参赛队员姓名：詹有丘

中学：上海交通大学附属中学

省份：上海市

国家/地区：中国

指导教师姓名：__

论文题目：A multifunctional hamiltonian mechanics simulator and some of its application examples

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员：詹有丘　　指导老师：__

2020 年 9 月 10 日

Self-supervised hamiltonian mechanics neural networks

詹有丘 (Youqiu Zhan)

Thursday 10th September, 2020

Abstract

TBD

Keywords— hamiltonian, machine learning, mechanics

Contents

1 Introduction

When one observes the motion of a system, he may be curious about what is the hamiltonian of it if such a hamiltonian exists.

Here there have been several approaches to do this, such as the hamiltonian neural network [4], the lagrangian neural network [3], and the neural ODE [2]. Like what we are going to do, they optimize a neural network that reads the state $\mathbf{x}$ of a system as the input and outputs the change of state w.r.t. time $\mathrm{d}\mathbf{x}/\mathrm{d}t$ of the system.

In this process, they have a common property: they train the neural network in a supervised mode, which means there are $\mathrm{d}\mathbf{x}/\mathrm{d}t$ ground truth in the dataset. However, in a physics experiment, measuring the change rate of the state is probably more difficult than directly measuring the state. Thus, we want to introduce a method to optimize the neural network in a self-supervised way.

2 Canonical equation and ODE

To build a common sense of the physics theories it is going to involve, the canonical equation is introduced.

The canonical equation is a set of ordinary differential equations (ODE) whose solution depicts the motion of the system. The equation in mathematical form is [5][1, p. 65][7, p. 132]

$$\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}, \tag{1}$$

where $\mathbf{q} \in \mathbb{R}^n$ is the **generalized coordinates**, $\mathbf{p} \in \mathbb{R}^n$ is the **generalized momentum**, and $\mathcal{H}$ is the **hamiltonian** of the system, which is a scalar function w.r.t. t, $\mathbf{q}$, and $\mathbf{p}$. n is the number of degrees of freedom (DOF). A hamiltonian is specific for a specific system.

The tuple $\mathbf{x} := (\mathbf{q}, \mathbf{p}) \in \mathbb{R}^{2n}$ is called the **canonical coordinates**. In computer programs, it is convenient to write Equation 1 in form of

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}), \tag{2}$$

which is the common form of ODE. Here in our specific case,

$$\mathbf{f}(t, \mathbf{x}) := \boldsymbol{\omega} \nabla_{\mathbf{x}} \mathcal{H}, \tag{3}$$

where the notion $\boldsymbol{\omega}\nabla_{\mathbf{x}}$ denotes the **symplectic gradient** w.r.t. $\mathbf{x}$, whose first n components is the gradient w.r.t. the last n components of $\mathbf{x}$, and the last n components is the negative gradient w.r.t. the first n components of $\mathbf{x}$.

One of properties of the symplectic gradient is that, moving along the symplectic gradient field of a scalar does not change the value of the scalar function, which means that the value of $\mathcal{H}$ is conserved if $\partial\mathcal{H}/\partial t = 0$ [1, p. 67][7, p. 132]. In fact, the physical meaning of $\mathcal{H}$ is the energy, so its conservation is obvious.

According to Equation 3, the difference between $\mathbf{x}$ at 2 different times is an integral

$$\mathbf{x}(t_2) = \mathbf{x}(t_1) + \int_{t_1}^{t_2} \mathbf{f}(t, \mathbf{x}(t))\,\mathrm{d}t. \tag{4}$$

The integral can be calculated using the torchdiffeq Python package [2].

