Controlling step-size with machine learning in solving ODE

Mikhail Kuimov
Optimization Class Project. MIPT

Introduction

The importance of numerical methods of solving complex differential equations shouldn't be underestimated. They help us to approximate the solution of equations, that can't be solved in closed form. That's why every new idea or method that makes the process more efficient deserves some time to be wasted. Let us look on Mahesh Narayanamurthi's method [1] of controlling step size with my modest improvements. The linear model is trained to predict, will the next step size be accepted or not in the method of automatic step size control. It is made to reduce the quantity of rejected step. The project is divided into two parts: the first part includes experiments for method with separate collecting of the data set and separate testing of the trained model, while the second part is mostly about the original method with training alongside the integration process.

Part 1

First of all, it was decided to test the idea on simple system of differential equations:

$$\begin{cases} y_1' = -\sin y_3 \\ y_2' = -100\sin(100 y_2) \\ y_3' = 1 \end{cases}$$

The data set was collected during the performance of automatic algorithm with Runge-Kutta-Fehlberg integration method (can be found here [2]). Then, the *LinearRegression* estimator was fitted and used to predict, whether the next step would be accepted or not. If not, the step size was shrunk, while not reaching the acceptable value.

Results

In the result, zero number of rejected steps was reached. However, the total number of steps increased, which is understandable, because the model is only used to shrink the step size.

	Number of accepted steps	Number of rejected steps	Total number of steps
Auto	5201	365	5566
My Auto	47804	6	47811
My ML	53833	0	53833
My ML dif.dat.	57180	2	57182
ML	8447	0	8447
ML dif.dat.	8447	0	8447
Auto dif. eq.	6957	1073	8030
ML dif. eq.	8498	0	8498

Part 2

Allen-Cahn reaction diffusion equation:

$$u_t = \alpha \nabla^2 u + \gamma(u - u^3), \quad (x,y) \in [0,1] \times [0,1], \quad t \in [0, 0.3]$$

The first term on the right hand-side is the diffusion term and the second, the reaction term. The problem can be made stiff by increasing the resolution of the spatial grid. The spatial grid resolution was set to 128×128 making the total number of states equal 16384. The initial and boundary conditions across all the experiments are kept the same.

Improvement 1

Let's allow the integration method to also grow step sizes if the model predicts that the future step will be accepted. Save accepted step size values, which were predicted correct in the validation window and use them for greater good.

```
1. $h_{new} := h + h/15$
2. $h_{new} := max(h_{new}, max\_validation\_window\_element)$
```

Coefficient $\frac{1}{15}$ is chosen experimentally to have the best balance between decrease in total steps and increase in rejected steps.

Improvement 2

Let's weight the validation window elements with step size values for model to understand, when it does something wrong or wright and achieve higher accuracy. Then the condition for prediction looks like:

```
if $accuracy\_score(true\_h, pred\_h, sample\_weight = weight) > valthreshold:$
        1. predict
```

So, now we push even more information in validation window.

Bad tests

- It was tried to implement growth of the step size in a loop, which was followed by worse results.
- The different mean with correctly accepted and rejected h was tried, but there were no positive results.
- It was tried to chose new h, depending on the information from previous wright predictions — also nothing good.

Results

- If we let the step size to be increased, in case the model predicts, it will be accepted, we achieve almost the same or even fewer total iterations made. The quantity of rejected steps grows a bit, but it is still lower than in automatic method.
- If we weight elements of validation window with current step size values, we achieve significant decrease in rejected steps in case of SGDClassifier (almost twice). On the other hand, the total number increases.
- If we test the method from previous section with separate collecting of training data, we get wonderful results in decreasing rejected steps quantity: zero rejected steps.

	Number of observed variables	Number of training samples	Number of accepted steps	Number of rejected steps	Total number of steps
Auto	-	-	10453	987	11440
SGD(original)	25	5	11365	751	12116
SGD(mean)	25	5	10557	855	11412
Perceptron(mean)	25	5	10282	984	11266
SGD(weighted)	25	5	18470	492	18962
Perceptron(weighted)	25	5	10882	957	11839
Separate method	-	-	11331	0	11331

Conclusion

New error control mechanism, where a machine learning model is trained and used to predict the acceptance of next step size, was examined and tested on different differential equations. The quantity of rejected steps was minimized. Morover, new error control mechanism, where a machine learning model is trained alongside the time integration process, was examined and tested on different differential equations and different linear models. Also, several adjustments were developed and showed to be not useless. The algorithm has started working faster in one case and more accurate and efficient in another. All code is here: [3]

References

[1] Mahesh Narayanamurthi. Applying machine learning to scientific computing problems. pages 1–7, 2018.

[2] John H. Mathews and Kurtis D. Fink. *Numerical Methods Using MATLAB*. Simon & Schuster, Inc., New York, NY, USA, 3rd edition, 1998.

[3] Code. https://github.com/kmisterios/opt_project.git.