Proposal

Erik Bigwood

April 23, 2025

Abstract

Proposal for a project regarding autograd implementation of Lagrangian mechanics in Python/torch, and extensions to learning Lagrangians from noisy data via modern machine-learning based techniques.

1

How do I want to format this? Different possible avenues to go down.

1.1 Toy model

Toy model – harmonic oscillator? particle in an EM field? something easy to produce and rely on to get the initial point across and demonstrate the guts of how this will work.

Start with initial conditions, calculate time derivatives, step forward (start with basic Euler integration), calculate derivatives, etc.

Compare speed of autograd method to numerically integrating equations of motion – I expect this to be very slow. In general, this method should really become useful when applied to Lagrangians that are too complex to derive equations of motions by hand.

1.2 Learning Lagrangians

Extension to learning Lagrangians from data.

Create synthetic data from autograd forward model, use that to train a trial Lagranian density

$$\mathcal{L} = f(q, \dot{q}) \tag{1}$$

Define $Q(\mathcal{L}[q_i, \dot{q}_i], q_i, \dot{q}_i)$ as the output $\ddot{q}$ from the autograd method

Given a set of synthetic data $(q_i, \dot{q}_i, \ddot{q}_i)$, minimize the following simple L^2 loss function:

$$\Big(\ddot{q}_i - Q(\mathcal{L}[q_i, \dot{q}_i], q_i, \dot{q}_i)\Big)^2 \tag{2}$$

Learning method Given an untrained Lagrangian $\mathcal{L}\left[q, \dot{q}, \vec{p}\right]$ with parameters $p_i \in \vec{p}$, dataset of N points $(q_i, \dot{q}_i, \ddot{q}_i)$, and learning rate δ:

$$\epsilon = \sum_{i=1}^{N} \Big(\ddot{q}_i - Q(\mathcal{L}[q_i, \dot{q}_i], q_i, \dot{q}_i)\Big)^2 \tag{3}$$

$$\frac{d}{dt}\vec{p} = \nabla_{\vec{p}}\epsilon \tag{4}$$

$$\vec{p}_{k+1} = \vec{p}_k - \delta\nabla_{\vec{p}}\epsilon_k \tag{5}$$

$$\tag{6}$$

Full routine For a known Lagrangian $\mathcal{L}_{known}$, sample N points in phase space to generate a dataset $\big(q_i, \dot{q}_i, Q(\mathcal{L}_{known}[q_i, \dot{q}_i], q_i, \dot{q}_i)\big)$ with q and $\dot{q}$ requires_grad == True.

Define internal structure of a new, untrained Lagrangian, for example:

$$\mathcal{L} = \sum_{k=0}^{N_1} a_k q^k + \sum_{l=1}^{N_2} b_l \dot{q}^l$$

$$\mathcal{L} = \sum_{k=0}^{N_1}\sum_{l=1}^{N_2} \Big(a_k b_l q^k \dot{q}^l\Big)$$

$$\mathcal{L} = f_1\Big(\sigma_1 f_0\big(\sigma_0(q, \dot{q}) + c_0\big) + c_1\Big)$$

Where a_k, b_l, c_i, σ_i are parameters, and $f_i(x)$ are non-linear activation functions.

For each iteration, calculate $Q(\mathcal{L}[q_i, \dot{q}_i], q_i, \dot{q}_i)$, loss function$\epsilon_k$ and its derivate w.r.t $\vec{p}$, then step $\vec{p}$ forward.

Repeat this iteration until reasonable convergence is achieved. Note – it is likely best to try a modern optimzer like Adam to improve performance here, especially given consideration of the potential roughness of this space.

1.3 Numerical considerations

Numerical methods relevant content

How stable can this be? How much noise can it tolerate? For some simple system (harmonic oscillator?), what does the optimization landscape look like? Are there local minima that can trap the system and block a good solution? How to avoid that?

How much error does this produce? Can this kind of system reliably capture conservation laws? How fast can we make this? Does running on GPU improve performance? How much noise can this tolerate? How rough is the parameter fitting space for a given class of Lagrangian? Can this be extended to accurately capture interactions between particles, and/or particles and fields? Can we extend this to capture quantum dynamics from measurement data? How would that work? [2]

Loss landscapes A major consideration in any gradient descent-based technique is the existence of local minima and roughness in the landscape of loss function values. As an example, take the harmonic oscillator Lagrangian[3]:

$$\mathcal{L} = \frac{1}{2}m\dot{q}^2 - \frac{1}{2}kq^2 \tag{7}$$

$$\ddot{q} = -\frac{k}{m}q \tag{8}$$

This has two learnable parameters m and k. For this system, $Q(\mathcal{L}[q_i, \dot{q}_i], q_i, \dot{q}_i)$ depends on the ratio $\frac{k}{m}$, so this landscape should reduce to a projective plane. To avoid this sublety, consider fixing either k or m and learning the other parameter.