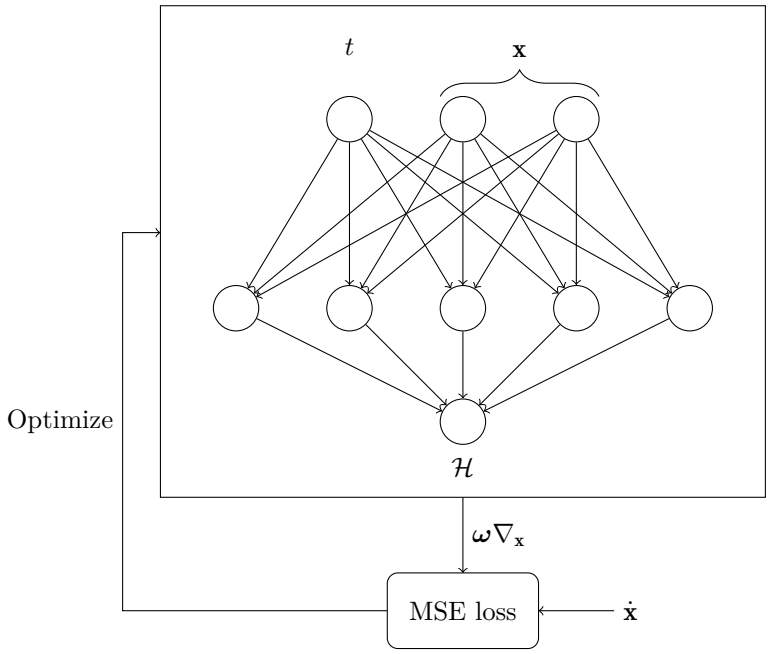


Figure 1: The train circle of a supervised hamiltonian neural network [4]

3 The training

Our goal is to derive the function $(t, \mathbf{x}) \mapsto \mathcal{H}$ according to the dataset containing a series of samples in form of $(t, \mathbf{x})$ on a series of possible motions of the system. To make our work simpler, we assume that $\partial\mathcal{H}/\partial t = 0$, which fits with most cases.

The dataset does not contain the $\dot{\mathbf{x}}$ infomation, which acts as the ground truth in the supervised model [4]. Our model is self-supervised, and thus does not need the $\dot{\mathbf{x}}$ infomation.

The model uses the loss inspired from Equation 4

$$\mathcal{L} := \mathrm{MSE}\left(\mathbf{x}(t_1) + \int_{t_1}^{t_2} \boldsymbol{\omega}\nabla_{\mathbf{x}}\mathcal{H}\mathrm{d}t, \mathbf{x}(t_2)\right), \tag{5}$$

where $(t_1, \mathbf{x}(t_1))$ and $(t_1, \mathbf{x}(t_1))$ are 2 samples from the same motion of the system. The complete process of a traning circle is shown in Figure 2. For comparison, the train circle of the supervised hamiltonian neural network is shown in Figure 1.

The Adam optimizer [6] is used for optimizing the neural network.

4 Few-parameters optimization

If the form of the hamiltonian is previously known, and there are just a few parameters in the formula of the hamiltonian to be decided, the few-parameters optimization.

For example, there is a harmonic oscillator with its frequency unknown. Its hamiltonian is

$$\mathcal{H} = \frac{1}{2}p^2 + \frac{1}{2}\omega^2 q^2, \tag{6}$$

where ω is the only parameter that should be optimized.

We can use the procedure explained in Section 3 to optimize the parameter.

Here we have gone on an experiment. Use the ground truth $\omega = 2$, and initialize the parameter as $\omega = 1$. Use a dataset with only 2 datas $\mathbf{x}(0) = (1, 0)$, and $\mathbf{x}\left(\frac{\pi}{2}\right) = (-1, 0)$. After we trained it for 1000 epoches with learn rate 1×10^{-3}, the parameter is optimized to the true value 2.

The change in loss and ω during the training is shown in Figure 3.

