May 29th Weekly Report

Erik Bigwood

May 30, 2025

Abstract

In total, this week I have extended my work to analyze the physics of two same-charged particles in a harmonic potential. Instead of learning the exact parameters k and α or fitting a polynomial over products of powers, I've used a Lagrangian containing a neural net coupling as shown later. This form has not converged well at all, so I broke it down to just fitting a neural net on the Coulomb interaction and found very poor performance.

Contents

1 Neural Net Structure

Basic neural nets I'll start by providing a quick summary of neural net structures. In this work I'll be restricting to 'multi-level perceptron' (MLP) networks[1]. These are the common feedforward-type networks with fully connected layers. More specifically, these are maps from an input space $\mathbb{R}^m$ to an output space $\mathbb{R}^n$ with l internal layers of N_k nodes per 'hidden' layer. Linking each layer is an array of weights $\sigma_{k,k+1}$, vector of biases $b_{k,k+1}$, and a non-linear activation function $f_{k,k+1}$. Each layer's state is contained in a vector of 'activations' x_k.

Evaluation of this neural network (NN) comes in three stages. To begin, propagate the input into the first hidden layer:

$$x_1 = f_{0,1}\Big(\sigma_{0,1}x_0 + b_{0,1}\Big)$$

Where x_0 is the input to the NN. For the remaining hidden layers, iterate the same procedure:

$$x_{k+1} = f_{k,k+1}\Big(\sigma_{k,k+1}x_k + b_{k,k+1}\Big)$$

For the final output, I apply the weights and biases but not the non-linear activation function to the activation of the final hidden layer x_l

$$x_{final} = \sigma_{l,final}x_l + b_{l,final}$$

For the remainder of this report, I'll stick to using the same non-linear activation function for all layers. In particular, I've chosen the rectified linear unit (ReLU) function, whose form is:

$$ReLU(x) = x \cdot (x > 0)$$

This is 0 for any $x \leq 0$ and x for any $x > 0$. ReLU is used very widely within industry as it doesn't suffer from the 'vanishing gradient' problem of other activation functions like the sigmoid, tanh, etc. functions, but can produce 'dead' neurons where the input to a neuron is always < 0. These situations produce neurons that effectively contribute nothing to the results and can be pruned away, but generally waste computational resources.

Compared to the exact Lagrangian, these neural nets consume massively increased amounts of computational resources. For a MLP mapping $\mathbb{R}^m \to \mathbb{R}^n$ with l hidden layers and N_k neurons per layer, the number of independent parameters Ω is:

$$\Omega = \Big[(m+1)N_1\Big]\Big[\sum_{k=1}^{l-1}(N_k+1)N_{k+1}\Big]\Big[(n+1)N_l\Big]$$

Assuming the same number of nodes at each layer, this becomes

$$\Omega = (m+1)(n+1)lN^2$$

Constructioning successful NN Lagrangians, like constructing any other machine learning model, becomes an exercise in balancing expressivity and computational constraints. Certain functions are much less representable via this method than others, as I will explore later in this report.

Loss functions Expanding to NNs also requires some restructuring of the training loss function. For smaller numbers of parameters and more inherent structure in the parameter meanings (i.e. each parameter corresponding to a term like $q_0^i q_1^j$), the general shape of the loss landscape should be relatively simple. With a NN, there are many more parameters (see above), and they do not correspond to anything concrete in the internal structure – they are simply numbers in a black box. As such, it can be quite helpful to add a 'regularization' term to the loss that penalizes any large parameter values.

In the remainder of this report, I'll be using the following loss function with regularization parameter λ:

$$L = (\ddot{q}_{fit} - \ddot{q}_{data})^2 + \lambda \left[\sum_{k=0}^{l} \sigma_{k,k+1} + \sum_{k=0}^{l} b_{k,k+1} \right]$$

2 Two particles in a harmonic potential with Coulomb repulsion, neural net Lagrangian

2.1 Basic results

Contrary to the previous results with the exact and polynomial Lagrangians, this NN Lagrangian approach for the system of two same-charged particles in a harmonic potential

2.1.1 Results

Appendix A Figures

A.1 Exact Lagrangian form

Appendix B Bibliography

References

[1] Yueyao Yu and Yin Zhang. "Multi-layer Perceptron Trainability Explained via Variability". In: (May 2021). URL: https://arxiv.org/abs/2105.08911v3.