A Data-Driven Chaos Indicator for Nonlinear Dynamics

Abstract

Ⅰ. Introduction and Related work

Li et al [1] proposed data-driven chaos indicator to characterize the degree of nonlinearity for dynamical systems. The indicator was represented by a polynomial regression model trained from simulation data. Since a system with higher degree of nonlinearity is harder to be precisely predicted, this model can successfully indicate the chaotic degree of nonlinearity according to prediction accuracy. However, linear regression may not be optimal solution to this problem. We applied ensemble learning algorithms to this problem, like random forest and GBDT, which have better performance.

Ⅱ. Our model for chaos indicator

On the whole, we do short-term tracking to simulation data. Then we expect machine learning methods to learn certain pattern of relation between initial coordinates and transformed coordinates after 20 turns. We train the model on training data without supervision, and use the chaos indicator to predict testing data's transformed coordinates based on initial coordinates and the pattern learnt. The deviation between the predicted coordinates and short-term tracking ground truth serves as prediction accuracy, which indicates the degree of nonlinearity of the system.

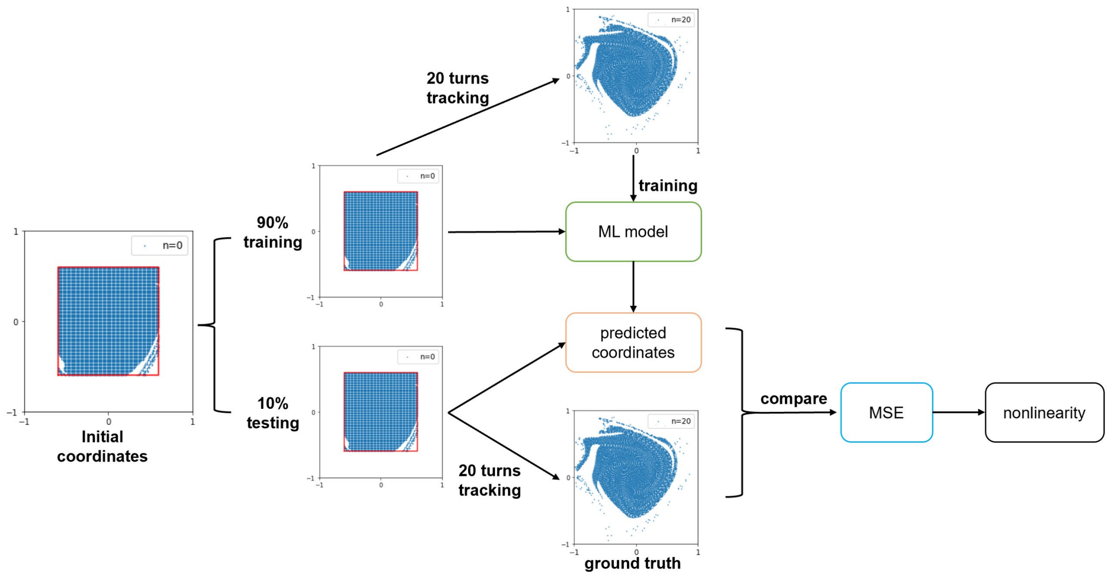

Figure 1. The data-driven chaos indicator, when v=0.205

The initial coordinates include x and p of particles. They are limited within [-0.6, 0.6], and we ignore the particles whose transformed coordinates exceed 1 in absolute value. In the [-0.6, 0.6] area, particles are more likely to be stable, thus it's easier for our machine learning model to learn the relation between initial coordinates and transformed coordinates after 20 turns. If we change area [-0.6, 0.6] to [-1, 1], particles are inclined to be more chaotic, which make it more challenging to learn the certain pattern.

The data-driven chaos indicator for nonlinear dynamics is based on Hénon map. The quadratic Hénon map equation is

$$\begin{pmatrix} x \\ p \end{pmatrix}_1 = \begin{pmatrix} \cos 2\pi\nu & \sin 2\pi\nu \\ -\sin 2\pi\nu & \cos 2\pi\nu \end{pmatrix} \begin{pmatrix} x \\ p - x^2 \end{pmatrix}_0$$

where v is the linear tune. By changing the value of v continuously, we simulate systems with different degree of nonlinearity. Since the performance of machine learning models is correlated to the nonlinearity of the system, a more nonlinear system ought to be more difficult to be learnt with worse prediction accuracy. It is reasonable that model will show worse performance at v=1/3, 1/4, 1/5, 1/7 and so on. Due to the quadratic perturbation, a peak occurs corresponds to a strong resonance line.

