Universal Representation of Image Functions by the Sprecher Construction

Mario Köppen[1] and Kaori Yoshida[2]

[1] Fraunhofer IPK, Pascalstr. 8-9, 10587 Berlin, Germany
mario.koeppen@ipk.fraunhofer.de
[2] Kyushu Institute of Technology, 680-4 Kawazu, 820-8502 Iizuka City, Japan
kaori@ai.kyutech.ac.jp

Summary. This paper proposes a procedure for representing image functions by a computation in two layers. It is recalled that the general function representation needs more layers than two, using the Stone-Weierstrass theorem for approximation in three layers, and the Kolmogorov theorem for representation in four layers. For achieving representation in two layers only, the requirement on a continuous representation has to removed. The Sprecher construction presented here is a general procedure for yielding such a representation in two layers. It can be used to compress images, to represent pixels and their neighborhoods directly, or to represent image operators.

1 Introduction

This paper is concerned with the representation of a continuous, bounded real-valued function of n variables by the superposition of functions of lower number of variables, and how this representation applies to the representation of image functions in image processing. The universal possibility of such a representation is granted by the KOLMOGOROV theorem [4]. It was Hecht-Nielsen [2], who rediscovered the importance of the KOLMOGOROV theorem for the theoretical understanding of the universal representation abilities of neural networks. The KOLMOGOROV theorem was also pointed out to be of importance for other designs of soft computing, as e.g. normal forms in fuzzy logic [6] [7]. The constructive aspects of this representation can be followed in a line of studies presented so far e.g. in [8], [1], [10], [11] and [5].

The starting point is the Kolmogorov theorem (here in the notation used by David Sprecher [10]):

Theorem 1 (Sprecher, 1996). *Every continuous function $f : I^n \to R$ can be represented as a sum of continuous real-valued functions:*

$$f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \chi_q(y_q). \tag{1}$$

In this representation, the $x_1, \ldots, x_n$ are the parameters of an embedding of I_n into R_{2n+1}:

$$y_q = \eta_q(x) = \sum_{p=1}^{n} \lambda_p \psi(x_p + qa) \tag{2}$$

with a continuous real-valued function ψ and suitable constants λ_p and a. This embedding is independent of f.

In [10] and [11], Sprecher gives a numerical procedure for computing the "inner" function ψ and setting the "outer" function" χ. that was corrected in order to give a continuous ψ function in [5].

The functions used in such a representation came out to be untractable in a practical sense (due to the pseudo-fractal nature of the "inner" function). Regarding images, in [5] it was demonstrated how even the second stage of approximating such a Kolmogorov representation needs intermediate storage of 10^6 data objects, while the next stage would demand 10^9 data objects to be stored. Even today, with more and more computer memory becoming available, this technical demand is hard to meet.

In this paper, the numerical procedures are extended for the case of a non-continuous representation, to have a means for representing general image functions in image processing in a more practical manner. This paper is organized as follows: section 2 summarizes what is known today about general function approximation and representation. In section 3, the general results of section 2 are extended and applied to images (i.e. image functions) for yielding a simpler representation, once we do not demand the use of continuous functions for superposition. That section also describes the Sprecher construction for obtaining such a superposition. In section 4, the use of this representation is demonstrated. Section 5 gives a short conclusion of the presented work.

2 Representation of Functions

An important aspect in the design of soft computing methods is the *representation* of functions. This has to be distinguished from the *approximation* of functions. Representation stands for the exact *computability* of a function f by the configuration of a certain fixed computation scheme.

A typical problem field, where we meet the question of function representation is Genetic Programming. Given a set of node and terminal functions, from which functions are composed by tree structures, we are faced with the question about the total set of functions that can actually be composed from the given function set. Here, the compuation scheme is the tree structure. Similar questions are related to function representation by a (fuzzy) neural network, or the representation of control functions by fuzzy patches.

