Improvement on Normal ETA Prediction

Author: Yucan Yu

November 19, 2022

Contents

1 Introduction **2**

2 Data Description **2**

3 Math Model Description **4**

3.1 Simplest Method 4

3.2 Linear regression 4

3.3 Polynomial regression 4

3.4 SVR (Partial Data Used) 5

3.5 Linear regression 2 (Partial Data Used) 5

4 Model Prediction **6**

5 Discussion **8**

1 Introduction

As we know, there are a lot of things that have their own progress. We have progress bars and usually are able to see a small number near it that estimate remaining time based on current work progress speed. That is also names ETA (estimated time of arrival).

By simply measuring the “speed” of current work, we could get a quick feedback on how long it will need. But usually this simple method is unsteady and can have a large error. In fact, they only considered two points, that is "the beginning" and "the current one", which means if one of those two points have a big error, the total error of this ETA method will be big.

If this prediction can be improved using a regression model(least squares), a more stable prediction may be obtained. If the algorithm were smarter and could choose an appropriate model, it could theoretically predict more accurately. That is what this project going to explore.

2 Data Description

The data in this project are collected in a variety of situations through Python. In order to get the real situation more accurately, I tried to record video directly several times in several different situations, and then used tracking software to track the video pixels to get the progress bar changes. Finally all the data are uniformed and saved to csv files to be analyzed later.

Mainly the data are situations for download files and move files. I also tried some data generated with a non-uniform random function (exponential distribution).

For details, please refer to the following figure:

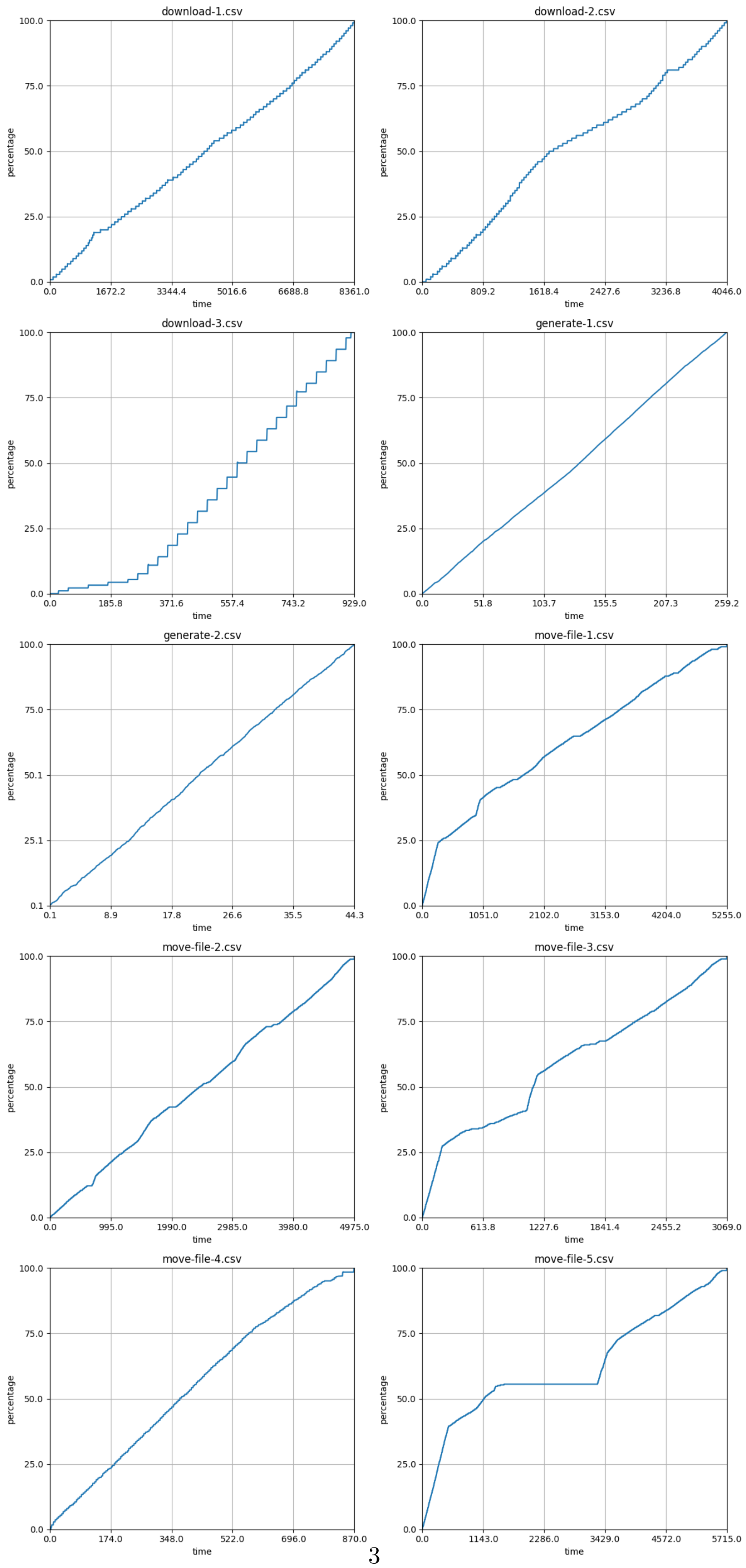


Figure 1: Input Data Set

3 Math Model Description

3.1 Simplest Method

Before going into my personal approach, I think it's worth mentioning how quickly this ETA should be calculated in general.