In previous work, Li et al [1] first applied 7th polynomial regression algorithm to the chaos indicator, which is a combination of polynomial expansion and linear regression. It's known that machine leaning model can be categorized as linear models, nonlinear models and ensemble learning models. To compare the performance of ensemble models and traditional machine learning models, we first try Logistic regression, Support Vector Machine regression (SVR), Bayesian Ridge regression and 7th polynomial regression again.

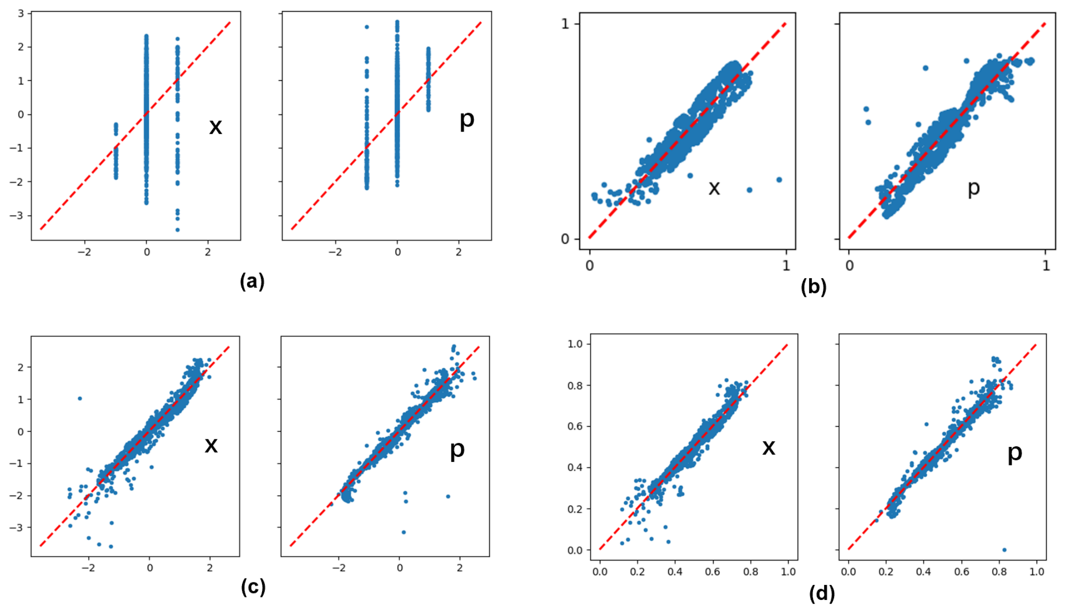

Figure 2. Prediction performance of four different machine learning algorithms in the x and p plane at a given tune v = 0:205. Each dot represents one initial condition in the testing cluster. The red dashed line is the desired expectation.

(a) Logistic regression. (b) SVR. (c) Bayesian Ridge regression. (d) 7th polynomial regression.

In addition to these four machine learning algorithms, we also test the performance of ensemble learning algorithms including random forest and Gradient boosting decision trees (GBDT).

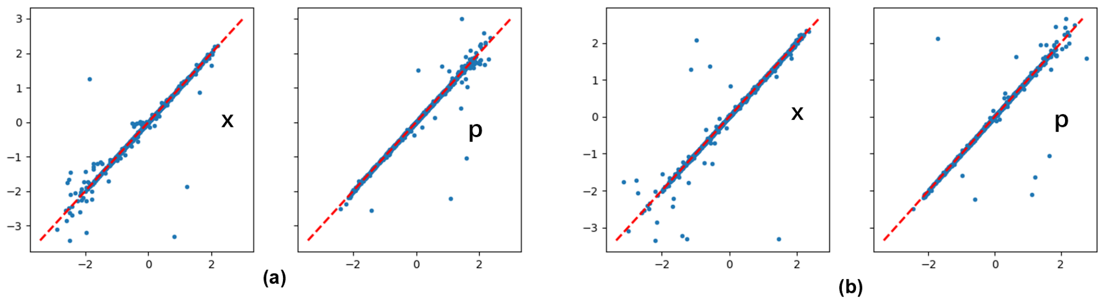

Figure 3. Prediction performance of two ensemble learning algorithms in the x and p plane at a given tune v = 0:205. (a) Random forest. (b) GBDT.

