NN-body: Approximating Solutions to the n-body Problem using Neural Networks

William H. Guss
Unviersity of California, Berkeley
Berkeley, CA 94720
wguss@ml.berkeley.edu

Abstract

The n-body problem, although simple in its formulation, is one of the most classic and unsolved problems in Newtonian mechanics. In this paper, we work towards a deeper understanding of the underlying dynamics of solutions by utilizing neural networks to approximate target states from initial conditions. Unlike previous numerical methods, neural networks are universal approximators which can capture and learn dynamics across variable timescales, without solving for intermediate solutions.

1 Background

Formally, the n-body problem considers the graviational dynamics of n different point masses in $\mathbb{R}^3$. Let m_i, p_i, and v_i denote the mass, position, and velocity of the ith point mass respectively. For every mass pair $i \neq j$, the force induced on mass m_i by mass m_j is given by Newton's law of gravitation

$$F_{ij} = G m_i m_j \frac{p_j - p_i}{\|p_j - p_i\|^3} \tag{1.1}$$

where G is the graviational constant. Intuitively, F_{ij} describes a force in the direction of p_j (relative to p_i) whose magnitude is dependent on both the distance and combined mass potential of the two point-masses.

Integrating all force data from (1.1) we yield that the acceleration applied to mass m_i is

$$\frac{d^2 p_i}{dt^2} = G \sum_{\substack{k=1 \\ k \neq j}}^{n} \frac{m_j (p_j - p_i)}{\|p_j - p_i\|^3}, \tag{1.2}$$

where the acceleration is yielded by means of Newton's first law. In vector form,

$$\begin{aligned} \frac{d^2 p}{dt^2} &= G \left(\left\langle \left(p^{(i)} - p_i \right) \odot \frac{1}{\|p - p_i\|^3} \;\middle|\; m^{(i)} \right\rangle \right)_{i=1}^{n} \\ &= G \tilde{M} \cdot \left(\left(p^{(i)} - p_i \right) \odot \frac{1}{\|p - p_i\|^3} \right)_{i=1}^{n} =: G \tilde{M} \Pi(t) \end{aligned} \tag{1.3}$$

where $m^{(i)}, p^{(i)}$ are the full mass and position vectors with ith element removed, and $\tilde{M}$ is a matrix whose ith row is $m^{(i)}$.

In its full statement, the n-body problem can be stated as follows: given arbitrary initial position and velocity vectors, p_0 and v_0, does there exist an analytical solution $\mathcal{P}$ to $\frac{d^2 p}{dt^2} = G \tilde{M} \Pi(t)$?

Submitted as the final project for ASTRO 9.

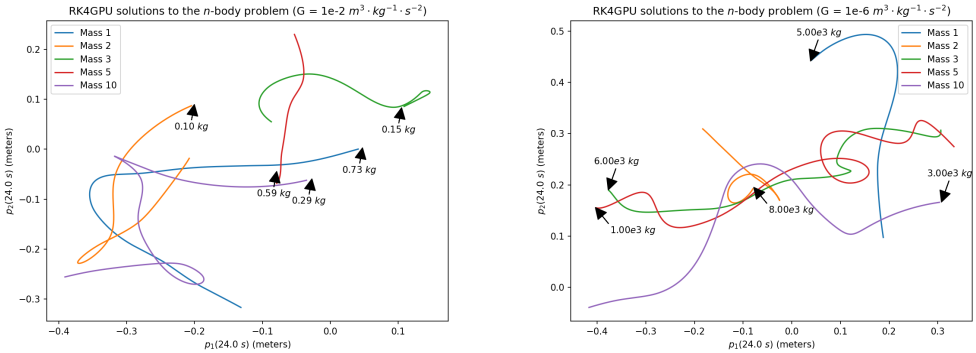

Figure 1: Seemingly chaotic solutions to the n-body problem with different iniital conditions and graivational constants as solved by RK4. Note the legends ommit some masses as activity localizes away from the origin with their initial conditions.