Here, we are considering the *computability in layers* of a given function as a general represenations scheme. With f standing for a continuous function of R_n into R, we define

Definition 1. *The function f is said to be computable-in-one-layer if one of the following conditions is fulfilled:*

- *f is a linear function $f(x_1, \ldots, x_n) = w_0 + w_1 x_1 + \ldots + w_n x_n$ with real values $w_0, w_1, \ldots, w_n$; or*
- *f is a function $f(x)$ of one argument.*

Definition 2. *The function f is said to be computable-in-m-layers ($m \geq 2$), if one of the following conditions is fulfilled:*

- *f can be written as a linear combination of the form $w_0 + w_1 f_1(x_1, \ldots, x_n) + \ldots + w_k f_k(x_1, \ldots, x_n)$ with continuous functions f_i that are computable-in-$(m-1)$-layers; or*
- *f can be represented in the form $g(h(x_1, \ldots, x_n))$, where g is a continuous function of one argument, and h is computable-in-$(m-1)$-layers.*

Given these definitions, two questions appear: the question about the *minimal* number of layers to compute any continuous function (if there is such a minimum at all), and the minimal number of layers to compute functions that can approximate any continuous function to any degree of precision.

The Stone-Weierstrass-Theorem gives an answer to the second question, giving $m = 3$. The proof for this was provided by Hornik [3].

Theorem 2. *Any continuous bounded function $f : [0,1]^n \to R$ can be approximated by a function that is computable-in-3-layers.*

In addition, we have (proof omitted here)

Theorem 3. *There exists a continuous bounded function $f : [0,1]^n \to R$ that can not be approximated by any function that is computable in 1 or 2 layers.*

Thus, we know that any function can at least be approximated by a function that is computable in 3 layers. As an application, this ensures that a perceptron with input, hidden and output layer suffices (given a suitable transition function, and assuming a bounded mapping from input to output) to approximate any function with any degree of precision.

The more general question about the *representation*, i.e. the first question, finds its answer in the Kolmogorov theorem, which ensures m to be not larger than 4. By showing that functions like $x_1 x_2 \cdot \exp((x_1^2 + x_2^2)/2)$ are not *computable-in-3-layers* (omitted here) we see that $m = 4$ is strict.

Theorem 4. *Any continuous bounded function $f : [0,1]^n \to R$ is computable-in-4-layers.*

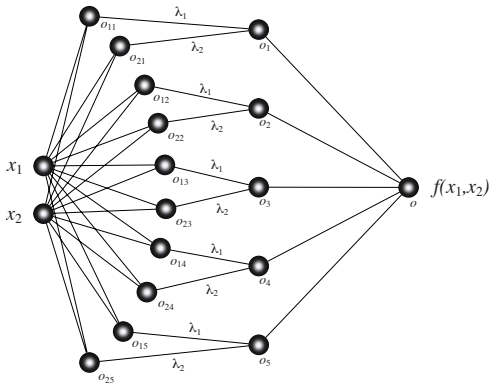

Fig. 1. Graphical representation of the representation of a function of two arguments by a KOLMOGOROV network. Thereby means $o_{pq} = \Phi_q(x_p)$ the "inner" function, $o_q = g_q\left(\sum_p \lambda_p o_{pq}\right)$ the "outer" function, and $o = \sum_q o_q$.

This theorem gave raise to the so-called KOLMOGOROV network, a neural net that is able to *represent* any function mapping $[0,1]^n$ into R (see fig. 1) and that was initially introduced by Hecht-Nielsen [2].

In the following, we consider these aspects of computability for image functions, where in practice we can relax from the requirement of having continuous functions for the representation.