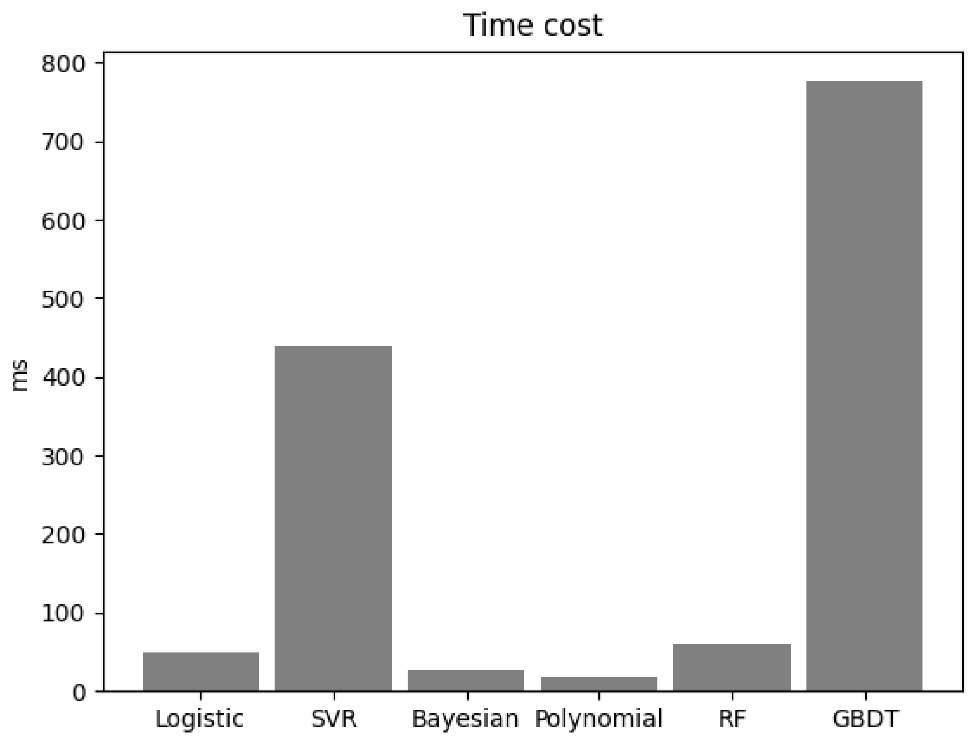

Figure 4. Time cost of six algorithms. Measure at n_estimators = 3 in random forest, n_estimators = 50 in GBDT. Larger n_estimator will improve prediction performance, but will also cost more time to do training and prediction.

From the aspect of prediction performance, the closer dots are to the red dashed line, the better performance the model has. That means the model is successfully learn the pattern of Hénon mapping and the relation between initial coordinates and transformed coordinates. Both random forest and GBDT show better performance compared to other four algorithms. Polynomial regression takes the least time during the training and prediction process, but it's unable to predict each particle in a very accurate way. GBDT has good prediction results but is expensive in time cost. By contrast, random forest is an excellent algorithm in this problem, both in prediction accuracy and time cost.

A tune scan means to characterize the nonlinear system behavior at different linear tunes. By doing long-term tracking to the testing data, we can know how many unstable particles are lost after 1000 turns. We assume that the percentage of lost particles represents the nonlinearity of the system, and the model performance is measured with the mean squared errors (MSE) between predicted 20-turns coordinates and tracking ground truth, which also serves as the data-driven chaos indicator.

