

Finding the Optimal Continuous Model for Discrete Data by Neural Network Interpolation of Fractional Iteration

Lars Kindermann¹, Achim Lewandowski² & Peter Protzel²

¹RIKEN Brain Science Institute, Lab for Mathematical Neuroscience
Wako-shi, Saitama 351-0106, Japan
kindermann@brain.riken.go.jp
www.mns.brain.riken.go.jp/~kinderma

²Dept. of Electrical Engineering and Information Technology
Chemnitz University of Technology, Germany
peter.protzel@e-technik.tu-chemnitz.de, lewandowski@alewand.de
www.infotech.tu-chemnitz.de/~proaut

Abstract. Given the complete knowledge of the state variables of a dynamical system at fixed intervals, it is possible to construct a mapping, which is a perfect discrete time model of the system. To embed this into a continuum, the translation equation has to be solved for this mapping. However, in general, neither existence nor uniqueness of solutions can be guaranteed, but fractional iterates of the mapping computed by a neural network can provide regularized solutions that exactly comply with the laws of physics for several examples. Here we extend this method to continuous embeddings which represent the true trajectories of the dynamical system.

1 Introduction

Data very often is sampled at fixed time intervals. If a model is derived from this data it is in general valid only on this grid. Making predictions for other input values means interpolation. Interpolation in general introduces a model: linear, bicubic or spline interpolation are most commonly used. Neural networks often assume to be model-free which just means that one does not really care about the model. Even if the model can represent the training data perfectly, there will be in general some interpolation error.

Think of having time series data sampled monthly and you have constructed a very good model to predict the next month. But now you are requested to make a prediction for next week. Or you have measured extensively how a one meter thick concrete wall blocks some radiation passing through it. Now you are asked how a half meter wall will perform instead. Having a mathematical model with “physical” meaning will usually allow you to address these questions in an intelligent way, but just based on a black-box style model like a neural network this is not a trivial task.

We will investigate this question in an otherwise ideal case and assume the following conditions fulfilled. For simplicity only discrete time-sampling is considered but the arguments can be transferred to space accordingly. Without loss of generality we will set the sampling interval Δt to “1” which will result in a nice formalism later. X_t denotes a vector, describing the state of the system at time t completely. We assume

the system is deterministic and autonomous, there is no external influence and the system is translation invariant, just adding a constant to every time will not change any experiment.

We also assume that we have a very large set of very exact data samples $(X_t, X_{t+\Delta t})$, which represents the *time-one mapping* function f of the system

$$X_{t+1} = f(X_t) \quad (1)$$

arbitrarily well, so that we can derive from it some (neural network) model, which approximates f arbitrarily exact. The discrete-time dynamics of the system is thus totally defined by f and a discrete trajectory of some initial state X_0 can be computed by repeatedly *iterating* the function f :

$$X_t = f^t(X_0) \quad t = 0, 1, \dots \quad (2)$$

Is it theoretically possible to compute the state at other (non-integer) times from this knowledge and if so, how to do it? Mathematically this would mean to find a function $F(X, t)$ for real t , that solves the translation equation

$$F(F(X, s), t) = F(X, s + t)$$

for real s and t under the condition $F(X, n) = f^n(X)$ for integer n . If we formally define $f^t(X) = F(X, t)$ for real t , we extend the concept of iteration to non-integer iteration counts.

2 A Historical View on this Problem

This questions belongs to the mathematical area of *functional equations*. The problem to find the state at time $t = 1/2$ is equivalent to solve the functional equation

$$\varphi(\varphi(x)) = f(x) \quad (3)$$

for the unknown function φ , which is usually called an *iterative root* of f .

This problem dates back to Charles Babbage and his *Essay towards an calculus of functions* from 1815 [1], where he investigates the solutions of $\varphi(\varphi(x)) = x$, the so called *roots of identity*. The widely used notation $\varphi = f^{1/2}$ suggests the idea to allow any exponent for the iteration: $f^{1/n}$ is called the n -th iterative root of f and $f^{m/n}$ a *fractional iterate*, also defined by a functional equation

$$\varphi^m(x) = f^n(x). \quad (4)$$

Abel found 1826 a method to solve some iterative roots, based on linearization of the now so called Abel functional equation $\varphi(f(x)) = \varphi(x + c)$ [2].

In his attempt to find closed expressions h for a given function f under iteration $f^n(x) = h(x, n)$, a problem today usually investigated by chaos theory, Schröder [3] extended the concept of iteration towards real iteration indices and showed how to find solutions of eq. (3) at least near fixpoints of f by solving the now famous Schroeder equation $\varphi(f(x)) = c\varphi(x)$, which represents the eigenvalue problem of functional algebra.

Of special interest e.g. in the theory of computational complexity are iterative roots of the exponential function, i.e. solutions of

$$\varphi(\varphi(x)) = e^x. \quad (5)$$

Hardy showed 1924 [4] that there are infinitely many solutions of eq. 5, but every member of a monotonically rising subset of these solutions has the remarkable property to grow faster than any polynomial but slower than any power function. All solutions depend on some arbitrary chosen function on an interval, so the question came up for a “natural” solution. Kneser found 1938 an analytical solution to this problem in response to a “request from industry” [5].

