4.2 Mapping functional equations to the topology of networks yields a natural interpolation method for time series data

Lars Kindermann
Alfred Wegener Institute
for Polar and Marine Research
Bremerhaven, Germany
lars.kindermann@awi.de

Achim Lewandowski
Austrian Research Institute
for Artificial Intelligence
Vienna, Austria
achim@oefai.at

Abstract

Typically machine learning methods attempt to construct from some limited amount of data a more general model which extends the range of application beyond the available examples. Many methods specifically attempt to be purely data driven, assuming, that everything is contained in the data. On the other hand, there often exists additional abstract knowledge about the system to be modeled, but there is no obvious method how to combine these two domains. We propose the calculus of functional equations as an appropriate language to describe many relations in a way that is more general than a typical parameterized model, but allows to be more specific about the setting than using an universal approximation scheme like neural networks. Symmetries, conservation laws, and concepts like determinism can be expressed this way. Many of these functional equations can be translated into specific network structures and topologies, which will constrain the possible input-output relations of the network to the solution space of the equations. This results in less data that is necessary for training and may lead to more general results, too, that can be derived from the model. As an example, a natural method for inter- or extrapolation of time series is derived, which does not use any fixed interpolation scheme but is automatically constructed from the knowledge/assumption that the data series is generated by an underlying deterministic dynamical system.

1 Introduction

To interpolate data which is sampled in finite, discrete time steps into a continuous signal e.g. for resampling, normally a model has to be introduced for this purpose, like linear interpolation, splines, etc. In this paper we attempt to derive a natural method of interpolation, where the correct model is derived from the data itself, using some general assumptions about the underlying process. Applying the formalism of generalized iteration, iteration semigroups and iterative roots from the mathematical branch of functional equations, we attempt to characterize a method to determine if such a natural interpolation for a given time series exists and give a method for it's calculation, a formal one for linear autoregressive time series and a neural network approximation for the general nonlinear case.

Let x_t be an auto regressive time series: $x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-n}) + \varepsilon_t$. We will not deal here with finding f, i.e. predicting the time series, instead we assume f is already known or already approximately derived from the given data. We will attempt to embed the discrete series of x_t, $t = 0, 1, 2, \ldots$ into a continuous function $x(t)$, $t \in R^+$. To clarify the idea we present the method at first for the case that the timeseries is generated totally deterministically ($\varepsilon_t = 0$) by an underlying autonomous dynamical system. Later we will consider the influences of additional external inputs and noise.

The time evolution of any autonomous dynamical systems is represented by a solution of the translation equation [1],

$$\Phi(x_0, t_1 + t_2) = \Phi(\Phi(x_0, t_1), t_2) \quad (1)$$

where x_0 is an state vector representing an initial condition and t_1, t_2 are arbitrary time intervals. For continuous time dynamical systems this equation holds for every positive t. If we assume that the given time series is a discrete time sampling of an underlying continuously evolving signal, we have to solve (1) under the conditions $\Phi(x, 0) = x$ and $\Phi(x, 1) = f(x)$, where f is the discrete time mapping represented by the data. (Without loss of generality we can assume the sampling rate of the discrete time data to be one, which will result in a nice and very intuitive formalism.)

To double the sampling rate for example, (1) becomes $f(x) = \Phi\left(\Phi\left(x, \frac{1}{2}\right), \frac{1}{2}\right)$.

Substituting $\varphi(x) \equiv \Phi\left(x, \frac{1}{2}\right)$ we get $\varphi(\varphi(x)) = f(x)$, the functional equation for the iterative root of the mapping f [3].

By introducing the formal notation $\Phi(x, t) \equiv f^t(x)$ the connection to iteration theory becomes clearly visible: Time evolution of discrete time systems can be regarded as the iteration of a time step mapping function (iterative map) and this concept extends to continuous time dynamical systems by means of generalized or continuous iteration, allowing for non-integer iteration counts. The following mathematical problems appear [3,4]:

- For a given function f, does there exist the iteration semigroup f^t?,
- Is the solution unique?
- How to calculate it explicitly or numerically.

To apply this theory, usually x has to be a complete state vector of the dynamical system. This means that f has to be a function of the last state only: $x_t = f(x_{t-1})$. When f also depends on earlier values of the time series $x_{t-2}, \ldots, x_{t-n}$, there must be some hidden variables. In order to obtain a self-mapping we introduce the function $F: R^n \to R^n$ which maps the vector

$\vec{x}_{t-1} = \left[x_{t-1}, \ldots, x_{t-n}\right]$ to $\vec{x}_t = \left[x_t, x_{t-1}, \ldots, x_{t-(n-1)}\right]$

with $x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-n})$. Except for the first element this is a trivial time shift operation, each element of $\vec{x}$ is just replaced by its successor. But because F is a self mapping within a n-dimensional space now, time development can be calculated by iterating F and we can try to find the generalized iteration with non-integer iteration counts to find a time continuos embedding F^t, the continuous iteration semigroup of F and extract a function $x(t)$ from this [2].