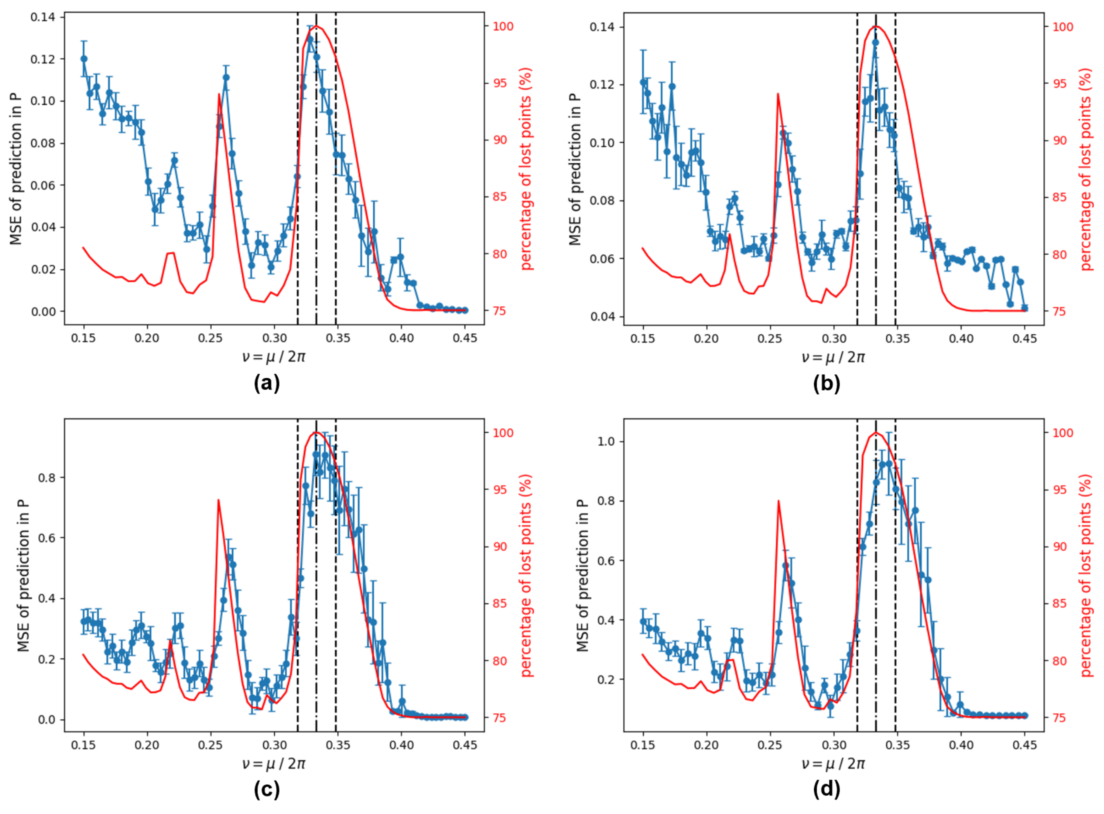

Figure 5. Prediction performance of four different models (blue line with error bars) vs. loss rate (solid red) of Hénon map at different tunes. The black dot-dash line are the 1/3 resonance line and its asymmetric stopband widths at each side with ±0:015. (a) 7th polynomial regression. (b) SVR. (c) Random forest. (d) GBDT.

Each of these four models indicates the nonlinearity of the system well. Each peak on the blue line corresponds to a higher order resonance line. The data-driven chaos indicator captures many useful information without doing long-term tracking. Given the initial coordinates and 20-turns tracking coordinates, the machine learning model can learn the pattern without supervision. Among them, random forest and GBDT model are better chaos indicators compared to polynomial regression and SVR model, especially when v<1/4 and v>1/3. And random forest model almost fits the red line (ground truth) perfectly, and the amplitude of the fluctuations are also predicted more accurately.