While Isaacs solved eq. 3 in 1950 [6] and Zimmermann eq. 4 in 1978 [9] for arbitrary self mappings of abstract sets based on the concepts of orbits, these results are hard to apply to concrete questions.

Even more complicated is the task to construct continuous embeddings which take the form of iteration semigroups [8], there exists a close relationship to dynamical systems and chaos theory.

Targonsky related this mathematical questions to a philosophical discussion about the concept of time: If we ever find a physical process represented by a mapping function which is not embeddable or does not have iterative roots of every order, this suggests a minimal time interval, the *chronon* [7].

The current standard opus on this topic is the book “iterative functional equations” by Kuczma et.al. from 1990 which summarizes most known facts about these topics with great detail [11]. But despite this long history the problem of finding iterative roots is still not yet solved. A recent (2001) survey article on the current research on iterative roots states “...one should not expect results on iterative roots in a general situation. In fact, even roots of polynomials are not described. Even worse: we do not know whether every complex cubic polynomial has a square root...” [12]

For polynomials of order two at least the question of existence had been solved by Rice et.al. in “When is $f(f(z)) = az^2 + bz + c$ for all complex z ?” [11] in 1980.

For linear self-mappings, i.e. square matrices, a *matrix power* operator A^t for real t based on eigenvalue decomposition is well known and implemented in numerical packages like Matlab, but for most other cases there is no numerical method published to compute iterative roots of given mappings.

3 Computing Fractional Iterates with a Neural Network

Confronted with an engineering problem from steel industry, we were required to compute iterative roots of a process we only knew by a dataset. The input-output mapping of the whole process could be well modelled by a neural network, but the process con-

sisted of a succession of 7 identical processing steps in a row which should be separated. Thus it was necessary to compute the 7th iterative root of this model. We found some straight forward extensions to the MLP model that can be used to compute approximations of iterative roots and fractional iterates [13].

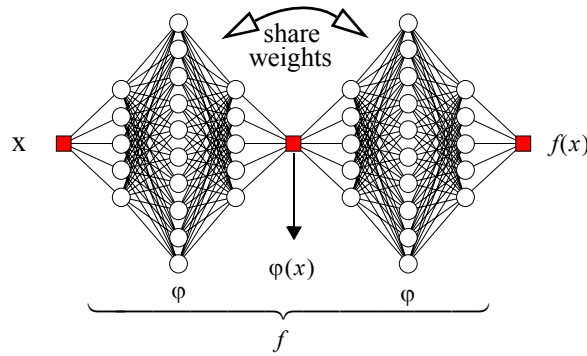


Figure 1 A MLP computes an iterative root of the function f if both subnets which represent ϕ posses identical weights.

The basic idea is to use a structured network like Figure 1 to approximate f and additionally force the weights of the two subnets to be identical. Once both goals are reached, each subnet will be a model for the iterative root of f . Different algorithms for training such networks were presented by us recently [14].

This method can be extended to n -th iterative roots, where a composition of n sub-networks is trained towards f . A fractional iterate $f^{m/n}$ can then be constructed by m of these sub-networks. This is similar to the backpropagation through time formalism except in BPTT the intermediate values are usually known. Another method is to use a recurrent network which is trained in n loops and later recalled in m loops.

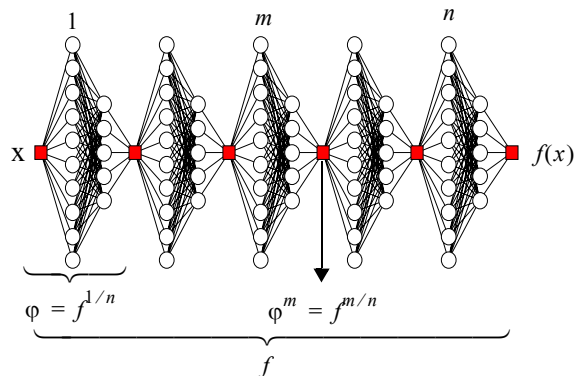


Figure 2 Finding a fractional iterate $f^{m/n}(x)$.

In our paper [16] we could show that several basic experiments from physics can be modelled perfectly by this method for fractions of the measured time interval. But

the number of sublayers is practically limited to about 10 do to learning restrictions imposed by the layer structure [14], so at most the 10th root could be calculated.

4 Continuous Embeddings

To construct a really continuous embedding for *real* t , we now apply three steps:

1. Compute fractional iterates of the desired function (defined by a dataset) as described in chapter 3. The number of sublayers should be selected as high as possible. This will theoretically allow to find the mathematically correct values of the fractional iterates.
2. Extend the original dataset ($X \rightarrow f(X)$) to $(X, t \rightarrow f^t(X))$ where $(X, 1 \rightarrow f(X))$ is the original data and the numerical values for other $t = \frac{1}{n}, \frac{2}{n}, \dots, 1$ are derived from step 1.
3. Use this data to train another network with X and t as inputs and $f^t(X)$ as the desired output.

