Applied Mathematical Sciences, Vol. 14, 2020, no. 13, 653 - 658
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ams.2020.914237

Why Cutting Trajectories Into Small Pieces Helps to Learn Dynamical Systems Better: A Seemingly Counterintuitive Empirical Result Explained

Olga Kosheleva and Vladik Kreinovich

University of Texas at El Paso
500 W. University, El Paso, TX 79968, USA

Abstract

In general, the more information we use in machine learning, the more accurate predictions we get. However, recently, it was observed that for prediction of the behavior of dynamical systems, the opposite effect happens: when we replace the original trajectories with shorter pieces – thus ignoring the information about the system's long-term behavior – the accuracy of machine learning predictions actually increases. In this paper, we provide an explanation for this seemingly counterintuitive result.

Mathematics Subject Classification: 68T07 37C99 62F12

Keywords: machine learning, dynamical systems, trajectories, short bursts

1 Cutting Trajectories Into Small Pieces Helps to Learn Dynamical Systems Better: A Seemingly Counterintuitive Empirical Result

General idea. In general, the more information we use for learning, the better the results of machine learning; see, e.g., [2, 3].

In some practical situations, we have a large amount of information, way exceeding the computer's ability to handle it. In such cases, we have to cut this information into smaller handleable pieces.

This is, e.g., how we determine the structure of a DNA: we cut it into smaller pieces, determine the structure of each of these pieces, and then try to reconstruct the structure of the whole DNA from the structures of these individual pieces; see, e.g., [1, 4].

However, in situations when it *is* possible to process the whole information, usually, better learning results are obtained when we consider the whole information.

A recent counterexample to the general idea. One of the things that we want to learn is the dynamical equations that describe how a given system evolves with time. We know the trajectories of this system, and we want to extract the corresponding dynamical equations from these trajectories.

This is what Newton did when he analyzed trajectories of celestial bodies, this is what physicists do when they try to find the corresponding dynamical equations.

In precise terms, what we have is trajectories

$$x_k = (x_k(1), x_k(2), \ldots, x_k(t), \ldots, x_k(T_k)), k = 1, 2, \ldots \tag{1}$$

that describe the state of the system $x_k(t)$ at different moments of time. In the ideal world, to train the corresponding machine learning algorithm, we should use, for each k and t, the tuple of the previous states

$$(x_k(1), \ldots, x_k(t-1)) \tag{2}$$

as the input, and the state $x_k(t)$ as the output.

However, most real-life machine learning procedure require that all the training inputs be of the same size. With this in mind, we select some large value t_0, and for each desired output $x_k(t)$, use, as input, the tuple

$$(x_k(t-t_0), \ldots, x_k(t-1)). \tag{3}$$

This *is* possible, and we do get some reasonable predictions, as shown in [5, 7]. However, surprisingly, as the same papers show, much better learning was achieved when the authors cut the trajectory into smaller pieces

$$(x_k(t), \ldots, x_k(t+h)), \tag{4}$$

for some small $h \ll t_0$ and all possible values t, and trained the network only on these pieces (the authors call them *bursts*). In other words, to train the machine learning algorithm, we use, as inputs, tuples

$$(x_k(t), \ldots, x_k(t+h-1)) \tag{5}$$

corresponding to different values k and t, and for each such tuple, the desired output is the corresponding value $x_k(t+h)$.

Why is this improvement in accuracy counterintuitive? When we divide each trajectory into small bursts, we delete some information: namely, we only keep information about the short-term dynamics, but we ignore information about the long-term dynamics. According to the above general idea, this ignoring an important part of information should make the results of machine learning worse – but actually, the results of learning based on cut inputs are much better.

How can we explain this seemingly counterintuitive behavior?

2 Our Explanation

What determines the accuracy of predictions: a brief reminder. The problem of determining parameters from observations is a typical problem in statistics; see, e.g., [6].

In particular, it is well known that the more observations we have, the more accurately we can determine the parameters – and thus, the more accurate is the result of prediction that use these parameters. The usual rule of thumb is that the standard deviation of the prediction error decreases as $\dfrac{1}{\sqrt{n_{\text{obs}}}}$, where n_{obs} is the number of observations.

It is also well known that the more parameters we need to determine, the smaller the accuracy with which we can determine each parameter – and thus, the less accurate is the result of predictions that use these parameters. In other words, as the number of parameters n_{par} increases, the prediction error also increases.

Crudely speaking, in general, if we use n_{obs} observations to determine n_{par} parameters, the resulting accuracy is equivalent to the case when these observations are divided into n_{par} groups of $\dfrac{n_{\text{obs}}}{n_{\text{par}}}$ observations each, so that each group determines its own parameter. As a result, for each parameter, the standard deviation of the estimation error is proportional to

$$\frac{1}{\sqrt{\dfrac{n_{\text{obs}}}{n_{\text{par}}}}} = \sqrt{\frac{n_{\text{par}}}{n_{\text{obs}}}}. \tag{6}$$

Let us consider the above situation from this viewpoint.

What happens when we cut the trajectory into bursts. To analyze how cutting changes accuracy, we need to analyze how cutting affects the number of observations and the number of parameters.

