Kolmogorov Superposition Theorem and its application to multivariate function decompositions and image representation

Pierre-Emmanuel Leni, Yohan Fougerolle, Frederic Truchetet

Kolmogorov Superposition Theorem and its application to multivariate function decompositions and image representation

Pierre-Emmanuel Leni
pierre-emmanuel.leni@u-bourgogne.fr

Yohan D. Fougerolle
yohan.fougerolle@u-bourgogne.fr

Frédéric Truchetet
frederic.truchetet@u-bourgogne.fr
Université de Bourgogne
Laboratoire LE2I, UMR CNRS 5158
12 rue de la fonderie, 71200 Le Creusot, France

Abstract

In this paper, we present the problem of multivariate function decompositions into sums and compositions of monovariate functions. We recall that such a decomposition exists in the Kolmogorov's superposition theorem, and we present two of the most recent constructive algorithms of these monovariate functions. We first present the algorithm proposed by Sprecher, then the algorithm proposed by Igelnik, and we present several results of decomposition for gray level images. Our goal is to adapt and apply the superposition theorem to image processing, i.e. to decompose an image into simpler functions using Kolmogorov superpositions. We synthetise our observations, before presenting several research perspectives.

1. Introduction

In 1900, Hilbert stated that high order equations cannot be solved by sums and compositions of bivariate functions. In 1957, Kolmogorov proved this hypothesis wrong and presented his superposition theorem (KST), that let us write every multivariate functions as sums and compositions of monovariate functions. Nevertheless, Kolmogorov did not propose any method for monovariate function construction and only proved their existence. The goal of this work is to express multivariate functions using simpler elements, i.e. monovariate functions, that can be easily processed using 1D or 2D signals processing methods instead of searching for complex multidimensional extension of traditional methods. To summarize, we have to adapt Kolmogorov superposition theorem to determine a decomposition of multivariate functions into monovariate functions, with a definition that allows post-processing. Recent contributions about KST are one hidden layer neural network identification (see figure 1), and construction methods for monovariate functions. Amongst KST-dedicated works, Sprecher has proposed an algorithm for exact monovariate function reconstruction in [7] and [8]. Sprecher explicitly describes construction methods for monovariate functions, introduces interesting notions for theorem comprehension (such as tilage), which allows direct implementation, whereas Igelnik's approach offers several modification perspectives about the monovariate function construction.

In the second section, we introduce the superposition theorem and several notations. Section 3 is dedicated to Sprecher's algorithm, and section 4 to Igelnik's algorithm. In section 5, we compare Igelnik's and Sprecher's algorithms. In the last section, we draw conclusions, consider several research perspectives, and potential applications.

Our contributions are the synthetic explanation of Sprecher's and Igelnik's algorithms, and the application of the superposition theorem to gray level images, using both Sprecher's and Igelnik's algorithm with bivariate functions.

2. Presentation of Kolmogorov theorem

The superposition theorem, reformulated and simplified by Sprecher in [6] is written:

Theorem 1 (Kolmogorov superposition theorem) *Every continuous function defined on the identity hypercube ($[0,1]^d$ noted I^d) $f : I^d \longrightarrow \mathbb{R}$ can be written as sums and*

compositions of continuous monovariate functions:

$$\begin{cases} f(x_1, ..., x_d) = \sum_{n=0}^{2d} g_n\big(\xi(x_1 + na, ..., x_d + na)\big) \\ \\ \xi(x_1 + na, ..., x_d + na) = \sum_{i=1}^{d} \lambda_i \psi(x_i + an), \end{cases} \tag{1}$$

with ψ continuous function, λ_i and a constants. ψ is called inner function and $g(\xi)$ external function.

The inner function ψ associates every component x_i from the real vector $(x_1, ..., x_d)$ of I^d to a value in $[0, 1]$. The function ξ associates each vector $(x_1, ..., x_d) \in I^d$ to a number y_n from the interval $[0, 1]$. These numbers y_n are the arguments of functions g_n, that are summed to obtain the function f.

