Linear interpolation of nonlinearities in Cellular Neural Networks (CNN)

Gunter Geis, Michael Reinisch, Ronald Tetzlaff, Frank Puffer

Institute of Applied Physics Johann Wolfgang Goethe University Frankfurt / Main

ABSTRACT: During the past few years an increasing number of researchers have started to investigate CNN with nonlinear weight functions for various applications. In different investigations nonlinear weights are represented using polynomials or a tabulated form combined with a cubic spline interpolation procedure. In this contribution we propose a new method for representing nonlinearities in CNN by using a linear interpolation technique.

1. Introduction

Recently in different investigations CNN with nonlinear weight functions are considered for various applications [1], [2], [3]. In order to enable a parameterised representation of a broad class of nonlinearities, polynomial approximations [4], [5] and tabulated functions combined with a cubic spline interpolation [6], [7], have been introduced and analysed in detail for different cases. Although, the results of these studies show, that generally a precise representation can be obtained, complexity reduced methods are preferable especially for CNN-UM realizations. In this contribution a new linear interpolation procedure leading to a accurate representation, also of strong nonlinearities, is proposed and discussed in detail. Therefor firstly a standard linear interpolation with equally spaced samples is applied. Secondly for a certain maximum approximation error redundant samples are eliminated leading finally to a representation with non-equidistant samples. In section 2. the interpolation algorithm, which is a kind of greedy algorithm, is introduced followed by results in section 3. and 4. obtained for nonlinear wave solutions of an one-dimensional Korteweg-de Vries equation.

2. The Table Minimising Algorithm (TMA)

For an elimination of samples in a linear interpolation as far as possible fullfilling an error ϵ constraint, we use a kind of a greedy algorithm which is in general defined by [8] as:

Definition 1 (greedy algorithm). An algorithm that always takes the best immediate, or local, solution while finding an answer. Greedy algorithms find the overall, or globally, optimal solution for some optimisation problems, but may find less-than-optimal solutions for some instances of other problems.

Such algorithms are much faster than algorithms that always finds the global extremum. The deviation compared to an global optimisation algorithm is in most cases negligible. We would like to describe the used algorithm on the basis of an example as shown in Fig. 1. As input the greedy algorithm needs, apart from the equidistant sample points x_i and their function values, a

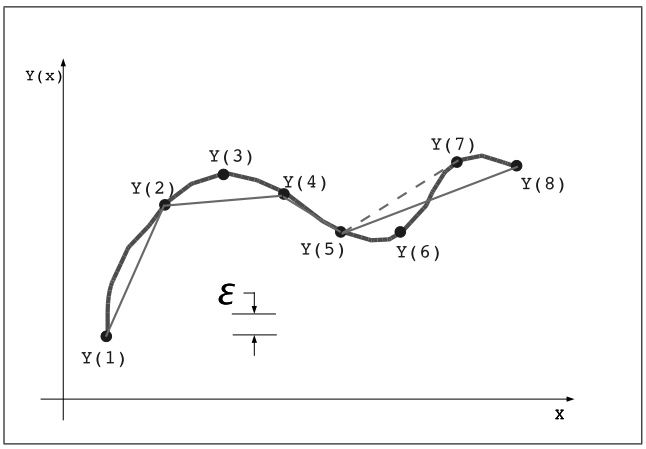

Figure 1: Example of non equidistant interpolation using an greedy algorithm

maximal acceptable error ϵ and a window of size R. For a given tabulated function $y_i = y(x_i)$ with N sample points the algorithm iteratively runs from a found point x_i – at the beginning the first x_1 – to the farthest next point fulfilling an error constraint, and at the end to the last sample point. In each step simply a linear representation g_k according to

$$g_k = \frac{y_{i+R} - y_i}{x_{i+R} - x_i} x_k - \frac{y_{i+R} - y_i}{x_{i+R} - x_i} x_i - y_i \quad \text{with} \tag{1}$$

$$k = i + R - 1, i + R - 2, \ldots, i + 1$$

is taken in the interval $[x_i, x_i + R]$. Each sample point in between x_i and $x_i + R$ is considered if the error

$$e = |y_k - g_k| \quad \forall k = i + R - 1, i + R - 2, \ldots, i + 1 \tag{2}$$

between g_k and the given function y_k does not exceed the given ϵ for each sample point. Then, the next interval is considered by setting $x_i = x_i + R$, and all sample points between the former x_i and $x_i + R$ are removed. If not, R is decremented for this partial step and the process is started again. Finally, if $x_i + R$ would exceed the end of the table, the last sample point is taken. Summarising, a sample point will be taken, which has the largest distance to x_i still leading to an error e smaller than ϵ for all points in between. Thus, the number of sample points taken for the linear interpolation is minimised.