Let us start with the number of observations. For each original trajectory k of length T_k:

- we have $n_{\text{obs}}^{\text{long}} = T_k - t_0 + 1$ examples of type (3), corresponding to

$$t = t_0 + 1, t_0 + 2, \ldots, T_k, \text{ and}$$

- we have $n_{\text{obs}}^{\text{cut}} = T_k - h + 1$ examples of type (5), corresponding to

$$t = 1, 2, \ldots, T_k - h.$$

We see that after cutting, we get slightly more examples, but – in view of the $\dfrac{1}{\sqrt{n_{\text{obs}}}}$-rule, not enough to explain the drastic improvement in accuracy.

Let us now consider the number of parameters. In the simplest case, when we only consider linear dependencies, for long sequences, the general prediction formula has the form

$$x_k(t) = a_0 + a_1 \cdot x_k(t-1) + \ldots + a_{t_0} \cdot x_k(t - t_0), \tag{7}$$

with $n_{\text{par}}^{\text{long}} = t_0 + 1$ parameters a_i. For short sequences, we similarly have

$$x_k(t+h) = a_0 + a_1 \cdot x_k(t+h-1) + a_2 \cdot x_k(t_h - 2) + \ldots + a_h \cdot x_k(t), \tag{8}$$

with $n_{\text{par}}^{\text{cut}} = h + 1$ parameters. Here, $h \ll t_0$, so $n_{\text{par}}^{\text{cut}} = h + 1 \ll t_0 + 1 = n_{\text{par}}^{\text{long}}$. Thus, due to the formula (6), the use of bursts shall indeed drastically decrease the estimation error – even in the linear case – by a factor of

$$\sqrt{\frac{t_0}{h}}. \tag{9}$$

Of course, actual dependencies are non-linear – OK, some are linear, but in the linear case, we do not need machine learning to describe the corresponding dynamical system. To accurately describe a linear system, it is not sufficient to use its linear approximation, we need to also take into account higher order terms in the Taylor expansion. If we take into account quadratic terms, then for long sequences, the general prediction formula will take the form

$$x_k(t) = a_0 + \sum_{i=1}^{t_0} a_i \cdot x_k(t-i) + \sum_{i=1}^{t_0} \sum_{j=1}^{t_0} a_{ij} \cdot x_k(t-i) \cdot x_k(t-j), \tag{10}$$

with $n_{\text{obs}}^{\text{long}} \sim t_0^2$ parameters, while for short sequences, we have

$$x_k(t+h) = a_0 + \sum_{i=1}^{h} a_i \cdot x_k(t+h-i) + \sum_{i=1}^{h} \sum_{j=1}^{h} a_{ij} \cdot x_k(t+h-i) \cdot x_k(t+h-j), \tag{11}$$

with $n_{\rm par}^{\rm cut} \sim h^2$ parameters. So, here, due to formula (6), the use of short bursts decrease the prediction error by an even larger factor of

$$\frac{t_0}{h}. \tag{12}$$

If we take into account all the terms up to some power $p > 2$, then we will need $\sim t_0^p$ parameters for long sequences and $\sim h^p$ parameters for short bursts, so the prediction error decreases by an even larger factor of

$$\left(\frac{t_0}{h}\right)^{p/2}. \tag{13}$$

In all these cases, shorter bursts lead to much more accurate predictions – exactly as it is observed.

Comment. On the qualitative level, the above arguments can be reformulated as follows. We know that for a dynamical system, usually, its state in the next moment of time depends only on the state in the few previous moments of time. For example:

- For systems described by first order differential equations, the initial state, in general, uniquely determines the next state – and thus, the whole trajectory.
- For systems described by second order differential equations – like Newton's equations – we need to know states at two moments of time, this uniquely determines the trajectory, etc.

When we consider short bursts, we, in effect, take this knowledge (about short-term character of the system's dynamics) into account. However, when we apply a general machine learning algorithm to long sequence, this knowledge is lost. So, the machine learning algorithm, crudely speaking, unproductively, looks at all possible dynamics – not necessarily short-term ones, and this explains why the use of long sequences leads to a much worse performance of the machine learning algorithms.

Acknowledgments. This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 (CAHSI Includes), and HRD-1242122 (Cyber-ShARE Center of Excellence).

References

[1] A. D. Baxevanis, G. D. Bader, and D. S. Wishart (eds.), *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*, Wiley, Hoboken, New Jersey, 2020.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Leaning*, MIT Press, Cambridge, Massachusetts, 2016.

[4] N. C. Jones and P. A. Pevzner, *An Introduction to Bioinformatics Algorithms*, MIT Press, Cambridge, Massachusetts, 2004.

[5] T. Qin, K. Wu, and D. Xie, Data driven governing equations approximation using deep neural networks, *Journal of Computational Physics*, **395** (2019), 620–635. https://doi.org/10.1016/j.jcp.2019.06.042

[6] D. J. Sheskin, *Handbook of Parametric and Non-Parametric Statistical Procedures*, Chapman & Hall/CRC, London, UK, 2011. https://doi.org/10.4324/9780203489536

[7] K. Wu and D. Xiu, Numerical aspects for approximating governing equations using data, *Journal of Computational Physics*, **384** (2019), 200–221. https://doi.org/10.1016/j.jcp.2019.01.030

Received: August 2, 2020; Published: August 17, 2020