3 Universal Representation of Image Functions

Usually in software development, images are represented by *buffers* in the working memory of the computer. This can be considered an unwinding of the image in a row-by-row manner, starting with the memory cell of the pixel $(0,0)$. This means, if the image is stored in a memory buffer starting at address S, then we find the grayvalue (assuming one byte per pixel) of pixel (i,j) at the address $S + j \cdot \lambda + i$ with λ standing for the *width* of the image. Formally, if $f(x,y)$ represents the image function, and $g[]$ the memory buffer:

$$f(i,j) = g[S + i + \lambda \cdot j]. \tag{3}$$

This equals the computation of an image function in two layers. With respect of the results summarized in the foregoing section, we may pose the question if this is really sufficient to have a representation of an image function (given that the KOLMOGOROV theorem states the need for at least four layers instead of two).

At first we may observe that this simple representation fixes the resolution of the image, thus it is naturally an approximation of the *true* image function.

Now we consider the following: Given a function $\phi : I^n \to I$ and the BANACH space S_ϕ of all functions $g[\phi(x)]$, with g an aarbitrary but not necessarily continuous function, mapping $I = [0, 1]$ into I. A function ϕ_1 is said to be *more efficient* than a function ϕ_2, if $S_{\phi_1} \supset S_{\phi_2}$. A function, for which S_ϕ is equal to the BANACH space S of all functions over I, is said to be *maximal efficient*. In our case, this would be a function that can provide the best approximation to S by means of a computation in two layers.

We see that an image function representation according to eq. 3 for a higher resolution is more efficient than for a lower resolution, but that there is no finite maximal efficient function. However, for fixed resolution, this representation $\phi(i, j) = i + \lambda \cdot j$ is maximal efficient in the space of all image functions of width λ. We find means like a finite lookup table to represent any bounded image function by *pixels*. However, we can easily find other maximal efficient representations, e.g. by any other total order of the image points (x, y) (note that such a different ordering of the image points can help to improve the weak representation of vertical pixel neighborhood in the image in the row-wise unwinding, e.g. by using finite approximates of PEANO space-filling curves).

Regarding general image functions, independent of their resolution, we see that the computation in two layers can not represent the image function, if we assume the representing functions to be continuously. However, we may revise the question about layer computability when we do not require continuous functions. This was initially considered by David Sprecher [9] and was answered by the following theorem:

Theorem 5. *For any integer $n \geq 2$ there exists real-valued, strictly monotone functions $h_p(x)$ with $1 \leq p \leq n$, which depend on n, having*
(i) The function

$$h(x) = \sum_{p=1}^{n} h_p(x_p)$$

separates all points in I^n:

$$\sum_{p=1}^{n} h_p(x_p) = \sum_{p=1}^{n} h_p(y_p)$$

if and only if $x_p = y_p$ is valid for all feasible values of p.
(ii) Any continuous function of n variables $f(x_1, \ldots, x_n)$ with domain I^n can be represented in the form

$$f(x_1, \ldots, x_n) = g\left[\sum_{p=1}^{n} h_p(x_p)\right] \tag{4}$$

with a (usually non-continuous) function g.

The proof for this is based on a construction. Here, we show this construction (called SPRECHER construction), the proof is more or less evident once having this construction.

By $[i_1, i_2, \ldots, i_k]_\gamma$ we denote the sequence of *digits* of a representation of a number $d_k \in I$ to the number base (radix) γ, with k positions after decimal point. Excluding sequences $[i_1, \ldots]$, having only digits $\gamma - 1$ from a certain position on, thus we can represent any real number in $[0, 1]$ in a unique manner.

Now, the representation of a number in I by the base n^n is composed into n representations of numbers in I to the base n. If

$$x = \sum_{r=1}^{\infty} \frac{i_r}{n^r} \tag{5}$$

then

$$h_p(x) = \sum_{r=1}^{\infty} \frac{i_r}{n^{nr-(p-1)}} = \sum_{r=1}^{\infty} \frac{i_r \cdot n^{p-1}}{(n^n)^r}. \tag{6}$$

Any digit i_r from $\{0, \ldots, n^n - 1\}$ of the representation of a number $x \in I$ to the base n^n has a representation to the base n with maximum n digits:

$$i_r = \sum_{s=0}^{n-1} i_{rs} n^s, \tag{7}$$

where for any i_{rs} $0 \leq i_{rs} < n$ holds. Thus, we also have

$$[i_1, \ldots, i_k]_{n^n} = h([i_{11}, \ldots, i_{k1}]_n, \ldots, [i_{1n}, \ldots, i_{kn}]_n) \tag{8}$$

and this is exactly a bijection between I^n and I.