According to the theorem, the multivariate function decomposition can be divided into two steps: the construction of a hash function (the inner function) that associates the components $x_i, i \in [\![1, d]\!]$ of each dimension to a unique number; and the construction of an external function g with the values corresponding to f for these coordinates. Figure 3 illustrates the hash function ξ defined by Sprecher in 2D. Both Sprecher's and Igelnik's algorithms define a superposition of disjoint hypercube translated tilages, splitting the definition space of function f. Then, monovariate functions are generated for each tilage layer. Figure 4 illustrates the superposition of tilage layers in 2D.

3. Sprecher's Algorithm

Sprecher has proposed an algorithm to determine the internal and external functions in [7] and [8], respectively. Because, the function ψ, defined by Sprecher to determine ξ is discontinuous for several input values, we use the definition proposed by Braun and *al.* in [2], that provides continuity and monotonicity for the function ψ. The other parts of Sprecher's algorithm remain unchanged.

Definition 1 ***(Notations)***.

- *d is the dimension, $d \geqslant 2$.*
- *m is the number of tilage layers, $m \geqslant 2d$.*
- *γ is the base of the variables x_i, $\gamma \geqslant m + 2$.*
- *$a = \frac{1}{\gamma(\gamma-1)}$ is the translation between two layers of tilage.*
- *$\lambda_1 = 1$ and for $2 \leqslant i \leqslant d$, $\lambda_i = \sum_{r=1}^{\infty} \frac{1}{\gamma^{(i-1)(d^r-1)/(d-1)}}$ are the coefficients of the linear combination, that is the argument of function g.*

Sprecher proposes a construction method for functions ξ and function ψ. More precisely, the definition of function

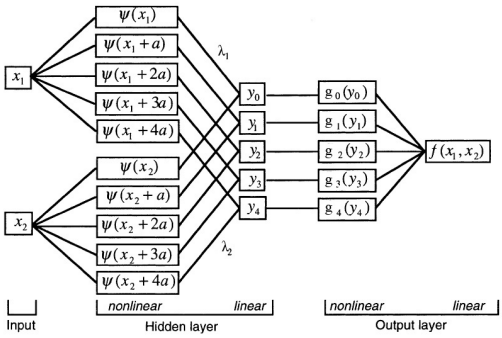

Figure 1. Illustration of the analogy between the KST and a one hidden layer neural network, from [8].

ψ and the structure of the algorithm are based on the decomposition of real numbers in the base γ: every decimal number (noted d_k) in $[0, 1]$ with k decimals can be written:

$$\begin{aligned} d_k &= \sum_{r=1}^{k} i_r \gamma^{-r}, \\ and\ d_k^n &= d_k + n \sum_{r=2}^{k} \gamma^{-r} \end{aligned} \tag{2}$$

defines a translated d_k.

Using the d_k defined in equation 2, Braun and *al.* define the function ψ by:

$$\psi_k(d_k) = \begin{cases} for\ k = 1: \\ d_k \\ for\ k > 1\ and\ i_k < \gamma - 1: \\ \psi_{k-1}(d_k - \frac{i_k}{\gamma^k}) + \frac{i_k}{\gamma^{\frac{d^k-1}{d-1}}} \\ for\ k > 1\ and\ i_k = \gamma - 1: \\ \frac{1}{2}(\psi_k(d_k - \frac{1}{\gamma^k}) + \psi_{k-1}(d_k + \frac{1}{\gamma^k})). \end{cases} \tag{3}$$

Figure 2 represents the plot of function ψ on the interval $[0, 1]$. The function ξ is obtained through linear combination of the real numbers λ_i and function ψ applied to each component x_i of the input value. Figure 3 represents the function ξ on the space $[0, 1]^2$.

Sprecher has demonstrated that the image of disjoint intervals I are disjoint intervals $\psi(I)$. This separation property generates intervals that constitute an incomplete tilage of $[0, 1]$. This tilage is extended to a d-dimensional tilage by making the cartesian product of the intervals I. In order to cover the entire space, the tilage is translated several times by a constant a, which produces the different layers of the final tilage. Thus, we obtain $2d + 1$ layers: the original tilage constituted of the disjoint hypercubes having disjoint images through ψ, and $2d$ layers translated by a along

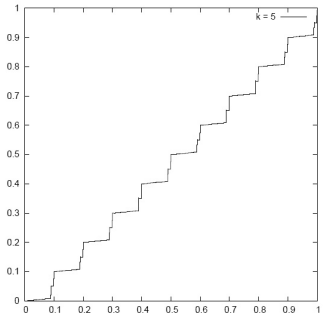

Figure 2. Plot of function ψ for $\gamma = 10$, from [2].