2 Learning the n-body problem

As of yet, there is no analytical solution to the arbitrary n-body problem, which given its simple formulation, raises questions as to our understanding of the complex dynamical patterns which emerge therefrom. Although numerical methods for solving (1.3) are sufficiently powerful for visualizing solutions, they are limited in their capacity for analysis; that is, on face value, such methods attempt to yield the temporally global solution by repeatedly and accurately modeling the local one.

In this paper, we approach numerical approximation from the alternative perspective of machine learning. Instead of iteratively arriving at a global solution, we will *learn* the map $\pi : (p_0, v_0, t) \mapsto (p(t), v(t))$ using a family of *hypothesis* function approximators whose analytical forms are condusive to statistical and spectral analysis globally in time. Formally we will solve the following optimization problem, usually called a *classification problem*.

Let $\mathcal{D}$ be the distribution induced by the jointly uniform random variables P_0, V_0, T_0. Given some hypothesis class $\mathcal{H} = \{h_\theta : \mathbb{R}^{3n} \times \mathbb{R}^{3n} \times \mathbb{R} \times [t_0, \infty) \to \mathbb{R}^{3n} \times \mathbb{R}^{3n}\}$ parameterized smoothly by $\theta \in \Theta$, we wish to find θ^* such that

$$\theta^* = \arg\min_{\theta \in \Theta} \mathop{\mathbb{E}}_{(p_0, v_0, t) \sim \mathcal{D}} \mathcal{L}\Big(\pi(p_0, v_0, t), h_\theta(p_0, v_0, t)\Big), \tag{2.1}$$

where $\mathcal{L}$ is some monotonic function called a *loss*. Usually we take $\mathcal{L}$ to be the ℓ_2 norm. For intuition, note that if $h_{\theta^*} = \pi \in \mathcal{H}_\theta$ then the minimal expectation above is zero.

In the foregoing regime, if we find a hypothesis class with which is sufficiently expressive, then the solution h_{θ_*} will approximate solutions to the n-body problem with arbitrary accuracy. It remains, however, to find such a hypothesis class $\mathcal{H}$ which also has the desired interpritability conditions and can yield novel statistical insights into the n-body problem. Luckily, artifical neural networks come close.

2.1 Deep Learning

Artificial neural networks (ANNs) are extremely powerful collection of machine learning algorithms, whose structure models the biological neuron closely. Over the past twenty years, the fields of computer vision and natrual language processing have seen exponential progress in solving machine perception as a result of deep learning.

In its simplest form, deep learning considers the hypothesis class of all ℓ-layer neural networks, $\mathcal{N}_\ell$, such that if $N : \mathbb{R}^n \to \mathbb{R}^m \in \mathcal{N}_\ell$ then $N = N_\ell$ and the following recursion relation is defined for all $1 \leq j \leq \ell$,

$$N_j(x) = \sigma\left(W_j N_{j-1}(x) + \beta_j\right); \quad N_1(x) = x \tag{2.2}$$

with matrices $W_\ell \in \mathbb{R}^{n_j \times n_{j-1}}$, vectors $\beta_j \in \mathbb{R}^{n_j}$, and $n_1 = n, n_\ell = m$. For the purposes of this work, it suffices to think of deep learning algorithms as functions which perform linear regression (W, β), apply some non-linearity (σ), and repeat this process ℓ times.

The recent success of deep learning is in its name, the greater ℓ (the deeper the network), the more expressive power $\mathcal{N}_\ell$ has. In the context of the n-body problem, $\mathcal{N}_\ell$ is a *universal approximator* for $\ell \geq 2$ and therefore is a proper candiate for the aforementioned hypothesis class. This means that given any continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$ and any $\epsilon > 0$ there exists an $N \in \mathcal{N}_\ell$ such that $\|N - f\|_\infty < \epsilon$.

Beyond expressivity, neural networks allow for inspection and interpretation of the mapping f being learned via examination and analysis of the $\theta^* = (W, \beta)$ learned in (2.1). In particular, one can analyze which states $x \in \mathbb{R}^n$ produce an output $y \in \mathbb{R}^m$ by updating some random initial x_0 so as to minimize $\|y - N(x)\|$. For the n-boduy problem, we therefore can determine eigenstates $(N(x) = \lambda x)$ for the system and other such fixed points merely by applying this optimization method. Although there are many other such tools for inspecting the mappings learned in $\mathcal{N}_\ell$, we shall restrict our attention to the foregoing.

