this network, the training set contains \( N = 3 \times 10^4 \) \{\( V, a_s(V) \)\} pairs. In order to prove weight matrix visualization, we set \( L_2 \) regularization strength be 0.1, although there is no noise in the training set. We check that there is no overfitting by using a validation set of size \( N_v = 3 \times 10^3 \). It typically takes five epochs for the network to converge, meaning the training is very easy. This is actually a hint that the network may fit to a very simple function of \( a_s(V) \).

**APPENDIX C: FIRST-ORDER BORN APPROXIMATION FOR HIGH PARTIAL WAVES**

In this Appendix, we report that the neural network could capture first-order Born approximation for \( p \)-wave and \( d \)-wave wave functions.

In the main text the first Born approximation is shown in Eq. (3) for \( s \) partial wave. For \( p \)-wave and \( d \)-wave wave functions, the approximation becomes

\[
\begin{align*}
v &= \int V(r)r^4 dr, \\
D &= \int V(r)r^6 dr.
\end{align*}
\] (C1) (C2)

where \( v \) and \( D \) are scattering volume and supervolume, respectively. We generate training data in the same way, and train neural networks with architecture \( N_0 = 64, N_1 = 70, N_2 = 32, N_3 = 1 \). Figure 6 shows weight matrix \( W \) and the fitting result of a typical neuron in the first hidden layer. As one can see, neurons trained on \( p \)-wave data behave as \( r^4 \), and neurons trained on \( d \)-wave data behave as \( r^6 \), as expected from Eqs. (C1) and (C2).