Example: With $n = 2$ we want to find the image of the point with decimal ($\gamma = 10$) representation $(0.625, 0.75)$. Using base 2, this point has the representation $(0.101_2, 0.11_2)$ and it follows

$$h_1(0.101_2) = \frac{1}{4} + \frac{1}{4^3}$$
$$h_2(0.11_2) = \frac{2}{4} + \frac{2}{4^2}.$$

Thus

$$h[(0.625, 0.75)] = \frac{3}{4} + \frac{2}{16} + \frac{1}{64} = 0.890625.$$

Now we want to find the point onto which $0.78125 = 0.302_4$ is mapped. From $3_4 = 11_2$, $0_4 = 00_2$ and $2_4 = 10_2$ by construction of two sequences of binary digits from the first or second element of this representation

$$h^{-1}(0.78125) = (0.101_2, 0.100_2) = (0.625, 0.5).$$

Given any f, the construction of g is simple now: replace any $y \in I$ by $g(y) = f[h^{-1}(y)]$.

4 Application

The SPRECHER construction, as presented in the foregoing section, offers several applications for representing functions in soft computing methods. We may consider compression as an application here. Figure 2 shows an interesting property. If we compute $g(y)$ for an image, delete a number of entries (means all values in an interval are replaced by a single value) and perform the reverse transformation, we achieve a compressed representation of the image. Being not really competitive to state-of-the-art compression methods nowadays, we find this a simple approach with excellent performance up to compression ratios $1:3$ and acceptable until $1:10$. This proves the better representation ability of the SPRECHER constructions for local neighborhoods in the image.

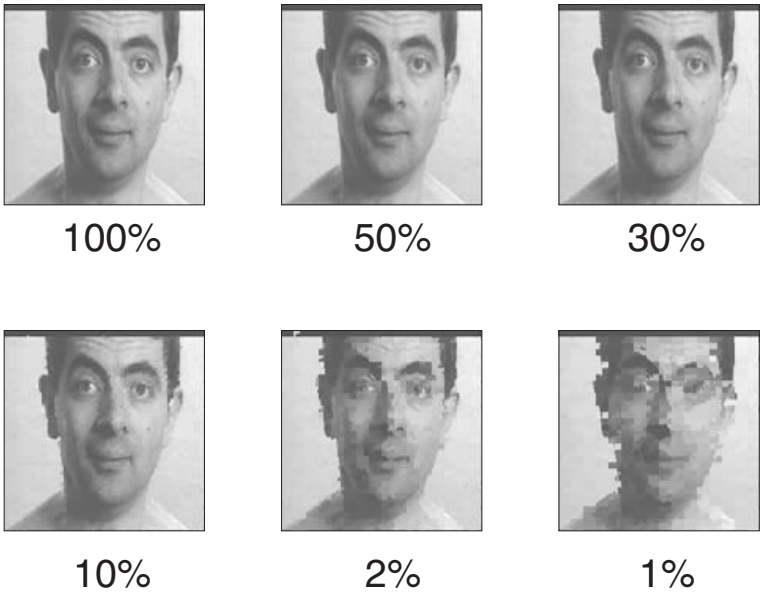

Fig. 2. Compression quality of the SPRECHER construction by deleting some points $g(y)$ and reconstruction from these reduced data sets. Upper left is the original image.

The approach can be extended to the efficient representation of pixel neighborhoods as well. By considering one pixel along with its four direct neighbors, we can map the grayvalues assigned to these five pixel positions (after normalization into the range $[0,1]$) to a real number $y \in I$. Using a higher bit-depth than 8bit (having 16bit almost technically feasible today) thus we can represent the same image by a pointwise representation of local neighborhoods (we may propose to call this kind of picture elements as *nexel*).