2.2 Training Neural Networks

Before we procede, we need specify if the optimization in (2.1) is even possible when $\mathcal{H} = \mathcal{N}_\ell$. Luckily, the observation that when σ is twice–differentiable, as a function of (W, β), $N(W, \beta; x)$ is twice-differentiable for all x. As such, we then can seek local minima using *gradient descent.* In the language of machine learning, the gradient descent algorithm changes the parameters θ such that $\mathcal{L}$ decreases in the direction of greatest change; this direction of greatest decrease is the gradient $\nabla_\theta \mathcal{L}$. Therefore we define the *gradient descent update rule* as follows,

$$\theta_{t+1} = \theta_t - \lambda \frac{\partial L(\theta_t)}{\partial \theta} \tag{2.3}$$

for some *learning rate*, $\lambda \in \mathbb{R}$.

In the context of neural networks, we need only compute $\partial \mathcal{L}/\partial W$ and $\partial \mathcal{L}/\partial \theta$ to move towards local optima. It's important to note that there are no guarentees that (2.3) will converge to the global optima unless the function being optimized is *g-convex* (geodesically convex). Despite this, the field of deep learning can attribute much if not all of its recent success to the empircal efficacy of (2.3) in training different neural networks, and therefore we shall assume convergence properties.

2.3 NN-body Architecture

To approximate solutions to (1.3), we will choose a subset of $\mathcal{N}_\ell$ over which to perform the optimization given in (2.1). In deep learning, selection of different subsets of the foregoing hypothesis class, called architecutres, can have a significant impact on the accuracy of the models learned via gradient descent. The architecture of the neural network choses is as follows. The first layer has 700 neurons, followed by two 400 neuron layers, followed by two 300 neuron layers, and then the output layer. Every layer uses $\sigma = \tanh$ as an activation funciton. The network is trained using the Adam Optimzier with a learning rate of 1e-3. Some networks with residual layers were considered, but those networks did not yield convergence.

3 Results

With the learning problem defined, we now pose several questions as to approximation of the n-body problem via neural networks.

First, is it possible to build some nueral network such that the loss on predictions of solutions far in the future is low and yet the computational complexity of the algorithm subsumes standard solvers in performance? In other words, is it possible to learn *one-shot*[1] prediction of solutions t-timesteps in the future with subquadratic complexity? Second, using the methods described in section 2.1, what insights arise from analysis of eigenstates in these learned models, and how well do these eigenstates coorespond with true eigenstates of the n-body problem?

[1] One-shot prediction means that the neural network will directly output the position and velocities of the n-bodies given some desired future time step t, *without* calculating the intermediate steps.

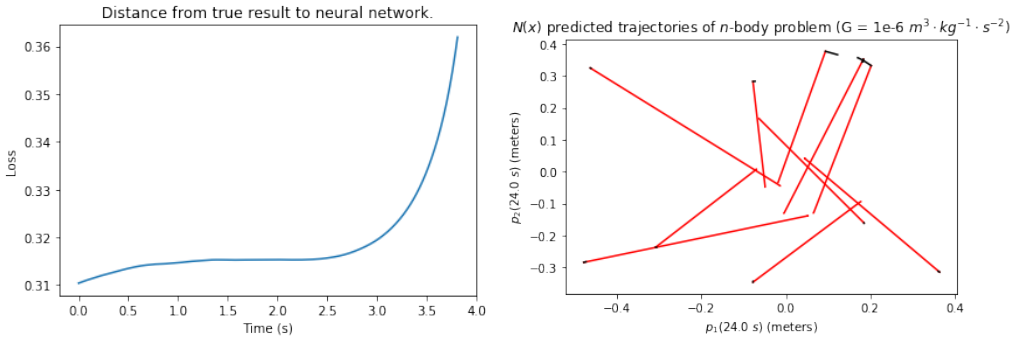

Figure 3: Left: The loss between ground truth solution to the $n = 3$-body problem and that predicted by the best neural network. Right: The predicted trajectory of the n-body problem at $t = 0.7s$ for one-shot and iterative, red and black respectively.