2 Linear Case

The idea is best demonstrated for the linear case, where it's application simplifies and unifies several problems. For a *linear* autoregessive time series AR(n) model with $x_t = \sum_{k=1}^{n} a_k x_{t-k}$, the mapping F can be written as a square matrix $F = \begin{bmatrix} a_1 & a_2 & \ldots & a_n \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

with the coefficients a_k in the first row and the lower subdiagonal filled with ones. Then we can compute $\vec{x}_t = F \cdot \vec{x}_{t-1}$ and the discrete time evolution of the system can now be calculated using the matrix powers $\vec{x}_{t+n} = F^n \cdot \vec{x}_{t-1}$.

This autoregressive system is called linear embeddable if the matrix power F^t exists also for all real $t \in R^+$. This is the case if F can be decomposed into $F = S \cdot A \cdot S^{-1}$ with A being a diagonal matrix consisting of the eigenvalues λ_i of F and S being an invertible square matrix which columns are the eigenvectors of F. Additionally all λ_i must be non-negative to have a linear and *real* embedding, otherwise we will get a *complex* embedding.

Then we can obtain $F^t = S \cdot A^t \cdot S^{-1}$ with $A^t = \begin{bmatrix} {\lambda_1}^t & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & {\lambda_n}^t \end{bmatrix}$.

Now we have a continuous function $\vec{x}(t) = F^t \cdot \vec{x}_0$ and the interpolation of the original time series $x(t)$ consists of the first element of $\vec{x}$.

In case there is also a constant term, i.e. the mean is not zero, $x_t = \sum_{k=1}^{n} a_k x_{t-k} + \boldsymbol{b}$, we just have to append a constant one to all the vectors $\vec{x}_t = \left[x_t, x_{t-1}, \dots, x_{t-(n-1)}, \mathbf{1}\right]$

and take $F = \begin{bmatrix} a_1 & a_2 & \dots & a_n & \boldsymbol{b} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$.

A special case is $F^{1/2}$, the square root of a matrix, which solves the matrix equation $F^{1/2} \cdot F^{1/2} = F$. It resembles the iterative root of linear functions and corresponds to a doubling of the sampling rate.

A few lines of Maple code can automate this procedure both for symbolic and numeric expressions. A sample worksheet is available at the authors web page.

3 Examples

We will now provide some simple examples to demonstrate this formalism.

3.1 One dimensional linear case

The time series given by some x_0 and $x_t = 2x_{t-1}$ simply doubles every time step. The natural interpolation we immediately get by applying the former formalism in the trivial one dimensional case is $x(t) = 2^t x_0$, which is of course exactly what we expect: exponential growth. But a little change makes the problem much more difficult: If $x_t = ax_{t-1} + b$ we expect a mixture of constant and exponential growth, but what is the exact continuous law?

Take $\vec{x}_t = [x_t, 1]$ then the series is generated by $F = \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix}$. We get immediately by eigenvalue decomposition $\vec{x}(t) = F^t \vec{x}_0 = SA^t S^{-1} \vec{x}_0 = \begin{bmatrix} 1 & 1 \\ \frac{1-a}{b} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & a^t \end{bmatrix} \begin{bmatrix} 0 & \frac{b}{1-a} \\ 1 & \frac{b}{a-1} \end{bmatrix} \begin{bmatrix} x_0 \\ 1 \end{bmatrix}$

and for the first component $x(t) = a^t x_0 + b\frac{a^t - 1}{a-1}$ (which equals $ax_0 + b$ for $t \to 1$).

We don't have to consider about stability or stationarity of the AR(1) model here but note that to obtain a completely real valued function $x(t)$, a has to be positive. Later we will discuss about the meaning of such cases with complex embeddings, but for short it means that there is no one-dimensional continuous time dynamical system that can generate such timeseries. In the linear case this should be clear because a negative a implies oscillatory behavior of $x(t)$. This means, some initial condition x_0 won't be enough to determine the continuation of the trajectory, it could be on the rising or falling slope. The underlying dynamical system needs to have one more hidden dimension to allow embedding. The other dimension can be represented by the imaginary part of $x(t)$ which will vanish at all integer times t. But taking only the real part will still result in a valid interpolation of the given series, the observable of the system.