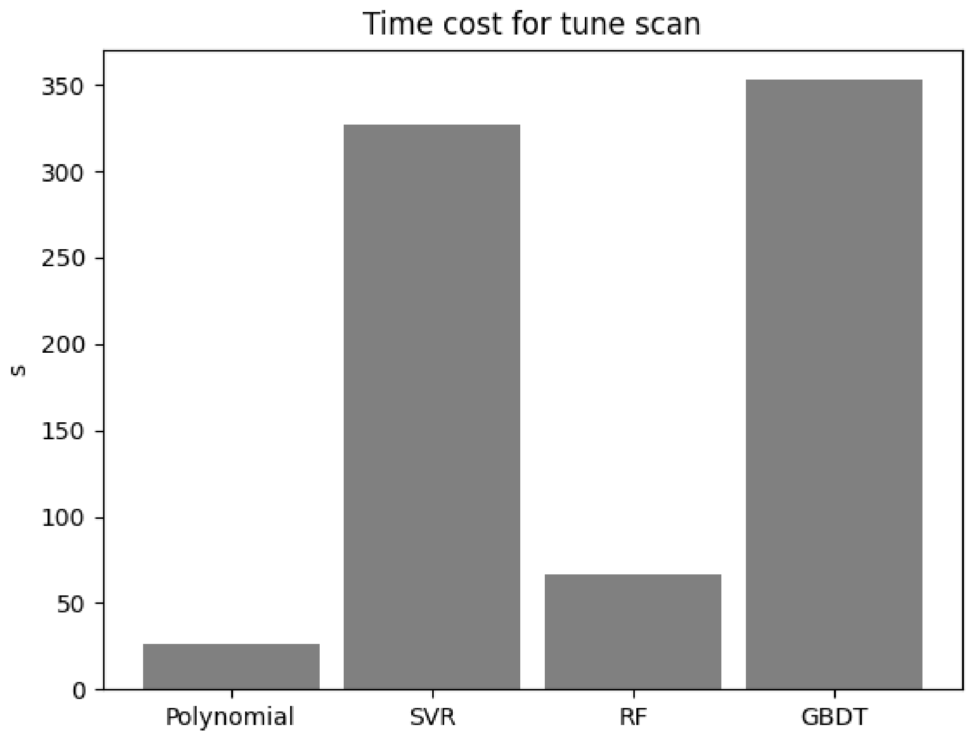

Figure 6. Time cost for tune scan of four algorithms. Measure at n_estimators = 5 in random forest, n_estimators = 50 in GBDT. Larger n_estimator will improve prediction performance, but will also cost more time to do training and prediction. All the experiments are running on a personal laptop computer.

Polynomial regression and random forest method are much computational faster than SVR and GBDT. From perspectives of prediction performance and time cost, random forest is definitely the best method to this problem. It reflects the nonlinearity of the system well, and at the same time, random forest doesn't take much time for model training and prediction.

Ⅲ. Our model for particles classifier

By doing long-term tracking to the simulation data, we can label them. Unstable particles are those whose coordinates tend to be very large, and stable particles are others. Then we notice that MSE of unstable particles is larger than MSE of stable particles. It's quite reasonable since an unstable particle has more uncertainty than a stable particle, which means it's more difficult to predict them. So prediction accuracy of chaos indicator declines when applied to prediction of unstable particles. And based on this fact, based on the idea of the chaos indicator, we wonder it is possible to distinguish every single stable and unstable particle according to the deviation between predicted coordinates and tracking ground truth?

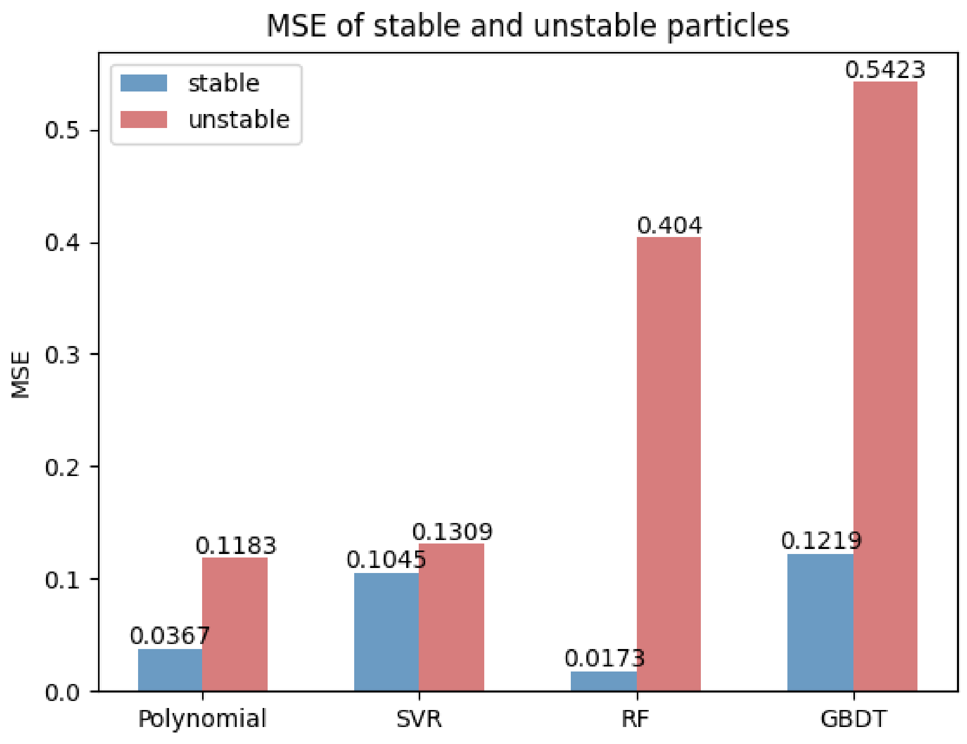

Figure 7. MSE of four different algorithms. Red bar represents MSE of unstable particles, blue bar represents MSE of stable particles.