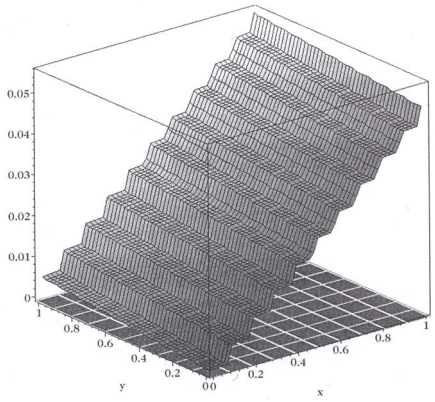

Figure 3. Plot of the hash function ξ for $d = 2$ and $\gamma = 10$, from [1].

each dimension. Figure 4 represents a tilage section of a 2D space: $2d + 1 = 5$ different superposed tilages can be seen, displaced by a.

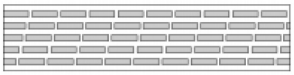

Figure 4. Section of the tilage for a 2D space and a base $\gamma = 10$ (5 different layers). From [1].

For a 2D space, a hypercube is associated with a couple $\mathbf{d_{k_r}} = (d_{k_r 1}, d_{k_r 2})$. The hypercube $S_{k_r}(\mathbf{d_{k_r}})$ is associated with an interval $T_{k_r}(\mathbf{d_{k_r}})$ by the function ξ. The image of a hypercube S is an interval T by function ξ, see figure 5.

Internal functions ψ and ξ have been determined. External functions g_n cannot be directly evaluated. Sprecher builds r functions g_n^r, such that their sum converges to the

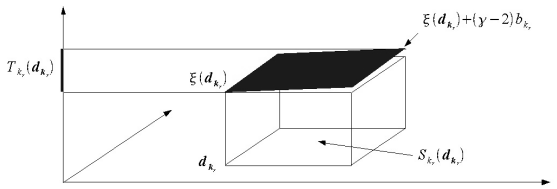

Figure 5. Function ξ associates each paving block with an interval T_k in $[0, 1]$.

external function g_n. The algorithm iteratively evaluates an external function g_n^r, in three steps. At each step r, the precision, noted k_r, must be determined. The decomposition of real numbers d_k can be reduced to only k_r digits (see equation 2). Function f_r defines the approximation error, that tends to 0 when r increases. The algorithm is initialized with $f_0 = f$ and $r = 1$.

3.1. first step: determination of the precision k_r and tilage construction

For two coordinates x_i and x_i' that belong to two sets, referencing the same dimension i and located at a given distance, the distance between the two sets $\mathbf{x}$ and $\mathbf{x}'$ obtained with f must be smaller than the N^{th} of the oscillation of f, *i.e.*:

$$if\,|x_i - x_i'| \leqslant \tfrac{1}{\gamma^{k_r}},$$
$$\left|f_{r-1}(x_1, ..., x_d) - f_{r-1}(x_1', ..., x_d')\right| \leqslant \epsilon \left\|f_{r-1}\right\|.$$

Once k_r has been determined, the tilage $d^n_{k_r 1}, ..., d^n_{k_r d}$ is calculated by:

$$\forall i \in [\![1, d]\!], d^n_{k_r i} = d_{k_r i} + n \sum_{r=2}^{k_r} \frac{1}{\gamma^r}.$$

3.2. second step: internal functions ψ and ξ

For n from 0 to m, determine $\psi(d^n_{k_r})$ and $\xi(d^n_{k_r 1}, ..., d^n_{k_r d})$ using equations 2 and 3.