This is such an embedding of the AR(2) process $x_t = -\frac{1}{2}x_{t-1} + \frac{1}{3}x_{t-2} + \frac{1}{2}$ with $x_0 = x_1 = 1$.

Circles mark the time series x_t, the left graph shows the real part of $x(t)$, our *natural interpolation*, the imaginary part is on the right.

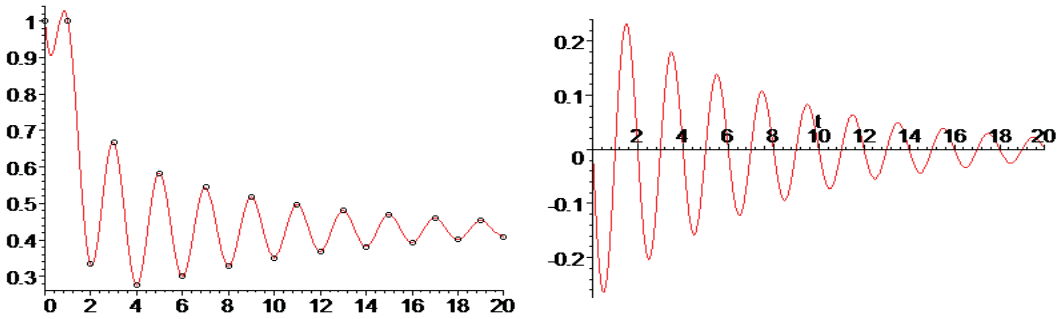

Figure 1: Embedding of an AR(2) process

.

3.2 Two dimensional linear case

The well known Fibonacci series $x_0 = 0$, $x_1 = 1$, $x_t = x_{t-1} + x_{t-2}$ can be generated in this manner by $F = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ and $\vec{x}_1 = [1, 0]$. By eigenvalue decomposition of F we get

$$\vec{x}_{t+1} = F^t \vec{x}_1 = SA^t S^{-1} \vec{x}_1 = \begin{bmatrix} \frac{1+\sqrt{5}}{2} & \frac{1-\sqrt{5}}{2} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \left(\frac{1+\sqrt{5}}{2}\right)^t & 0 \\ 0 & \left(\frac{1-\sqrt{5}}{2}\right)^t \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{1}{2} - \frac{1}{2\sqrt{5}} \\ -\frac{1}{\sqrt{5}} & \frac{1}{2} + \frac{1}{2\sqrt{5}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

which turns out to evaluate exactly to Binet's famous formula for the Fibonacci series in the first component $x_t = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^t - \left(\frac{1-\sqrt{5}}{2}\right)^t\right]$ [5].

Because the second eigenvalue is negative, a *real* linear continuous time embedding does not exist and $x(t)$ takes complex values on non-integer x. Figure 2 shows real and complex part of $x(t)$.

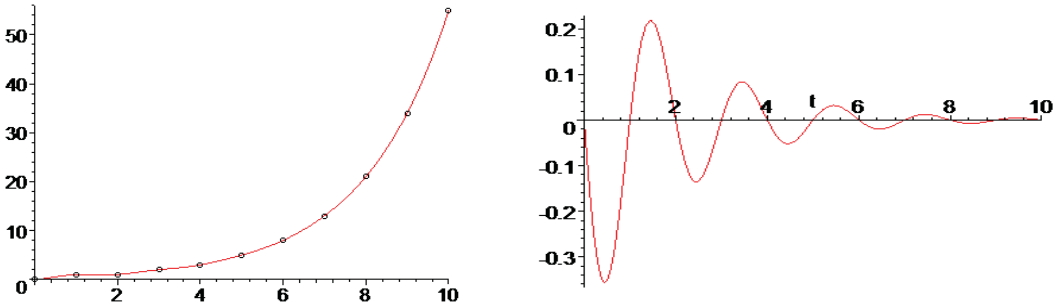

Figure 2: Embedding of the Fibonacci series.

4 The Nonlinear Case

So far we considered only linear dependencies of the past f which could easily be mapped to matrix expressions. The problem becomes much more complicated if we allow for arbitrary f. Even for one dimension this cannot be solved analytically in the general case, so we use neural networks to compute approximations for fractional iterates of arbitrary functions [7].