Noticed that random forest has the least MSE of stable particles, but large MSE of unstable ones, which is one of the reasons why it shows good prediction performance. For each single particle, we want to build a particle classifier to predict this particle is stable or unstable (will be lost in long-term tracking) based on the predicted 20-turns coordinates and tracking ground truth. Here we visualize the prediction results of stable and unstable particles.

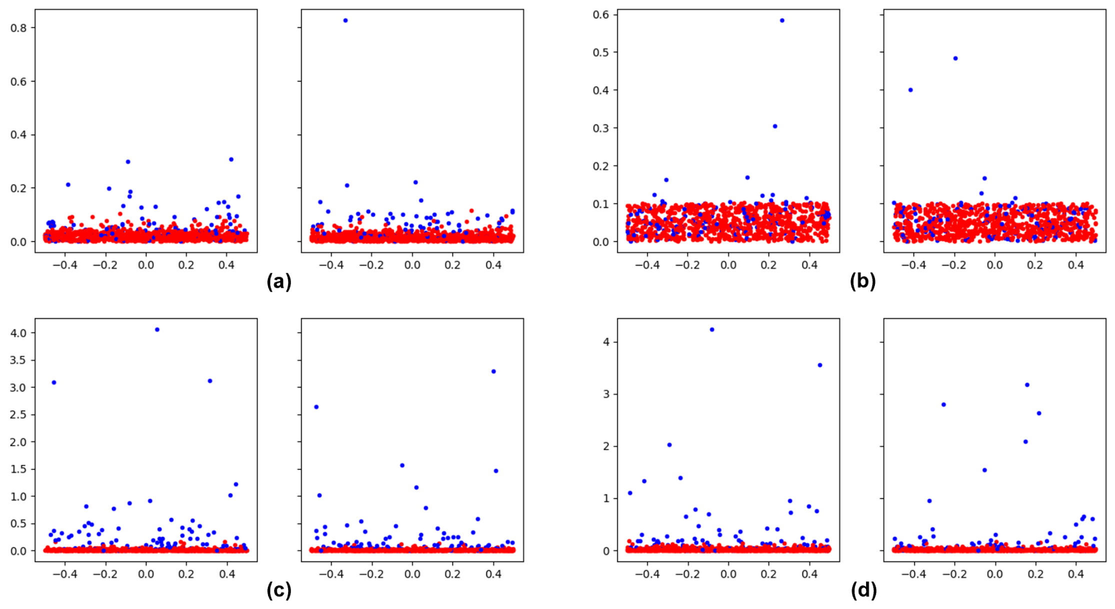

Figure 8. Visualization of four different algorithms. Red points represent stable particles, blue points represent unstable particles. The abscissa is randomly generated from [-0.5, 0.5] to separate points and make them clearer. The ordinate is the deviation between predicted coordinates and tracking coordinates. (a) 7th polynomial regression. (b) SVR. (c) Random forest. (d) GBDT.

In random forest and GBDT, unstable particles are well separated from stable ones. We take Δx is the deviation between predicted coordinates x and tracking coordinates x, similarly Δp.

$$C = \Delta x^2 + \Delta p^2$$

Since we normalize the data during data preprocessing. Coordinates x and p should be regarded as equal. Then C serves as a measurement of energy of each particle. We use C as a criterion to determine a single particle is stable or not.