One advantage of this representation, which comes out naturally when considering the SPRECHER construction is the better representation of image sensor activations. We may note that an arbitrary image of y values may not be mapped back to an image due to the inconsistency in the local neighborhood. Each neighbor positions is represented by four other points in the mapped image, and the reverse mapping may give different values here.

However, this pictures matches better with the technical procedure to obtain an image from a sensing device: within a CCD chip, there is no such real consistency in the neighborhood of a sensitive cell. Each cell is partially activated from the neighboring cells during acquisition as well. Camera effects like *blurring* are well-known artefacts of this technical procedure. We have a rather unsymmetrical influence of neighboring cells onto each cell, and thus the representation of cell neighborhoods instead of cells only has the potential to give a better representation of the acquired image signal. By the reverse SPRECHER construction, the usual optical image can be constructed easily, but other procedures may take advantage of the more complete neighborhood representation as well.

In the same fashion, image operators can be represented by such a mapping. As an easy example, greyscale morphology, taking the maximum greyvalue in the neighborhood of a pixel as result of the transform at this pixels positions, can be directly taken from this representation of each pixel neighborhood.

5 Conclusion

In this paper, we have studied the representation of images in a formal manner. The computability of layers, often employed already in soft computing application, provides a general means for such a representation. From general results about representation and approximation of functions, we learned that any function can be approximated by functions that can be computed in three layer to any degree of precision, and can be exactly represented by functions that can be computed in four layers. Leaving out the requirement for continuous representation, there is even a two-layer computation of any continuous function. The SPRECHER constructions is one means to obtain such a representation. It was demonstrated that on compressed versions local neighborhoods of an image are represented better than in the default buffer-wise manner used today. One other prospect here is the ability of representing pixel neighborhoods in the image directly, which might support a better representation of the technical acquisition procedure in camera devices. Also, image operators can be represented this way in a formal manner, for e.g. employing search spaces of image operators.

The arguments given in this study will be followed by considerations of this representation in image operation adaptation tasks.

References

1. Neil E. Cotter and Thierry J. Guillerm. The CMAC and a theorem of Kolmogorov. *Neural Networks*, 5:221–228, 1992.
2. Robert Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of the First International Conference on Neural Networks*, volume III, pages 11–13. IEEE Press, New York, 1987.
3. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
4. Andrej Nikolajewitsch Kolmogorov. On the representation of continuous functions of several variables by superpositions of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, 114:679–681, 1957. (in Russian).
5. Mario Köppen. On the training of a kolmogorov network. In José R. Dorronsoro, editor, *Artificial Neural Networks - ICANN 2002*, LNCS 2415, pages 474–479, Madrid, Spain, August 2002, 2002. Springer-Verlag Heidelberg.
6. V. Kreinovich, H.T. Nguyen, and D.A. Sprecher. Normal forms for fuzzy logic — an application of kolmogorov's theorem. Technical Report UTEP-CS-96-8, University of Texas at El Paso, Januar 1996.
7. H.T. Nguyen and V. Kreinovich. Kolmogorov's theorem and its impact on soft computing. In Ronald R. Yager and Janusz Kacprzyk, editors, *The Ordered Weighted Averaging operators. Theory and Applications.*, pages 3–17. Kluwer Academic Publishers, 1997.
8. D.A. Sprecher. On the structure of continuous functions of several variables. *Transcations of the American Mathematical Society*, 115(3):340–355, 1965.
9. D.A. Sprecher. A representation theorem for continuous functions of several variables. *Proceedings of the American Mathematical Society*, 16:200–203, 1965.
10. D.A. Sprecher. A numerical implementation of Kolmogorov's superpositions I. *Neural Networks*, 9(5):765–772, 1996.
11. D.A. Sprecher. A numerical implementation of Kolmogorov's superpositions II. *Neural Networks*, 10(3):447–457, 1997.