What is the Time Complexity of Linear Regression?

Asked 3 years, 1 month ago Active 2 years, 4 months ago Viewed 11k times

6

3

I am working with linear regression and I would like to know the Time complexity in big-O notation. The cost function of linear regression without an optimisation algorithm (such as Gradient descent) needs to be computed over iterations of the weight combinations (as a brute force approach). This makes computation time dependent on the number of weights and obviously on the number of training data.

If n is the number of training data, W is the number of weights and each resolution of the weight space is set to m meaning that each weight will iterate through m number of possible values. Then the time complexity of this linear regression is

$$O(m^W n)$$

Is this correct?

machine-learning regression statistics linear-regression cost-function

Share Edit Follow Flag

asked Jul 20 '18 at 15:47
user134132523
79 1 3 10

ai.stackexchange.com/questions/5728/... – DuttaA Jul 20 '18 at 20:02

@DuttaA how is this relevant? – user134132523 Jul 22 '18 at 15:27

Sorry I read with an optimisation algo...I didn't know you were trying by matrix methods – DuttaA Jul 22 '18 at 15:52

1 Answer

Active Oldest Votes

2

It highly depends on the "solver" you are using.

Calling n the number of observations and p the number of weights, the overall complexity should be $n^2p + p^3$.

Indeed, when performing a linear regression you are doing matrices multiplication whose complexity is n^2p (when evaluating $X'X$) and inverting the resulting matrix. It is now a square matrix with p rows, the complexity for matrix inversion usually is p^3 (though it can be

Side notes

However, numerical simulations (using python's scikit library) seem to have a time complexity close to $n^{0.72}p^{1.3}$

This may be due to the fact that no implementation actually perform the full inversion (instead, the system can be solved using gradient descents) or due to the fact that there are other ways to calibrate the weights of a linear regression.

Source

An article from my blog : computational complexity of machine learning algorithms

Share Edit Follow Flag

answered Sep 3 '18 at 15:23

RUser4512
157 4

why is the exponent of n, squared. if I have 3 weights then the error function computed over n training sets have to be recomputed for all the possible combinations of those 3 weights right. same goes for higher number of weights. that is why my reasoning was that the number of weights should be an exponent. and also in your calculation where is the limit on the number of possible iterations per weight? I am talking about a purely brute force approach even though its not the optimal way to go about it. – user134132523 Sep 3 '18 at 18:01

"why is the exponent of n, squared. if I have 3 weights then the error function computed over n training sets have to be recomputed for all the possible combinations of those 3 weights right." No. This would be true if you were doing a naive approach, but based on the closed formula for $\hat{\beta}$, you do not have to try every combination of weights – RUser4512 Sep 4 '18 at 8:09

3 from : math.stackexchange.com/questions/84495/..., you may have inverted the exponent on n and p. The complexity should be np^2, this is also in line with practical time complexity. – lcrmorin Apr 15 '20 at 15:17

Indeed n and p are inverted, it confused me too. It would be nice if the author would fix the answer. – offlinehacker Jul 13 at 8:19