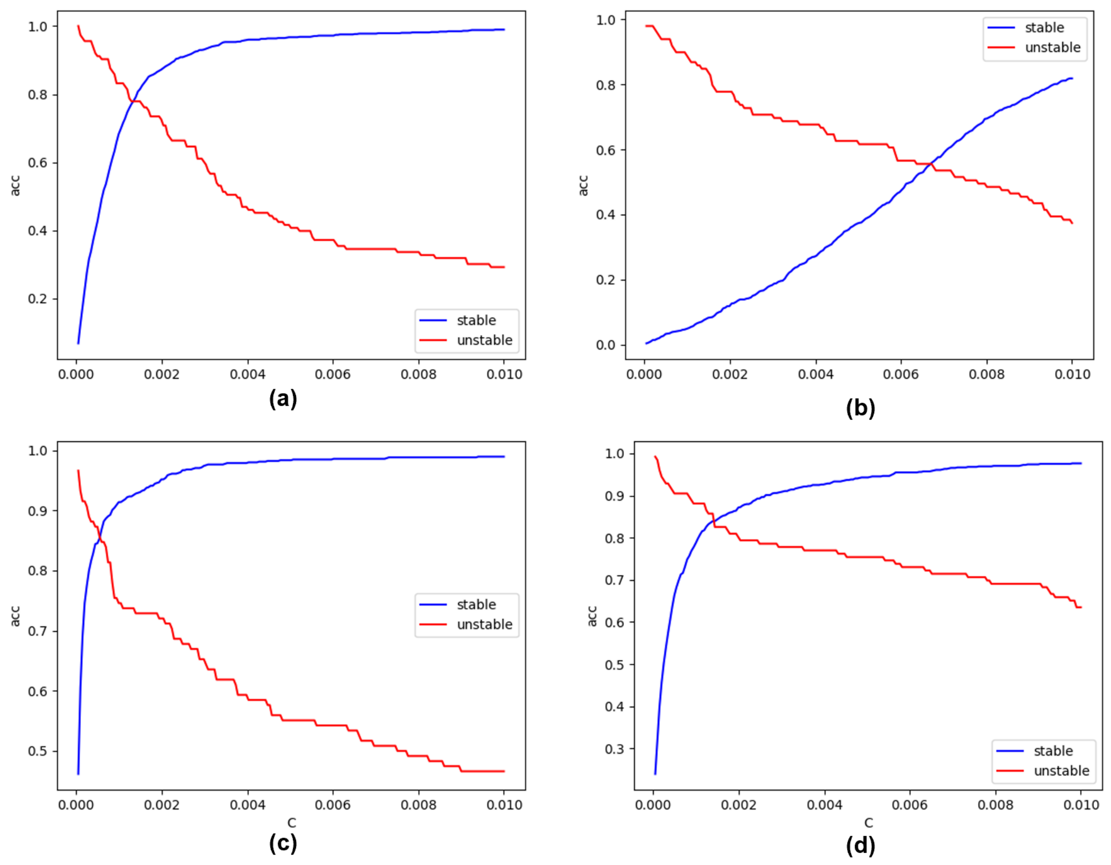

Figure 9. Classification accuracy of four algorithms. Blue line represents the recall radio of all stable particles, and red line represents the recall radio of unstable particles. The crossover point means classification accuracy when recall radio of stable and unstable particles equals. (a) 7th polynomial regression. (b) SVR. (c) Random forest. (d) GBDT.

We evaluate classification performance based on the fact that we have already label the data. Since it is so, we wonder what will happen if we train the model only using data of stable particles. In this way, classifier is able to learn stable pattern instead of unstable pattern. Thus model trained with stable particles can distinguish stable particles better. However, label the data in real world is unpractical, because that is computational expensive. We wonder how to improve the model performance using unsupervised learning methods. One idea is that improve the model performance by choosing training data selectively and wisely. As proportion of stable particles in training data is related to the classification performance, we introduce 2-dimension Gaussian distribution in choosing training data. It's widely acknowledged that the points close to fix point, i.e. (0, 0), are more likely to be stable. According to Gaussian distribution, the data selected for training has higher chance to be stable particles. As a result, the modified model's performance lies between the original one and the one trained by stable data.