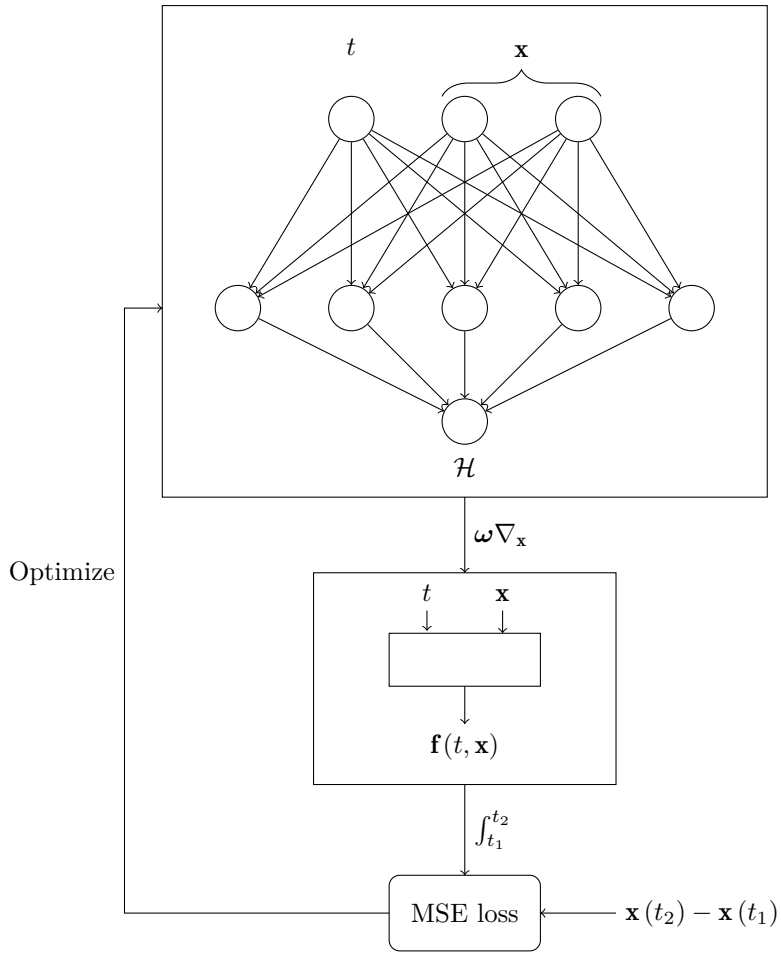


Figure 2: The train circle of a self-supervised hamiltonian neural network

Although the few-parameters optimization experiment does not have much value because we often do not know the form of the hamiltonian before we train the model, this experiment shows that the method introduced in Section 3 is feasible.

However, this method has some flaw. If we take the initial value of ω as 5.0, the final value of ω is not correct. This is because the loss has multiple minimals w.r.t. ω [8, p. 121], one of which is 2, taken when $\omega = 4$. Specially, in some cases, the wrong minimal of loss can be 0, so we cannot judge whether the result of the optimized parameter is correct by looking at whether the loss is 0.

This reminds us that the method has its limitations. To avoid the wrong result, the dataset should be comprehensive enough. After some trials, we found that the best strategy is to switch to another data when the loss converges. After switching the data several times, the parameters should converge to a correct value.

5 Tasks

5.1 Free particle

A free particle is a system with 1 DOF whose hamiltonian is [7, p. 133][1, p. 66]

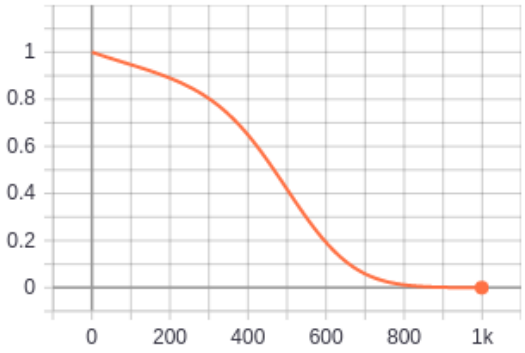
$$\mathcal{H}(t, q, p) := \frac{p^2}{2m},$$

where m is the mass of the particle. To be simple, we take $m := 1/2$.

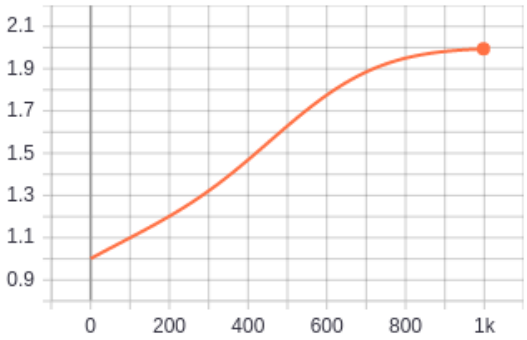
The model of a free particle is the simplest model. We take this model as the first task to test the basic capability of our idea.

The dataset is generated using a ODE solver. The dataset has totally 16 datas. Every data consists of the size of considered time interval (lies in the range $(0, 2)$), a series of 200 different initial conditions (lies in the range $(-1, 1)^2$) to be learned as a batch, and the corresponding state of the system at the end of the time interval.

Every data is learned for 160 epoches. When a data is finished, the neural network moves on to the next data. Thus there are totally 2560 steps. The loss change in shown in Figure 5.