It should be noted that the obtained solution is not optimal, due to the fact that not all combinations are evaluated, since the program stops as soon as a suitable sample point has been found. In a more general examination it may occur that one could eliminate more points in a further step by checking other combinations which also fulfil the error constraint. An example is shown in Fig. 1; regarding the position $x_i = 2$, i.e. $y(2)$, the algorithm stops at $y(4)$ owing to the result that $e < \epsilon$ and the next found sample point lies inevitably at $y(5)$. If instead one would go to $y(3)$, the point $y(6)$ would be the next; and thus requires one sample point less. Although, the general procedure may lead in some cases to improved performance, the substantial reduction of the computing complexity by applying the proposed greedy algorithm is attractive for several applications in practice. For the so-called computation costs $T(N)$ [9] follows $T(N) = O(N^2)$ in a general optimization procedure. By applying the greedy algorithm only the worst-case

reaches $T(N) = O(N^2)$. In the 'best-case' it follows $T(N) = N - 1$ and in the 'average-case' $T(N) = O(N)$. However, in the optimal solution case there is always the complexity as given above for the general optimization procedure.

3. Linear interpolation of nonlinear functions

The performance of the TMA has been evaluated in several cases, a typical result is given below. Since this algorithm may be used to reduce the representation complexity of a given tabulated function, it could be applied by adapting R for a given error ϵ. In the following Fig. 2 the numbers of remaining sample points vs. the error ϵ (absolute) and the window size R (in sample points) are given for the polynomial

$$y(x) \quad = \quad 1.0x + 50.0x^2 - 17.0x^3 \tag{3}$$

which is tabulated with $N = 100$ equidistant samples in the interval $[0, 5]$.

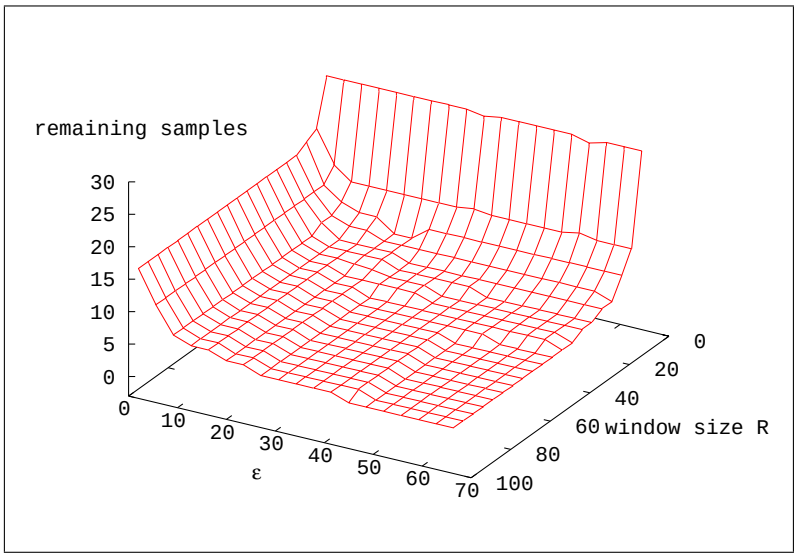

Figure 2: Remaining sample points vs. ϵ and R

Significant changes of remaining points can only be observed for small values of ϵ. By decreasing the window size R, more sample points remain in the resulting table. This follows because in the best case, for a given window size R all sample points between x_i and $x_i + R$ are removed, so decreasing R leeds finally to one point per window. In Fig. 3 a typical result of the TMA for $\epsilon = 20$ and $R = 50$ is shown which is compared to the function $y(x)$ of Eq. 3 and the corresponding absolute error values e for each sample point are given. It is evident that the error increases between remaining function values $y(x)$ up to the maximum acceptable value ϵ. It can be observed (marked with a rectangle) that about $x = 0.5$ the error takes the value $e = 0.0$, because the curve of the function approximation obtained in a linear interpolation with non equidistant points show a crossing of the function curve $y(x)$.