The simplest approach to this is by calculate the current speed. Then we have

$$ETA = \frac{(RemainingPercentage)}{(CurrentSpeed)}$$

That is all.

3.2 Linear regression

By minimizing the sum of squared errors, we can get a linear regression model from the data. Obviously, linear regression could give us a better prediction than simple division method that only considers 2 points, because linear regression cares all the data and it will do a best fit. From its nature, we could know that it is not sensitive to large deviations of a single data, so a higher stability can be expected too. However, there may be two problems: can be a little slow (as it uses all data point) and not sensitive to changes.

3.3 Polynomial regression

Since the linear regression model is linear, and our actual situation may not be mostly linear, I considered trying polynomial fitting. But the question becomes how do we know that the degree polynomial is a suitable choice. If the degree is too high, overfitting is likely to occur. For simplicity, I use the second degree. I wasn't too expecting how good the polynomial fit would be, just to try it out.

3.4 SVR (Partial Data Used)

The support vector regression of the rbf kernel may surprise us. Because it gives good predictions for less regular curves. But I also take into account that this must be a slow calculation, but it's worth it if it leads to a decent model. If we adopt this method, we also have to think about how to adjust its parameters to optimize it.

In detail, I only used last 200 data samples. The speed is still MUCH slower than every other methods.

3.5 Linear regression 2 (Partial Data Used)

After countless attempts, I found that these seemingly irregular curves may still be the best predicted by linear models. In order to improve the excessive computational consumption and insensitivity to changes caused by training the model with all the data, I thought of using only the last part of the data for linear fitting. Of course, in many cases, the fitted line will not pass the origin or deviate very far, but this will not be a problem.

In detail, I only used last 1000 data samples. The speed is nearly same with using whole data samples.

4 Model Prediction

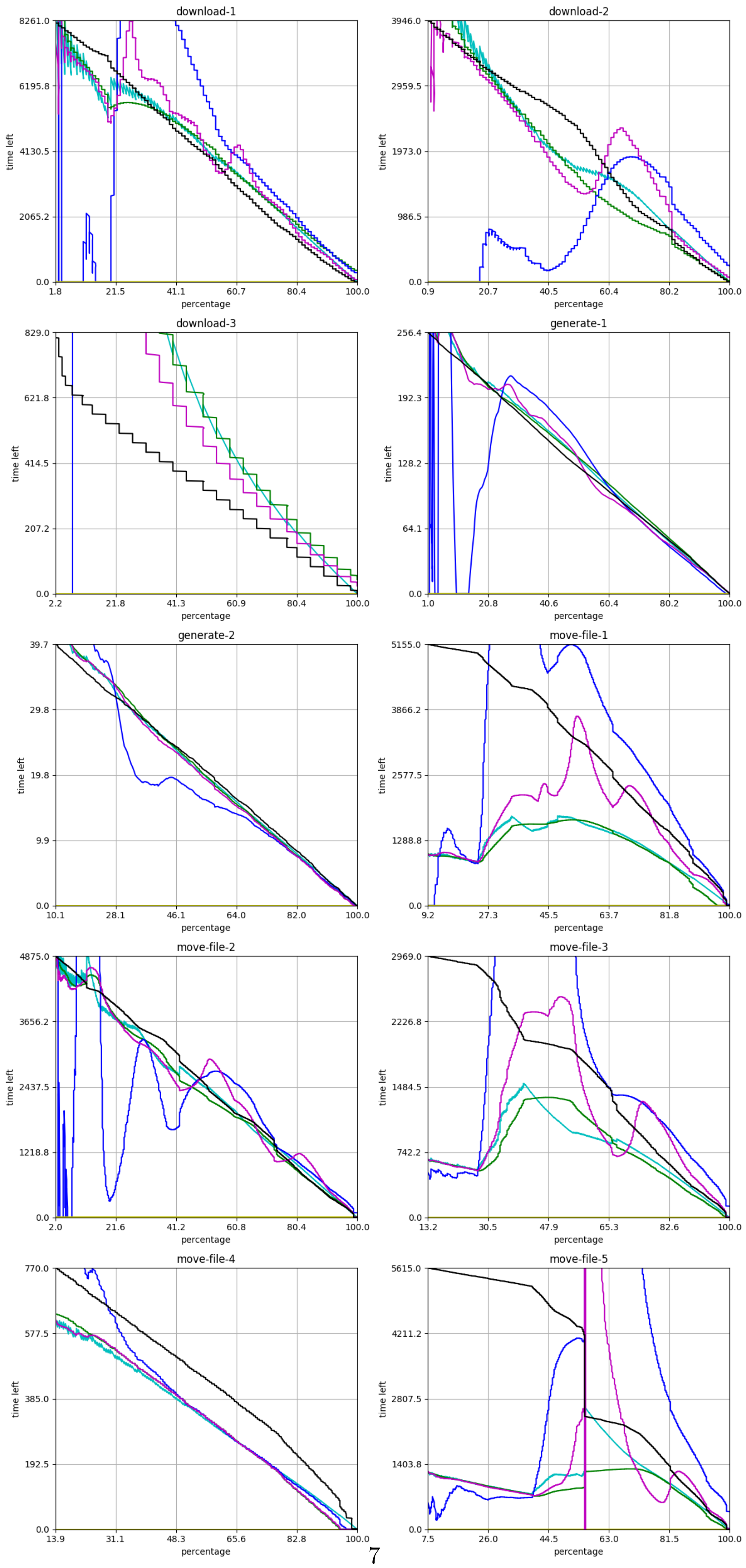
Since the focus of this project is different from general prediction models(we need something can do near real-time), it is more accurate to obtain a model in real time and make predictions, or it should be considered as an algorithm. Therefore, it is necessary to consider how to evaluate the quality of different prediction algorithms. I mainly considered three aspects: stability, prediction accuracy, and calculation time. The degree of stability can be measured by the number of changes in the data (the data here plus the consumption time, that is, the final time consumption of the prediction, which will be a constant). For the prediction accuracy I used the mean sum of squared errors, which to some extent tells us how accurate the data is from the algorithm. Computational time is also an important aspect (if we are going to use this prediction method), and by summing the computation time in the code, we can accurately compare the efficiency of the algorithms.

