MODELLING ITERATIVE ROOTS OF MAPPINGS IN MULTIDIMENSIONAL SPACES

Lars Kindermann
RIKEN Brain Science Institute
Lab for Mathematical Neuroscience
kindermann@brain.riken.go.jp
www.mns.brain.riken.go.jp/~kinderma

Pando Georgiev
RIKEN Brain Science Institute
Lab for Advanced Brain Signal Processing
georgiev@bsp.brain.riken.go.jp
www.bsp.brain.riken.go.jp/~pando

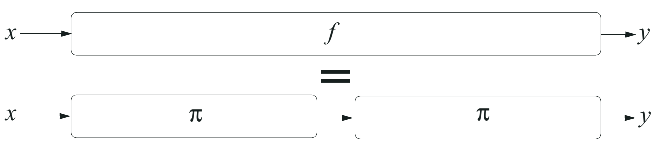
Solutions π(x) of the functional equation π(π(x)) = f(x) are called iterative roots of the given function f(x). They are of interest in dynamical systems, chaos and complexity theory and also in the modelling of certain industrial and financial processes. Furthermore they are related to the optimal interpolation method. But the problem of computing this "square root" in function (or operator) spaces remains a hard task and is, for the general case, still unsolved. While the theory of functional equations provides some insight for real and complex valued functions, iterative roots of mappings from R^n to R^n are not well understood by mathematics and there even exist no published numerical algorithms for their computation. Here we prove existence of iterative roots of a certain class of monotonic mappings in R^n spaces and demonstrate how a method based on neural networks can find their solutions. This is demonstrated on some examples that model fundamental physical systems.

Iterative Roots

Given a self mapping f of some set X onto itself, the mapping π is called an iterative root of f, if it solves the functional equation

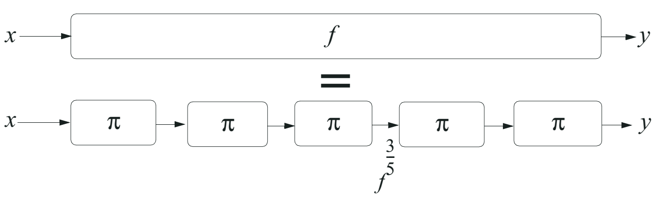
$$\pi \circ \pi = f$$

Finding π can be considered as the inverse problem of iteration. For real or complex valued functions $f(x)$, the equation $\pi(\pi(x)) = f(x)$ is discussed in mathematics since a long time.



However, the problem can be solved analytically for a limited set of functions only, even if existence of solutions may be proved more easily. E.g. every continuous monotonic rising real valued function has iterative roots, but it is not possible to give a closed form solution for $\pi(\pi(x)) = e^x$.

In the manner of writing $\pi^2 = f$, an iterative root of f is often denoted as $\pi = f^{1/2}$. Furthermore, solutions of $\pi^n = f^m$ generalize the concept of iteration to non-integer iteration counts: $\pi = f^{m/n}$ is the fractional iterate of f.



Application to Dynamical Systems

Let X be the state space of a dynamical system, $x_t \in X$ the actual state at time t. Let f be the *time one mapping* of the system: $x_{t+1} = f(x_t)$. The dynamics of a *discrete time* dynamical system is completely defined by the *iterated function system* $f^t(x_0)$, $t = 0 \ldots \infty$ with $f^0(x) = x$ and $f^{t+1}(x) = f(f^t(x))$.

A famous textbook example of a discrete time dynamical systems based on an iterated function is the logistic map which show a very complicated chaotic behavior: $f(x) = \lambda x(1-x)$. Hundreds of papers have been written about iterating this function, but only one [1] asks, if this process can be extended into the other direction: Is $f(x)$ the result of some other iteration itself? Is it the stroboscopic sampling of a continuous evolution process? Or: is there a solution of the equation $\pi^t(x) = \lambda x(1-x)$.

Interpolating Experimental Data

Experimental data in general can be sampled only in discrete or finite time steps. Models derived from this data are strictly speaking discrete time models only. To get a continuous time description (or higher resolution at least), some kind of interpolation is necessary. And the true or "natural" interpolation will necessarily be a fractional iterate of this model.

The traditional analytical way of dealing with these kind of problems is trying to find a differential equation that will describe this behavior. However, to derive a continuous trajectory $f(x_0, t)$, this differential equation has to be integrated. But only in simple cases this is possible analytically, in general a numerical method will be used to approximate the infinitesimal dt's by some small Δt's. The whole process consists of three steps:

1) Data is sampled in finite time steps. (Time series)
2) Find a theory or model, usually a differential equation that describes the underlying continuous process.
3) It is numerically solved (integrated) by simulating very small finite time steps again.