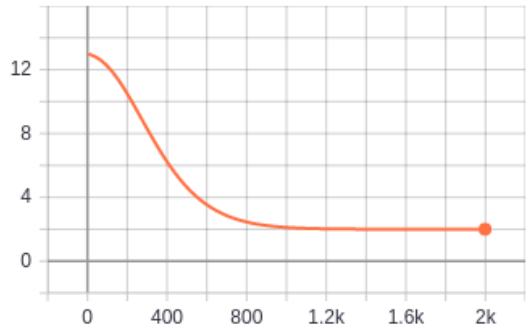


(a) The change in loss during the training

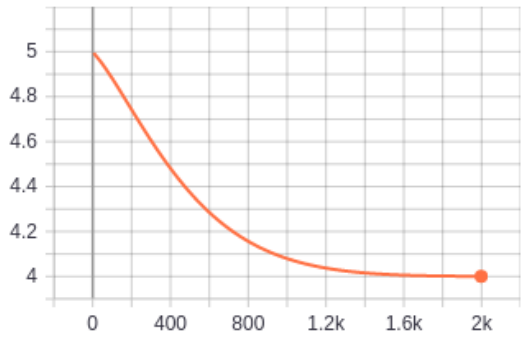


(b) The change in ω during the training

Figure 3: The process of training using the few-parameters optimization (abscissa is the epoch number)



(a) The change in loss during the training, converging to a non-zero value



(b) The change in ω during the training, converging to a wrong answer

Figure 4: The process of training using the few-parameters optimization, but coming out with a bad result

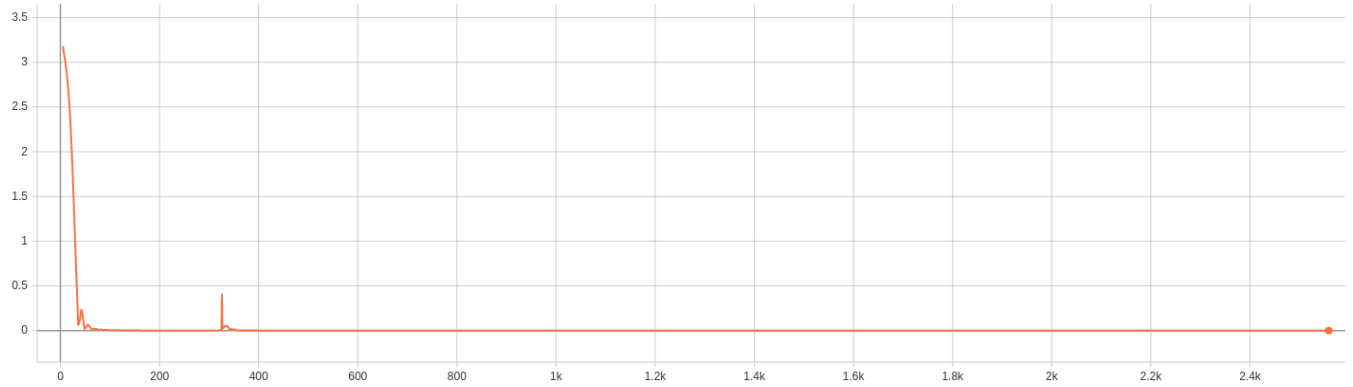


Figure 5: The loss change during the training of the model of free particle

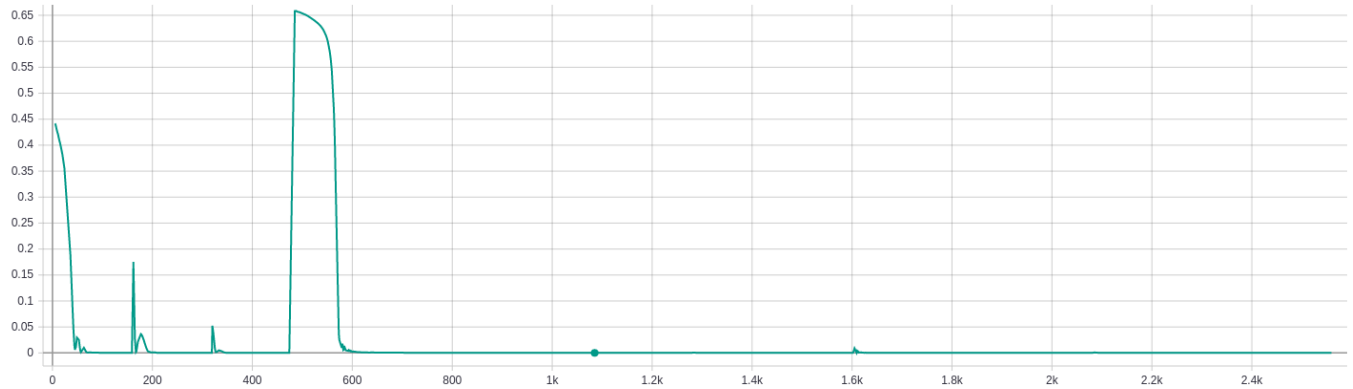


Figure 6: The loss change during the training of the model of a harmonic oscillator

5.2 Harmonic oscillator

A harmonic oscillator is a system with 1 DOF whose hamiltonian is [7, p. 157]

$$\mathcal{H}(t,q,p) := \frac{p^2}{2m} + \frac{1}{2}kq^2,$$

where m is the mass of the particle, and k is a parameter. To be simple, we take $m := 1/2$ and $k := 2$.