Of course step 3 is only an interpolation, but should give a reasonable interpolation for any value of t when f is relatively smooth in t . This method will also make a good model for short term extrapolation, i.e. $t > 1$ [15].

4 Example

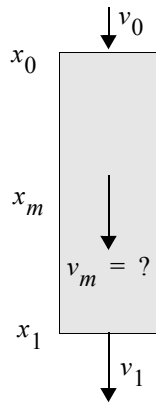


Figure 3 What is the speed of the object at point x_m ?

An object falls down within a closed tube of length $\Delta x = 1$. We inject it with different speeds v_0 and measure its velocity v_1 after passing the tube. The goal is to compute from this data only the speed at any other point within the tube. In case when there is friction involved this is analytically a rather difficult task that involves solving complicated differential equations, even if we know the exact physics of the process. If the exact differential equation is not known, e.g. we do not know which types of frictions are involved, the exact solution may only be computed by numerical approximation anyway.

If we use the method described in this paper and think of the experimental data as a representation of the function f that maps incoming to outgoing speed, $v_1 = f(v_0)$, we could use an iterative root of $f^{1/2}$ to compute the speed at $x = 1/2$. Because of the non-uniqueness of iterative roots, it is necessary to select the solution which conforms to physics, in [16] we demonstrated recently that “small” networks will impose some regularization that will *exactly* select the correct solution for $x = m/n$ with about $n < 10$ and $m \leq n$. The network was able to compute the correct speed at such a point with a relative numerical precision of up to 10^{-6} .

To get a model for the complete continuous function of the speed vs. height, $v = f(v_0, x)$, we used another MLP and trained it with this data. Because it interpolates between these exact points, the mean error increased to about 10^{-5} but it is still a very good model for the actual physical process, especially when considering, that we used no physical laws at all to compute these results, just experimental data.

5 Conclusions

The mathematically as interesting as difficult problem of iterative roots and continuous iteration remains a pretty much unsolved problem with even no standard numerical methods available yet. However, neural networks have successfully proven to give sufficient approximations to some problems from industrial system identification [17] and also showed their capability to model several fundamental experiments from physics exactly. This may open the door to other applications of this mathematical theory.

References

1. C. Babbage: Essay towards the Calculus of functions. Phil. trans. Royal Soc. London 105 (1815) 389-424
2. N. H. Abel: Determination d'une fonction au moyen d'une equation qui ne contient qu'une seule variable. Manuscript 1824, in: Oeuvres compl'etes, Christiania (1881)
3. E. Schröder: Über iterierte Funktionen. Math. Ann. 3 (1871) 296-322
4. G.H. Hardy, E. Cunningham: Orders of infinity. Cambridge Tracts in Mathematics 12 (1924)
5. H. Kneser: Reelle analytische Lösungen der Gleichung $f(f(x)) = e^x$ und verwandter Funktionalgleichungen. J. reine angew. Math. 187 (1950) 56-67
6. R. Isaacs: Iterates of fractional Order. Canad J. Math. 2 (1950) 409-416
7. G. Targonski: An Iteration theoretical approach to the concept of time. Colloques Internationaux du C.N.R.S. 229, Transformations ponctuelles et leurs applications, Toulouse (1973) 259-271
8. M.C. Zdun: Continuous iteration semigroups. Boll. Un. Mat. Ital. 14 A (1977) 65-70
9. G. Zimmermann: Über die Existenz iterativer Wurzeln von Abbildungen. Dissertation, Marburg/Lahn (1978)
10. R.E. Rice, B. Schweizer & A. Sklar: When is $f(f(z)) = az^2 + bz + c$ for all complex z ? Amer. Math. Monthly 87 (1980) 252-263
11. M. Kuczma, B. Choczewski & R. Ger: Iterative Functional Equations. Cambridge University Press, Cambridge (1990)
12. K. Baron & W. Jarczyk: Recent results on functional equations in a single variable, perspectives and open problems. Aequationes Math. 61. (2001), 1-48
13. L. Kindermann: Computing Iterative Roots with Neural Networks. Proc. Fifth Conf. Neural Information Processing, ICONIP (1998) Vol. 2:713-715
14. L. Kindermann & A. Lewandowski: A Comparison of Different Neural Methods for Solving Iterative Roots. Proc. Seventh Int'l Conf. on Neural Information Processing, ICONIP, Taejon (2000) 565-569
15. L. Kindermann & T.P. Trappenberg, Modeling time-varying processes by unfolding the time domain. Proc Int'l Joint Conf. on Neural Networks, IJCNN, Washington DC (1999)
16. L. Kindermann & P. Protzel: Physics without laws - Making exact predictions with data based methods. Proc. Int'l Joint Conf on Neural Networks, Honolulu (2002)
17. L. Kindermann, P. Protzel, F. Schmid & O. Gramckow, (SIEMENS AG): Process and device for determining an intermediate section of a metal strip. International Patent WO9942232 (1999)