All methods are starting calculation from 100th samples as it will have a large disturbance no matter which methods used starting from the beginning.

We could see that blue one(SVR) gives a lot of floating, I guess that is not a good choice. The megenta one give some disturbance too but gives a better "attraction" to the real time with a fast speed. Cyan one is the simplest one, however, it gives a not-too-bad prediction. (That make the common usage more sense)

In detail, The color in following graphs means:

Black	Cyan	Green	Blue	Yellow	Magenta
Real	Division	LR	Poly	SVR (Partial)	LR (Partial)



download-1
8261.0
6195.8
4130.5
2065.2
0.0
1.8
21.5
41.1
60.7
80.4
100.0
download-2
3946.0
2959.5
1973.0
986.5
0.0
0.9
20.7
40.5
60.4
80.2
100.0
download-3
829.0
621.8
414.5
207.2
0.0
2.2
21.8
41.3
60.9
80.4
100.0
generate-1
256.4
192.3
128.2
64.1
0.0
1.0
20.8
40.6
60.4
80.2
100.0
generate-2
39.7
29.8
19.8
9.9
0.0
10.1
28.1
46.1
64.0
82.0
100.0
move-file-1
5155.0
3866.2
2577.5
1288.8
0.0
9.2
27.3
45.5
63.7
81.8
100.0
move-file-2
4875.0
3656.2
2437.5
1218.8
0.0
2.0
21.6
41.2
60.8
80.4
100.0
move-file-3
2969.0
2226.8
1484.5
742.2
0.0
13.2
30.5
47.9
65.3
82.6
100.0
move-file-4
770.0
577.5
385.0
192.5
0.0
13.9
31.1
48.3
65.6
82.8
100.0
move-file-5
5615.0
4211.2
2807.5
1403.8
0.0
7.5
26.0
44.5
63.0
81.5
100.0
time left
percentage

Figure 2:

5 Discussion

In this project, I tried many methods and found that, although it looks like a non-linear model, however, linear models are indeed relatively more suitable for such progress forecasting in general.

The whole process of the project was kind of tortuous. At first I thought that using linear regression would be a very good choice, but after a lot of experimentation I found that simply using linear regression for all would make the computation time much longer than simple division (of course, this is relative to the consumption of other machine learning models) is much lower), so that I want to find a non-linear model, I tried a lot, I found that there is not such a simple method, maybe by training a neural network can get a better model than these I tried, But I don't have that much data and do not sure if that is a good way to go. Finally I turned my attention back to simple linear regression. I guess it doesn't make sense to choose linear regression, because, by nature, general progress will appear linear. But closer to reality is a discounted linearity. So I tried linear regression using only the last part of the data, on the one hand this speeds up the calculation and on the other hand makes the prediction more accurate.

References

Here are some websites that did similar research (but not real-time calculation) on this topic which could be useful:

https://stats.stackexchange.com/questions/114211/predicting-time-to-finish

https://arxiv.org/abs/2108.11482