3.3. third step: determination of the approximation error

$\forall n \in [\![0, m]\!]$, evaluate:

$$g_n^r \circ \xi(x_1 + an, ..., x_d + an) =$$
$$\tfrac{1}{m+1} \textstyle\sum_{d^n_{k_r 1}, ..., d^n_{k_r d}}$$
$$f_{r-1}\big(d_{k_r 1}, ..., d_{k_r d}\big)\theta_{d^n_{k_r}}\big(\xi(x_1 + an, ..., x_d + an)\big),$$

where θ is defined in equation 4. Then, evaluate:

$$f_r(x_1, ..., x_d) = f(x_1, ..., x_d)$$
$$- \textstyle\sum_{n=0}^{m} \sum_{j=1}^{r} g_n^j \circ \xi(x_1 + an, ..., x_d + an).$$

At the end of the r^{th} step, the result of the approximation of f is given by the sum of the $m+1$ layers of r previously determined functions g_n^r:

$$f \approx \sum_{n=0}^{m} \sum_{j=1}^{r} g_n^j \circ \xi(x_1 + an, ..., x_d + an).$$

Definition 2 *The function θ is defined as:*

$$\begin{aligned} \theta_{d_k^n}(y_n) = \sigma\left(\gamma^{\frac{d^{k+1}-1}{d-1}}\left(y_n - \xi(d_k^n)\right) + 1\right) \\ -\sigma\left(\gamma^{\frac{d^{k+1}-1}{d-1}}\left(y_n - \xi(d_k^n) - (\gamma-2)b_k\right)\right), \end{aligned} \quad (4)$$

where σ is a continuous function such that:

$$\begin{cases} \sigma(x) \equiv 0, pour\ x \leqslant 0 \\ \sigma(x) \equiv 1, pour\ x \geqslant 1 \end{cases},$$

and:

$$b_k = \sum_{r=k+1}^{\infty} \frac{1}{\gamma^{\frac{d^r-1}{d-1}}} \sum_{p=1}^{n} \lambda_p.$$

A unique internal function is generated for every function f. Only the external functions are adapted to each approximation. At each iteration, a tilage is constructed. The oscillation of the error function f_r tends to 0, consequently, the more iterations, the better is the approximation of function f by functions g.

3.4. Results

We present the results of the decomposition applied to gray levels images, that can be seen as bivariate functions $f(x, y) = I(x, y)$. Figure 6 represents two layers of the approximation obtained after one and two iterations. The sum of each layer gives the approximation of the original image. White points on the images 6(b) and 6(e) correspond to negative values of external functions. Figure 7 shows two reconstructions of the same original image after one and two iterations of Sprecher's algorithm.

The layers obtained with the decomposition are very similar, which is coherent since each layer corresponds to a fraction of a sample of the function f, slightly translated by the value a. For 2D functions, we observe that the reconstruction quickly converges to the original image: few differences can be seen between the first and the second approximations on figures 7(b) and 7(c).

Figure 6. (a) and (b) The first layer ($n = 0$) after one and two iterations ($r = 1, r = 2$) respectively. (c) Sum of (a) and (b), partial reconstruction given by the first layer. (d) and (e) The last layer ($n = 5$) after one and two iterations ($r = 1, r = 2$) respectively. (f) Sum of (d) and (e), partial reconstruction given by the last layer.

Figure 7. (a) Original image. (b) and (c) Reconstruction after one and two iterations, respectively.

4. Igelnik's Algorithm

The fixed number of layers m and the lack of flexibility of inner functions are two major issues of Sprecher's approach. Igelnik and Parikh kept the idea of a tilage, but the number of layers becomes variable. Equation 1 is replaced by:

$$f(x_1, ..., x_d) \simeq \sum_{n=1}^{N} a_n g_n\left(\sum_{i=1}^{d} \lambda_i \psi_{ni}(x_i)\right) \quad (5)$$

This approach present major differences with Sprecher's algorithm:

- ψ_{ni} has two indexes i and n: inner functions ψ_{ni}, independent from function f, are randomly generated for each dimension i and each layer n.

- the functions ψ and g are sampled with M points, that are interpolated by cubic splines.

- the sum of external functions g_n is weighted by coefficients a_n.

A tilage is created, made of hypercubes C_n obtained by cartesian product of the intervals $I_n(j)$, defined as follows:

Definition 3

$$\forall n \in [\![1, N]\!], j \geqslant -1, I_n(j) = [(n-1)\delta + (N+1)j\delta, (n-1)\delta + (N+1)j\delta + N\delta],$$

where δ is the distance between two intervals I of length $N\delta$, such that the function f oscillation is smaller than $\frac{1}{N}$ on each hypercube C. Values of j are defined such that the previously generated intervals $I_n(j)$ intersect the interval $[0, 1]$.