If we are not interested in scientific theory, it might be possible to avoid step 2, (which is rather difficult to take by artificial intelligence) and compute the small timesteps to simulate a continuous trajectory directly by means of iterative roots. But the mathematical theory of iterative roots is well developed only for real and complex valued functions, while dynamical systems in general live in a higher dimensional state space. So we attempt now to extend several theoretical findings and a neural network based method for the approximation of iterative roots to the n-dimensional case.

Monotone mappings in R^n

It is well known that continuous monotone iterative roots of all orders exist for continuous strictly monotone function on the real line. We could generalize this result for the multidimensional case of strictly monotone mappings defined in R^n and prove the following:

Let C be a closed, convex, pointed cone $(C \cap (-C) = \{0\})$ in R^n with $\mathrm{int}\,C \neq \emptyset$. We shall write $x \geq_C y$ if and only if $x \in y + C$, and $x >_C y$ if and only if $x \in y + \mathrm{int}\,C$.

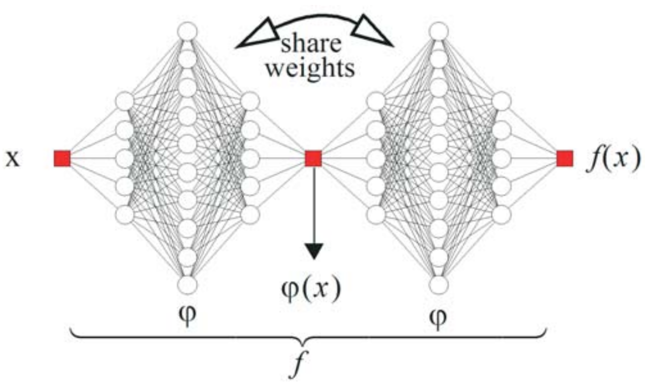
Definition: The mapping $F: R^n \to R^n$ is said to be monotone (resp. strictly monotone) with respect to C, iff $x \geq_C y$ (resp. $x >_C y$) implies $F(x) \geq_C F(y)$ (resp. $F(x) >_C F(y)$) for any $x, y \in R^n$.

Theorem: Suppose that $F: R^n \to R^n$ is a continuous strictly monotone mapping with respect to the cone C, which is coercive (i. e. $\lim_{\|x\| \to \infty} \|F(x)\| \to \infty$) and the inverse mapping F^{-1} is strictly monotone with respect to the cone $-C$.
Then for any initial point x_0 satisfying $F(x_0) >_C x_0$ there exists a continuous strictly monotone iterative root of $F_{|D}$ (F restricted to D) where $D = x_0 + C$, i.e. there exists a continuous strictly monotone mapping $\lambda: D \to R^n$ such that $\lambda(\lambda(x)) = F(x)$ for any $x \in D$.

This proves the existence of iterative roots for a very special case of high dimensional mappings only, but also in the real or complex domain proofs for more complicated functions are difficult. For example it is still an open question, whether all polynomials of order 3 have iterative roots or not! A general answer is still far out of reach, however, in the case of data sampled from a continuously evolving system it is clear by evidence that there must be iterative roots of all orders!

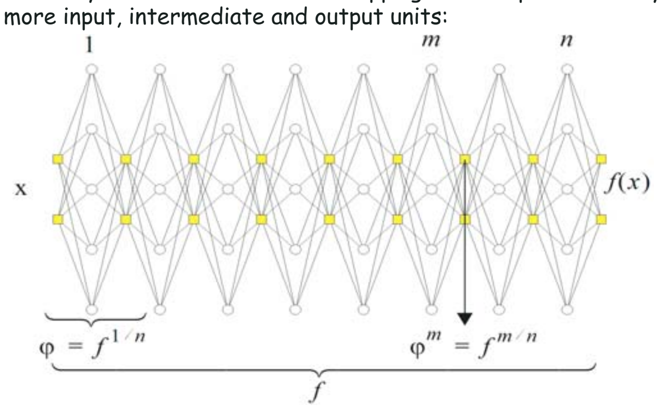
Neural Network Model

The idea how to map the problem of finding iterative roots of a given function to a neural network is straightforward:



A network of this topology with the additional constraint that all the weights in both subnets are pairwise identical, is trained to approximate f. If this succeeds, each of the two identical subnets approximates it's iterative root!