4.1 One dimensional nonlinear systems

The general solution of the real valued translation equation $\Phi(x_0, t_1 + t_2) = \Phi(\Phi(x_0, t_1), t_2)$ $\Phi: R \times R \to R$ with Φ being continuous and strictly monotonic in x and t is given by $\Phi(x, t) = \varphi^{-1}(\varphi(x) + t)$. If the discrete time mapping $\Phi(x, 1) = f(x)$ is given, this is Abel's functional equation $f(x) = \varphi^{-1}(\varphi(x) + 1)$. If f has a fix point this can further be transformed to Schröder's functional equation $f(x) = \varphi^{-1}(c\varphi(x))$, the eigenvalue problem for nonlinear functions. In those cases when either of these equations can be solved for the unknown function φ, it is immediately possible to obtain the embedding by $f^t(x) = \varphi^{-1}(\varphi(x) + t)$ [2].

This can be easily solved for example in the linear case: If we take $x_t = ax_{t-1}$ and initial value x_0 the time step mapping is $f(x) = ax$, we get the Abel type functional equation $ax = \varphi^{-1}(\varphi(x) + 1)$ or $\varphi(ax) = \varphi(x) + 1$, which is solved by $\varphi(x) = \log_a x$. So we get for the continuous embedding the expected result again, exponential growth $f^t(x) = a^{\log_a x_0 + t} = x_0 a^t$.

However, this analytical method is limited to a small selection of functions and it can be shown that there exist embeddings for a much wider range of mappings which cannot be calculated analytically yet. Furthermore the theory so far is developed mainly for real or complex valued functions, solving Abel or Schröder type functional equations in higher dimensions is currently for the general case beyond reach.

But simple neural networks can be used to find precise approximations for those embeddings [7,8]. The basic idea is to use a MLP with a special topology which approximates $f(x)$. To compute the *fractional iterate* $f^{m/n}$, we use a network that consists of n subnetworks in a row with *pairwise identical weight matrices*. The use of special training algorithm allows to perform the function approximation with the whole network and keep the subnets identical at the same time [9]. The fractional iterate of the function can be read out after the m-th subnet.

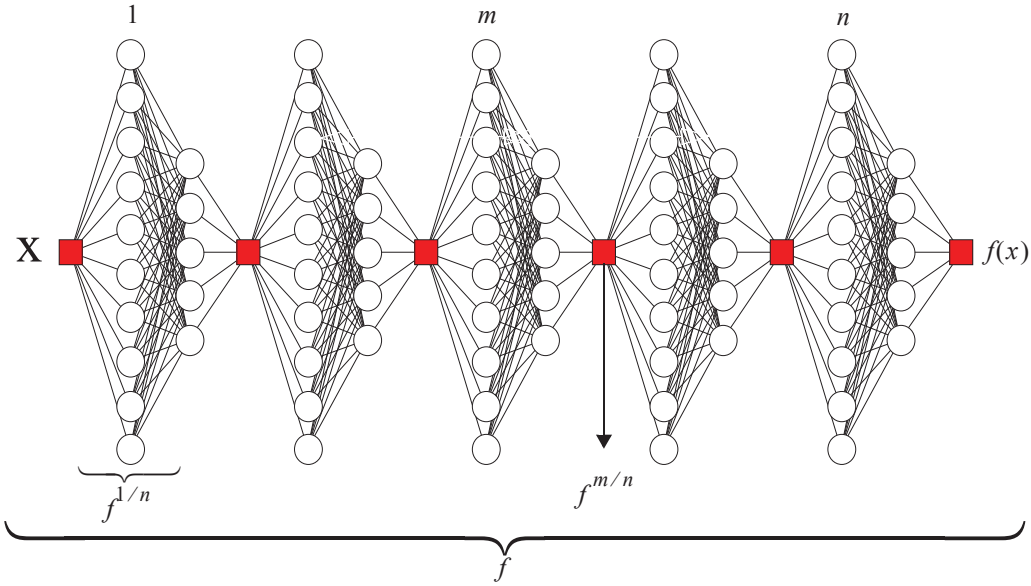

Figure 3: MLP for computing fractional iterates.

4.2 Multidimensional nonlinear system

We took a time series of yearly snapshots from a discrete non linear Lotka-Volterra type predator - prey system (x = hare, y = lynx) as training data.

$$x_{t+1} = (1 + a - b\, y_t)\, x_t$$

$$y_{t+1} = (1 - c + d\, x_t)\, y_t$$

From these samples only we calculated the monthly population by use of a neural network based method to compute iterative roots and fractional iterates with a pseudo newton algorithm [9].

This figure shows the yearly training data as circles and the interpolated monthly values. Additionally the forecasted values for the next 12 months are shown, together with the true value after one year which was excluded from model fitting.