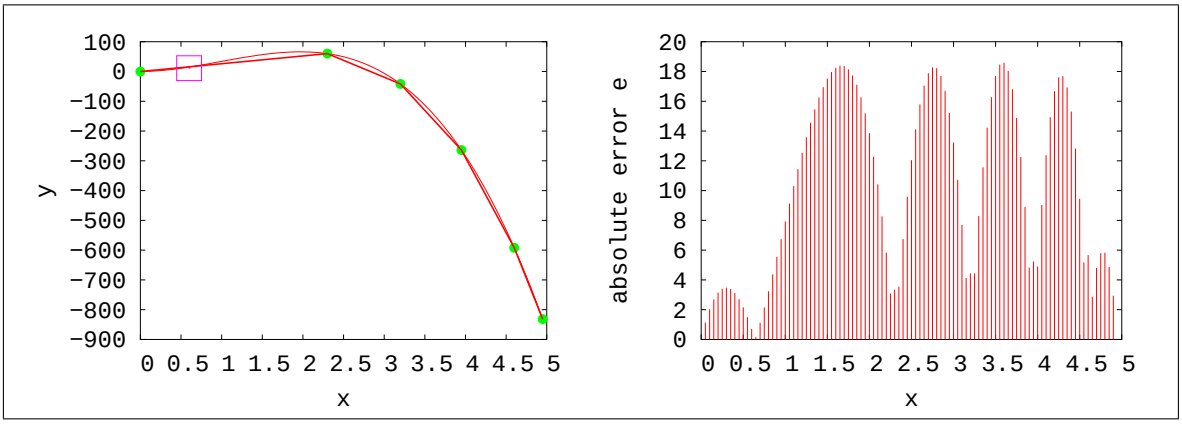

Figure 3: Result of the TMA by approximating $y(x)$ given in Eq. (3) for $\epsilon = 20$ and $R = 50$ (left side) the interpolation result is compared to $y(x)$ (right side) approximation error e vs. x

4. Nonlinear weight functions of CNN

We present in the following first results using the TMA by representing nonlinear weight functions in CNN templates. Hereby autonomous CNN with states $s(t)$ and cell outputs $o(t) = id(s(t))$ are defined by the state equation

$$\frac{ds_i(t)}{dt} = -cs_i(t) + q \quad + \sum_{j \in \mathcal{N}(i)} a_{j-i}\left(o_i(t), o_j(t)\right) \tag{4}$$

with the neighbourhood $\mathcal{N}(i)$ and the polynomial weights

$$a_{j-i}\left(o_i(t), o_j(t)\right) \quad = \sum_{\substack{k+l \le D \\ k \ge 0, l \ge 1}} p_{a,j-i,k,l} (o_i(t))^k (o_j(t))^l \tag{5}$$

The Korteweg-de Vries-equation (KdV equation)[10], [11]

$$\frac{\partial u(x,t)}{\partial t} \quad = \quad 2\frac{\partial^3 u(x,t)}{\partial x^3} - \frac{\partial u^2(x,t)}{\partial x} \tag{6}$$

is a nonlinear partial differential equation. For simulating the KdV it is necessary to perform a spatial discretisation of Eq. (6) . We can approximate the two spatial derivatives as

$$\frac{\partial^3 u(x,t)}{\partial x^3} \quad \approx \quad \frac{u(x_{i+2},t) - 2u(x_{i+1},t) + 2u(x_{i-1},t) - u(x_{i-2},t)}{2(\Delta x)^3} \tag{7}$$

$$\frac{\partial u^2(x,t)}{\partial x} \quad \approx \quad \frac{u^2(x_{i-1},t) - u^2(x_{i+1},t)}{2\Delta x}. \tag{8}$$

Substituting Eq. (7) and (8) into Eq. (6), we obtain the following equation:

$$\begin{aligned} \frac{\partial u(x_i,t)}{\partial t} \quad \approx \quad & \frac{u(x_{i+2},t) - 2u(x_{i+1},t) + 2u(x_{i-1},t) - u(x_{i-2},t)}{(\Delta x)^3} \\ & - \frac{u^2(x_{i-1},t) - u^2(x_{i+1},t)}{2\Delta x} \end{aligned} \tag{9}$$

$-1.0s$	$2.0s - 0.5s^2$	0.0	$-2.0s + 0.5s^2$	$1.0s$

Table 1: KdV template function

In the following investigations the template given in Table 1 is used, which results from Eq. (9) for $\Delta x = 1$, and is considered for a one dimensional CNN with 128 cells. Fig. 4 represent the initial state of the CNN. In the simulation the 4-step Runge-Kutta method is used. In Fig. 5 the states of the CNN after 30000 simulation steps are shown. On the left side the results represent the state s of the CNN solving the KdV equation using above mentioned weights. In the middle

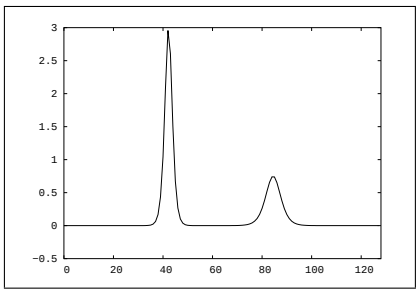

Figure 4: Initial state used in the simulation of the KdV Eq. (9).