One-Shot Prediction. In the first experiment, we train the NN-body architecture using gradient descent against data sampled from π. We then measure performance by comparing the average loss of the neural network against the RK4 solutions to (1.3) on random initial conditions (P_0, V_0, T_0), called a *test set*, never shown to the neural network. Furthermore, we restrict the test set to particular timesteps and consider the performance degredation of the algorithm as $T_0 \to \infty$.

Specifically, we train the neural network by drawing samples of $x_0 := (P_0, V_0, T_0)$ from a jointly uniformly random distribution. The samples are then solved by an RK4 solver applied to (1.3) implemented on a first GPU, and then the list of time-steps $\Gamma(x_0) = \{(P_\tau, V_\tau, h\tau)\}_{t=1}^{T_0/h}$, where h is the stepsize for RK4, is then consolidated into initial-final pairs; that is, we let $B(x_0) := \{x_0\} \times \Gamma(x_0)$ where $\times$ is the cartesian product. The product $B(x_0)$ now consists of input and desired output pairs for which we will minimize $\mathcal{L}(N(x), y)$ when $(x, y) \in B(x_0)$. We repeat this data generation and processing step untill the neural network N, trained on a second GPU, reaches sufficiently low error.

As seen in Figure 2, the rate of convergence (learning) to a local optima decreases as the number of bodies increase. This namely is because as n increases, the non-linearity of the system so too explodes. In Figure 2, we normalize each error rate to the number of bodies to show how on average each body diverges from the ground-truth in position and velocity. Although there is a clear degredation in performance, the error rates observed are still sufficient to perfrom one-shot prediction.

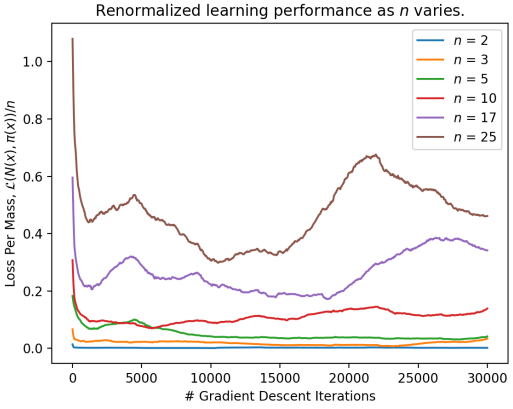

Figure 2: Left: A plot of loss (2.1) as gradient descent is applied to $\mathcal{N}_\ell$ for various different n.

Next, and most importantly, Figure 3 shows the relationship between error (on the test-set) for one-shot prediction as time increases. As expected, an increase in both time and number of bodies indicates and increased uncertainty in the system. Although the uncertainty should naturally arise as a result of the zero probability density, ergodicity of n-body problem, it remains to ascertain whether or one-shot prediction with the learned neural network leads to less uncertainty than the iterative approach. To determine the foregoing comparison, we visually compare the trajectories generated by varying the time input T_0 to the neural network and those generated by iteratively by the neural network. Demonstrated by Figure 3 (Right), and the one-shot algorithm predicts the target position correctly without error propagation.

Eigenstate Analysis. With the NN-body learned, and one-shot prediciton sufficient to a given time, we now attempt to generate states which satisfy $N(x, t_{period}) = \lambda x$; that is, we analyze the states for which N predicts a scalar cyclicity. For $\lambda = 1$, these states are surely period t_{period} fixed points of N. By drawing initial $x := (P_0, V_0)$ pairs, fixing $T_0 := t_{period}$, and performing gradient descent on

$\mathcal{L}(N(x,t),x)$ with respect to x, we yield some sample fixed points. Although time-limits placed on the project prevented further analysis, such an approach for determining fixed-points of dynamical systems other than the n-body system seems wrought with potential.

4 Appendix

Figure 4: The neural network architecture of NNBody as displayed by tensorboard. Each fully connected layer is a weight matrix multiply followed by a non-linearity.