To investigate the performance of this learning method, calculate $Q(\mathcal{L}[q_i, \dot{q}_i], q_i, \dot{q}_i)$ for a large number of sample points in phase space (either by the autograd method or analytically) with a fixed (m, k). Then, plot the loss function for a grid of points in (m, k) space to visualize the landscape to be explored by the optimizer.

1.4 NN Lagrangians

Extensions to neural network Lagrangians What classes of systems can this learn given a certain NN architecture? What is out of reach?

How do symmetries of the system in question show up in the NN? Do we get any new symmetries within the NN representation, and how do we find that (beyond the basic permutation symmetry that NNs have...)

How does this algorithm deal with potential issues hidden outside the sampled area? How much information is lost, and how catastrophic is that missing information?

1.5 Deliverables

Class scope Produce a basic model to demonstrate as the project. Pick a couple of simple Lagrangians and simulate them. Compare error and speed for traditional differentiation vs. autograd.

Compare results to exact solutions for a few simple problems – nD harmonic oscillator, particle in magnetic field, etc.

Optional extension – learning a simple Lagrangian from synthetic data generated by toy model. Potentially characterize noise tolerance.

Written paper detailing findings and future directions.

Presentation to class (per project spec document, 5 minutes)

2 Derivations

2.1 Lagrangian

For Lagrangian mechanics[3], we have the action

$$S = \int_{t_0}^{t_f} \mathcal{L}(q_i, \dot{q}_i dt = \int_{t_0}^{t_f} \big(T - U\big) dt \tag{9}$$

From Hamilton's principle we get the Euler-Lagrange equation (derivation from [1]):

$$\frac{d}{dt} \nabla_{\dot{q}} \mathcal{L} = \nabla_q \mathcal{L} \tag{10}$$

Where $(\nabla_{\dot{q}})_i \equiv \frac{\partial}{\partial \dot{q}_i}$. Expanding the time derivative through, we get

$$\big(\nabla_{\dot{q}} \nabla_{\dot{q}}^T \mathcal{L}\big) \ddot{q} + \big(\nabla_q \nabla_{\dot{q}}^T \mathcal{L}\big) = \nabla_q \mathcal{L} \tag{11}$$

$$\ddot{q} = \big(\nabla_{\dot{q}} \nabla_{\dot{q}}^T \mathcal{L}\big)^{-1} \big[\nabla_q \mathcal{L} - \big(\nabla_q \nabla_{\dot{q}}^T \mathcal{L}\big) \dot{q}\big] \tag{12}$$

With $\big(\nabla_q \nabla_{\dot{q}}^T \mathcal{L}\big) = \frac{\partial^2 \mathcal{L}}{\partial q_j \partial \dot{q}_i}$.

2.2 Analytical equations of motion

Harmonic Oscillator Take the harmonic oscillator Lagrangian

$$\mathcal{L} = \frac{1}{2} m \dot{q}^2 - \frac{1}{2} k q^2 \tag{13}$$

Apply the Euler-Lagrange equations[3]:

$$\frac{\partial \mathcal{L}}{\partial q} = \frac{d}{d} \frac{\partial \mathcal{L}}{\partial \dot{q}} \tag{14}$$

$$-kq = \frac{d}{dt} m \dot{q} \tag{15}$$

$$\ddot{q} = -\frac{k}{m} q \tag{16}$$

References

[1] Miles Cranmer et al. "Lagrangian Neural Networks". In: (Mar. 2020). eprint: `2003.04630`. URL: `https://arxiv.org/pdf/2003.04630.pdf`.

[2] P. A. M. Dirac. "The Lagrangian in Quantum Mechanics". In: *Physikalische Zeitschrift der Sowjetunion* 3 (1933), pp. 64–72.

[3] John R Taylor. *Classical Mechanics*. Ed. by Lee Young. 1st ed. Mill Valley, CA: University Science Books, 2005.