This scheme can easily be extended to compute higher order iterative roots and fractional iterates by simply introducing more layers. Multidimensional mappings are represented by more input, intermediate and output units:



The main problem is to find a training algorithm for these networks, that incorporates the constraints of equal weights in all subnets and still has a fast convergence behavior.

We derived previously two methods that give good results. One starts with different subnets and uses a penalty term, that assigns an error to the sum of all squared differences of corresponding weights in the subnetworks. The second one calculates the exact derivatives in the case of initially identical subnetworks (or a finitely recurrent network) which can be used in a slightly modified pseudo Newton algorithm.

Initialization

The more complex the function gets, the more difficult it becomes for the network to find a solution for it's iterative roots. Local minima become a serious problem with this network topology. Therefore it becomes important to start somewhere near the desired solution if possible.
While it is difficult to calculate an exact solution of the iterative root, it can often be guessed how it looks like. This guess can be used to pre-train all subnetworks before the whole network is trained. For monotone mappings a linear interpolation method can be used to find a rough approximation which speeds up convergence dramatically and avoids training failures due to local minima.

Examples

We will demonstrate this method on some simple physical systems to reconstruct trajectories from sampled data:

Free Fall

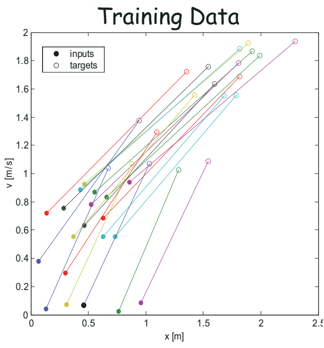
The state of falling body is completely defined by its position x and speed v. For the free fall with gravitational acceleration a we know how an initial state (x_0, v_0) evolves in time:

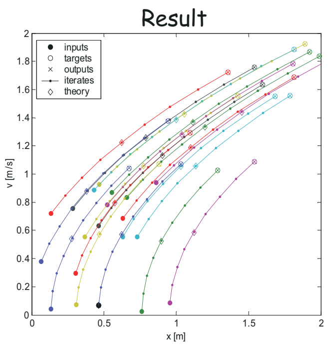
$$x(t) = x_0 + v_0 t + \frac{1}{2}at^2,\; v(t) = v_0 + at$$

From this system we generated a set of data pairs (x_0, v_0), $(x(\Delta t), v(\Delta t))$ as training examples with a fixed time interval Δt. Thus the free fall of an object, measured in one second steps is given by the iterative map

$$v_{t+1} = v_t + a,\; x_{t+1} = x_t + v_t + \tfrac{1}{2}a\,.$$

The figure shows initial states as closed circles and final states as open circles. To reconstruct the complete trajectories from this data, we trained the network from figure 2 with this data and computed the fractional iterates up to order 8. The (2 - 5 - 2)^8 network was able to reproduce the targets with an accuracy (relative mean square error) of 10^{-8} and the iterative root with MSE of 10^{-6}:



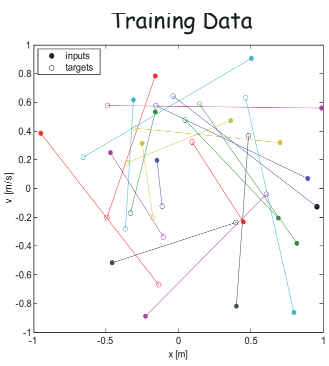


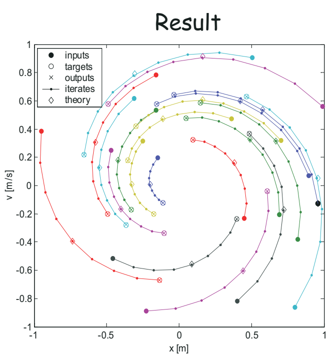
Damped Harmonic Oscillator

The state of a harmonic damped oscillator

$$x(t) = A\sin(\omega t - \varphi)e^{-ct},\; v(t) = \dot{x}(t)$$

was sampled stroboscopically at fixed time intervals Δt. We used the same network again to compute the 8th fractional iterate on this data set to calculate trajectories with an 8-fold increased time resolution. The diamonds show the expected positions at times $\Delta t/8$ which are again modelled perfectly by the network. The accuracy is about the same as above.





Conclusion

It is possible to extend known properties of iterative roots to the multidimensional case and neural networks still provide an excellent method to compute approximate solutions here.