Figure 8 illustrates the construction of intervals I. The tilage is defined once for all at the beginning of the algorithm. For a given layer n, d inner functions ψ_{ni} are generated (one per dimension). The argument of function g_n is a convex combination of constants λ_i and functions ψ_{ni}. The real numbers λ_i, randomly chosen, must be linearly independent, strictly positive and $\sum_{i=1}^{d} \lambda_i \leqslant 1$. Finally, external functions g_n are constructed, which achieves layer creation.

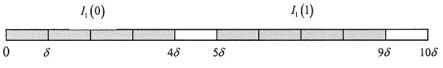

Figure 8. From [5], intervals $I_1(0)$ and $I_1(1)$ for $N = 4$.

4.1. Inner functions construction

Each function ψ_{ni} is defined as follows:

- Generate a set of j distinct numbers y_{nij}, between Δ and $1-\Delta$, $0 < \Delta < 1$, such that the oscillations of the interpolating cubic spline of ψ values on the interval δ is lower than Δ. j is given by definition 3. The real numbers y_{nij} are sorted, *i.e.*: $y_{nij} < y_{nij+1}$. The image of the interval $I_n(j)$ by function ψ is y_{nij}.

- Between two intervals $I_n(j)$ and $I_n(j+1)$, we define a nine degree spline s on an interval of length δ, noted $[0, \delta]$. Spline s is defined by: $s(0) = y_{nij}$, $s(\delta) = y_{nij+1}$, and $s'(t) = s^{(2)}(t) = s^{(3)}(t) = s^{(4)}(t) = 0$ for $t = 0$ and $t = \delta$.

Figure 9 gives a construction example of function ψ for two consecutive intervals $I_n(j)$ and $I_n(j+1)$. Function $\xi_n(x) = \sum_{i=1}^{d} \lambda_i \psi_{ni}(x)$ can be evaluated. On hypercubes $C_{nij_1,\ldots,j_d}$, function ξ has constant values $p_{nj_1,\ldots,j_d} = \sum_{i=1}^{d} \lambda_i y_{nij_i}$. Every random number y_{nij_i} is selected providing that the generated values p_{nij_i} are all different, $\forall i \in [\![1, d]\!], \forall n \in [\![1, N]\!], \forall j \in \mathbb{N}, j \geqslant -1$.

To adjust the inner function, Igelnik use a stochastic approach using neural networks. Inner functions are sampled by M points, that are interpolated by a cubic spline. We can consider two sets of points: points located on plateaus over intervals $I_n(j)$, and points located between two intervals $I_n(j)$ and $I_n(j+1)$, placed on a nine degree spline. These points are randomly placed and optimized during the neural network construction.

4.2. External function constructions

Function g_n is defined as follows:

- For every real number $t = p_{n,j_1,\ldots,j_d}$, function $g_n(t)$ is equal to the N^{th} of values of the function f in the center of the hypercube $C_{nij_1,\ldots,j_d}$, noted $b_{n,j_1,\ldots,j_d}$, *i.e.*: $g_n(p_{n,j_1,\ldots,j_d}) = \frac{1}{N} b_{n,j_1,\ldots,j_d}$.

- The definition interval of function g_n is extended to all $t \in [0, 1]$. Consider $A(t_A, g_n(t_A))$ and $D(t_D, g_n(t_D))$ two adjacent points, where t_A and t_D are two levels $p_{n,j_1,\ldots,j_d}$. Two points B et C are placed in A and D neighborhood, respectively. Points B and C are connected with a line defined with a slope $r = \frac{g_n(t_C) - g_n(t_B)}{t_C - t_B}$. Points $A(t_A, g_n(t_A))$ and $B(t_B, g_n(t_B))$ are connected with a nine degree spline s, such that: $s(t_A) = g_n(t_A)$, $s(t_B) = g_n(t_B)$, $s'(t_B) = r$, $s^{(2)}(t_B) = s^{(3)}(t_B) = s^{(4)}(t_B) = 0$. Points C and D are connected with a similar nine degree spline. The connection condition at points A and D of both nine degree splines gives the remaining conditions. Figure 10 illustrates this construction.