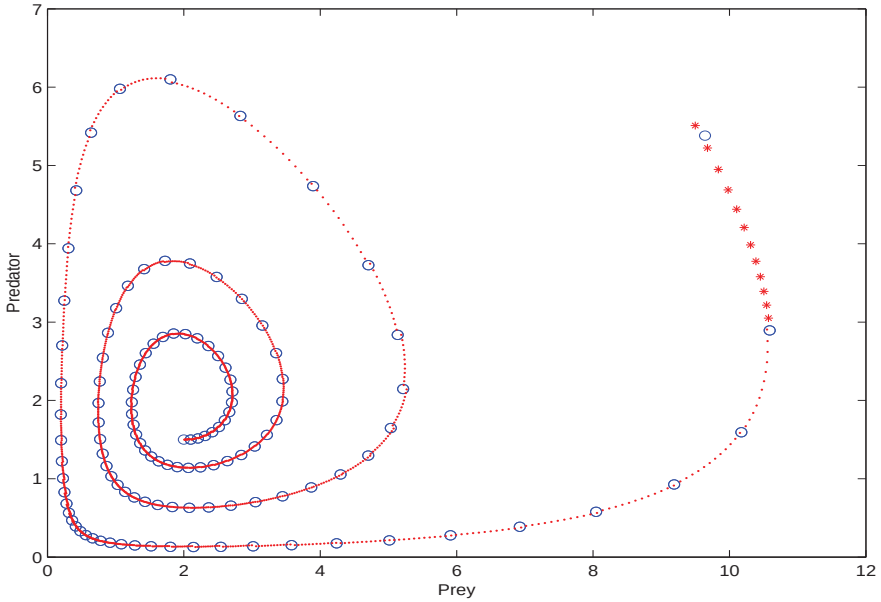

Figure 4: Embedding of a Volterra type system.

The given method provides a natural way to estimate not only the values over a year, but also to extrapolate arbitrarily smooth into the future.

5 Discussion and Outlook

The method demonstrates that there is a close relation between prediction and interpolation. A necessary condition for the existence of a natural interpolation of a time series is predictability. If there are random influences and we require that the values $x(t)$ coincide with x_t for integer t, we can still use the embedding function to get a near fit and add an additional interpolation method for the residuals $x_t - x(t)$. This has again to be selected freely of course, but it minimizes the amount of arbitrariness involved in interpolating.

Another problem are impossible embeddings. Take $x_{t+1} = 4\lambda x_t(1-x_t)$, the iterated logistic map, which is a favorite textbook example for the emergence of chaotic behavior within a simple dynamical system. However, this is a discrete time system, so the question should arise naturally if it is possible to embed the x_t into continuous trajectories $x(t)$ which now obey the functional equation $x(t+1) = 4\lambda x(t)(1-x(t))$ for any non-integer t. Or even more general, is there *any* continuous time system that takes the same values at integer times? Iteration theory proofs that the answer is no if $\lambda > 3/2$ [6], but this could be expected also by the theorem of Poincare-Bendixon, which implies that chaotic behavior is impossible in continuous time systems of less than three dimensions. To obtain a continuous embedding of this series, we had to introduce some hidden dimension, like allowing complex values for *x*, in iteration theory these generalized solutions are called phantom roots of functions [1]. In neural networks this could be accomplished by introducing additional hidden nodes, allowing to address the general embedding problem.

Acknowledgements

Part of the work was conducted at the RIKEN Brain Science Institute, Wako-shi, Japan.
The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education Science and Culture.

References

1. G. Targonski: *Topics in Iteration Theory.* Vandenhoeck und Ruprecht, Göttingen (1981)
2. M.C. Zdun: *Continuous iteration semigroups.* Boll. Un. Mat. Ital. 14 A (1977) 65-70
3. M. Kuczma, B. Choczewski & R. Ger: *Iterative Functional Equations.* Cambridge University Press, Cambridge (1990)
4. K. Baron & W. Jarczyk: *Recent results on functional equations in a single variable, perspectives and open problems.* Aequationes Math. 61. (2001), 1-48
5. R.L. Graham, D.E. Knuth & O. Patashnik: *Concrete Mathematics.* Addison-Wesley, Massachusetts (1994)
6. 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
7. L. Kindermann: *Computing Iterative Roots with Neural Networks*. Proc. Fifth Conf. Neural Information Processing, ICONIP (1998) Vol. 2:713-715
8. E. Castillo, A. Cobo, J.M Gutiérrez & R.E Pruneda: *Functional Networks with Applications. A Neural-Based Paradigm*. Kluwer Academic Publishers, Boston/Dordrecht/London (1999)
9. 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