The dataset is generated using a ODE solver. The dataset has totally 16 datas. Every data consists of the size of considered time interval (lies in the range $(0, 2)$), a series of 200 different initial conditions (lies in the range $(-0.71, 0.71)^2$) to be learned as a batch, and the corresponding state of the system at the end of the time interval.

Every data is learned for 160 epoches. When a data is finished, the neural network moves on to the next data. Thus there are totally 2560 steps. The loss change in shown in Figure 6.

The loss may suddenly jump to a high value when the neural network switches to a different data because it may have converged to a wrong minimal of loss when it learns with the last data.

Taking the initial condition $\mathbf{x} = (0.1, 0)$ to test the trained model and compare it with the ground truth, the result can be seen in Figure 7. The figure shows the trajectory of the motion in space of $\mathbf{x}$, which is the phase path [7, p. 146][1, p. 68]. As can be seen, it can give the correct result: the phase path is a circle [1, p. 17].

5.3 Kepler's problem

The Kepler's problem is the motion of a particle in Coulomb field. Its hamiltonian is

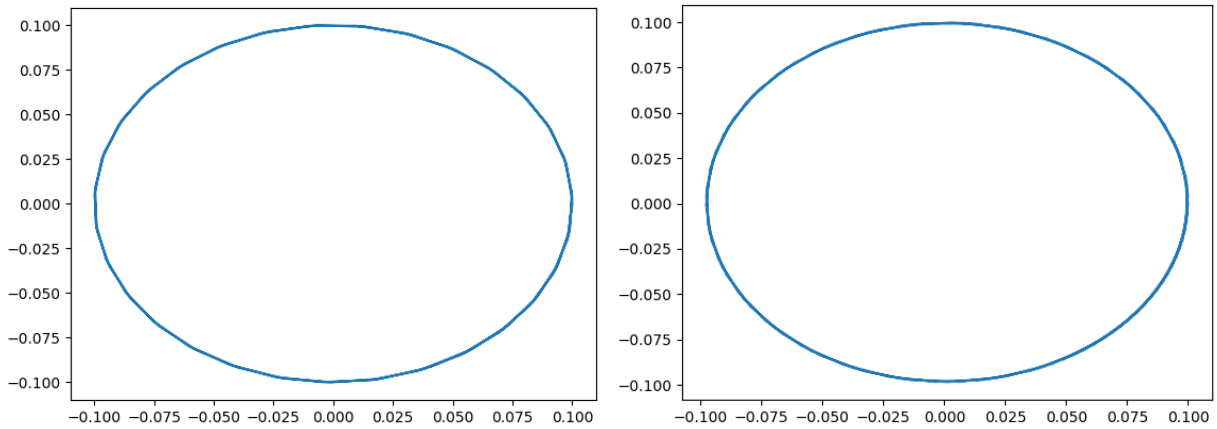
$$\mathcal{H}(t,q,p) := \frac{p_0^2 + p_1^2}{2m} - \frac{\alpha}{\sqrt{q_0^2 + q_1^2}}.$$

To be simple, we take $m := \frac{1}{2}$ and $\alpha := 1$.

References

[1] V. I. Arnol d, K. Vogtmann, and A. Weinstein. *Mathematical methods of classical mechanics*. Springer, 2nd edition, 1989.

[2] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.



(a) The result given by the trained model

(b) The ground truth

Figure 7: The comparation of the result given by the trained model and the ground truth

[3] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks, 2020.

[4] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks, 2019.

[5] L. N. Hand and J. D. Finch. *Analytical mechanics*. Cambridge University Press, 2008.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[7] L. D. Landau, L. E. Mikhaĭlovich, J. B. Sykes, and J. S. Bell. *Mechanics*. Butterworth-Heinemann, 3rd edition, 1976.

[8] R. D. Reed and R. J. Marks. *Neural smithing: supervised learning in feedforward artificial neural networks*. The MIT Press, 1999.