part of Fig. 5 a CNN solution obtained by using linear interpolated weights in $[-50, 50]$ with equidistant points having a step size of $\Delta t = 0.01$. On the right side the results which has

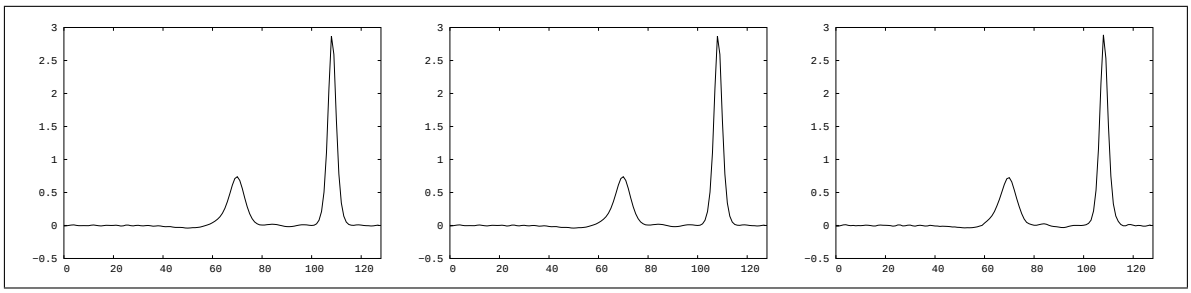

Figure 5: Left side: KdV equation using a polynominal representation for the weights; middle: using equidistant tabulated representation for the polynominal weights; and right side: CNN solution calculated using the reduced tabulated weights.

been obtained using a reduced tabulated version of the polynomial nonlinearity evaluated by means of the TMA taking an absolute error of $\epsilon = 3.0$ and a window size of $R = 20$. As compared to the linear approximation wDocumentsith N_{lin} equidistant points, the application of the TMA led to weight function representation with N_{TMA} non equidistant points giving a factor $\frac{N_{lin}}{N_{TMA}} \approx 20$.

5. Conclusions

We have shown in this paper that the application of the table minimising algorithm lead in all treated cases to a precise representation of nonlinear functions by using a linear interpolation of a small number of non-equidistant points. It is obvious that an implementation of such an linear interpolation algorithm is flexible with a low complexity and possibly solve the problem of finding a parametrised representation of nonlinear weight functions in CNN.

References

[1] Mika Laiho, "Mixed-Mode Cellular Array Processor Realisation for analyzing Brain Electrical Activity in Epilepsy", Helsinki University of Terchnology, Electronic Circuit Design Laboratory, 2003.

[2] R. Kunz, R. Tetzlaff, "Spatio-Temporal Dynamics of Brain Electrical Activity in Epilepsy: Analysis with Cellular Neural Networks (CNN)", Journal of Circuits, Systems and Computers, in print.

[3] D. Feiden, R. Schoenmeyer, R. Tetzlaff, "On-Chip template Training for Pattern Matching by Cellular Neural Networks Universal Machines" ISCAS 2003, Bangkok, 2003.

[4] F. Puffer, R. Tetzlaff, D. Wolf, "A Learning Algorithm for the Dynamics of CNN with Nonlinear Templates Part II: Continuous-Time Case", 4^{th}CNNA'96, Seville, pp. 467-472, 1996.

[5] F. Puffer, R. Tetzlaff, D. Wolf,Documents "Cellular Neural Networks with Nonlinear Weight Functions – Applications to Texture Classification" European Conf. on Circuit Theory and Design, Budapest, 1997

[6] R.Tetzlaff, A. Loncar, D.Wolf, "Modeling Chaotic Systems by Cellular Neural Network", ICNF'99, Hong Kong, 1999.

[7] A. Loncar, R.Tetzlaff "Cellular Neural Networks with nearly arbitrary nonlinear weight functions", 6^{th}CNNA'00, Seville, 2000.

[8] National Institute of Standards and Technology "Dictionary of Algorithms and Data Structures", http://www.nist.gov/dads/HTML/greedyalgo.html

[9] T.H. Cormen, C.E. Leiserson, R.L. Rivest, "Introducing to Algorithms", MIT Press Cambridge Massachusetts, 1999.

[10] F. Puffer,R.Tetzlaff, D.Wolf, "A Learning Algorithm for Cellular Neural Networks (CNN) solving nonlinear Partial Differential Equations" ISSSE'95, San Francisco, pp. 105-405, 1995.

[11] F. Puffer,R.Tetzlaff, D.Wolf, "Modeling Nonlinear Systems with Cellular Neural Networks", ICASSP'96, Atlanta , pp.3513-3516, 1996.