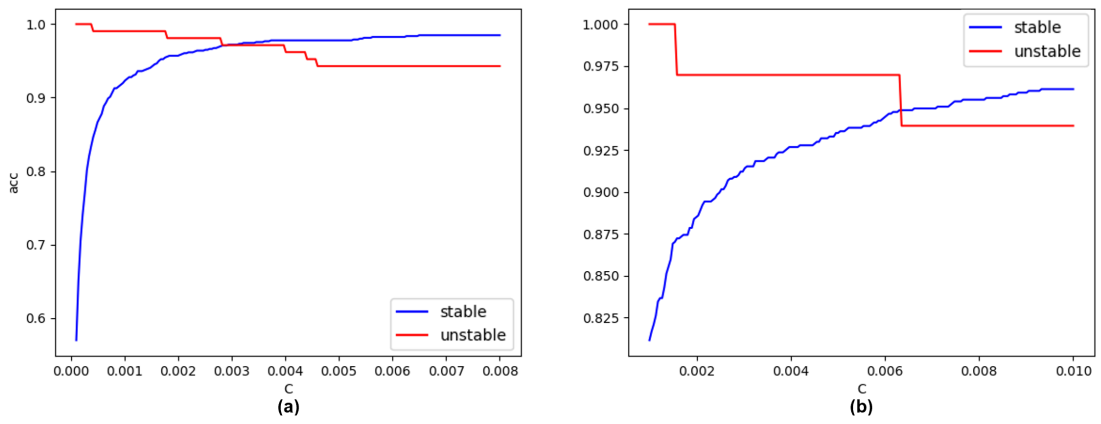

Figure 10. Random forest classification accuracy using two different set of data. (a) only stable particles. (b) data selected by Gaussian distribution.

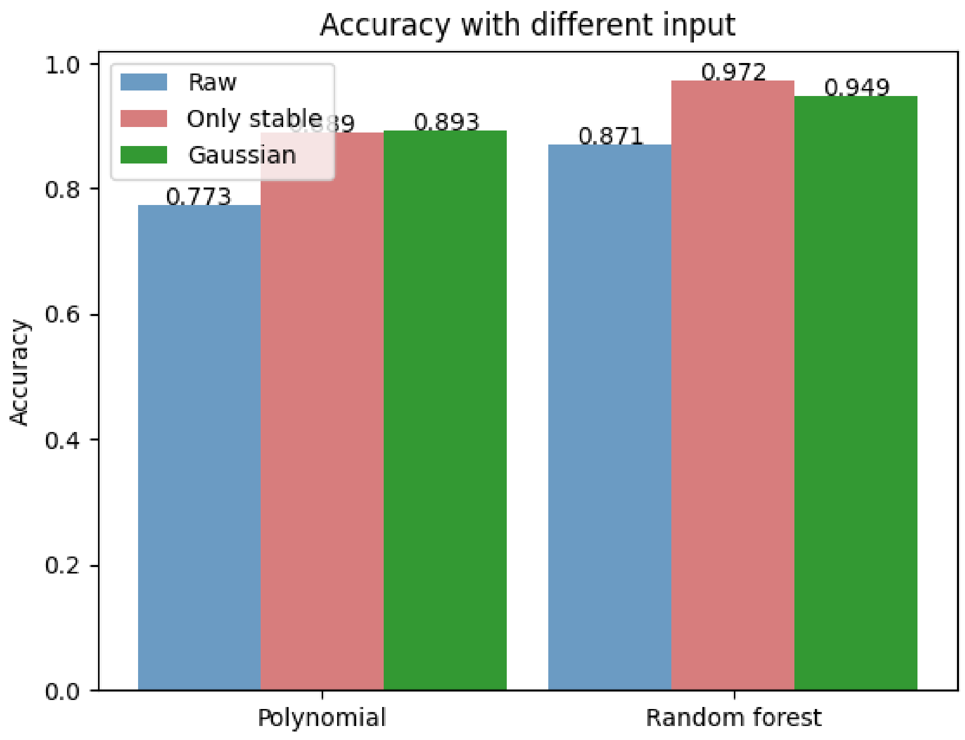

Figure 11. Classification accuracy with different training data. Blue bars represent that input is raw data without preprocess. Red bars represent that input is all stable particles selected by long-term tracking label. Green bars represent that input is selected by Gaussian distribution with more stable particles.

Ⅳ. Discussion and Conclusion

After introducing random forest and Gaussian distribution into our algorithm. The machine learning model can serve as a chaos indicator or a particle classifier, which obtains a classification accuracy of 94.9% without doing long-term tracking.

Reference

[1] Y Li, J Wan, Allen Liu, Yi Jiao, and Robert Rainer, "Data-driven chaos indicator for nonlinear dynamics and its applications on storage ring lattice design"

[2] 万金宇，孙正，张相，等. "机器学习在大型粒子加速器中的应用回顾与展望", doi: 10.11884/HPLPBxxxx