Remark 1 *Points A and D (function f values in the centers of the hypercubes) are not regularly spaced on the interval $[0, 1]$, since their abscissas are given by function ξ, and depend on random values $y_{nij} \in [0, 1]$. The placement of points B and C in the circles centered in A and D must preserve the order of points: A, B, C, D, i.e. the radius of these circles must be smaller than half of the length between the two points A and D.*

To determine the weights a_n and to choose the points in function ψ, Igelnik creates a neural network using a stochastic method (ensemble approach). N layers are successively built. To add a new layer, K candidate layers are generated. Every candidate layer is added to the existing neural

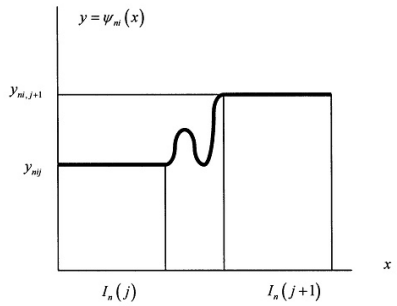

Figure 9. From [5], plot of ψ. On the intervals $I_n(j)$ and $I_n(j+1)$, ψ has constant values, respectively y_{nij} and y_{nij+1}. A nine degree spline connects two plateaus.

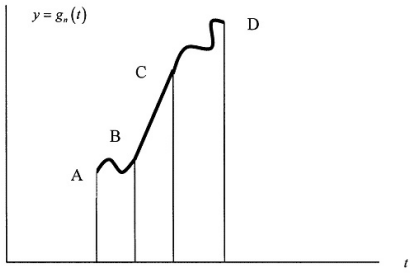

Figure 10. From [5], plot of g_n. Points A and D are obtained with function ξ and function f.

network to obtain K candidate neural networks. We keep the layer from the network with the smallest mean squared error. The set of K candidate layers have the same plateaus y_{nij}. The differences between two candidate layers are the set of sampling points located between two intervals $I_n(j)$ and $I_n(j+1)$, that are randomly chosen, and the placement of points B and C.

4.3. Neural network stochastic construction

The N layers are weighted by real numbers a_n and summed to obtain the final approximation of the function f. Three sets of points are constituted: a training set D_T, a generalization set D_G and a validation set D_V. For a given layer n, using the training set, a set of points constituted by the approximation of the neural network (composed of $n-1$ layers already selected) and the current layer (one of the candidate) is generated. The result is written in a matrix, Q_n, constituted of column vectors $q_k, k \in [\![0, n]\!]$ that corresponds to the approximation $(\tilde{f})$ of the k^{th} layer for points set $\big((x_{1,1}, ..., x_{d,1}), ..., (x_{1,P}, ..., x_{d,P})\big)$ of D_T:

$$Q_n = [q_0, q_1, ..., q_n],\ with\ \forall k \in [0, ..., n],$$
$$q_k = \begin{bmatrix} \tilde{f}_k(x_{1,1}, ...x_{d,1}) \\ ... \\ \tilde{f}_k(x_{1,P}, ...x_{d,P}) \end{bmatrix}.$$

Remark 2 *$N+1$ layers are generated for the neural network, whereas only N layers appear in equation 5: the first layer (corresponding to column vector q_0) is a initialization constant layer.*

To determine coefficients a_n, the gap between f and its approximation $\tilde{f}$ must be minimized:

$$\|Q_n a_n - t\|,\ noting\ t = \begin{bmatrix} f(x_{1,1}, ..., x_{d,1}) \\ ... \\ f(x_{1,P}, ..., x_{d,P}) \end{bmatrix}. \quad (6)$$

The solution is given by $Q_n^{-1}t$. An evaluation of the solution is proposed by Igelnik in [4]. The result is a column vector $(a_0, ..., a_n)^T$: we obtain a coefficient a_l for each layer l, $l \in [\![0, n]\!]$. To choose the best layer amongst the K candidates, the generalization set D_G is used. Matrix Q'_n is generated as matrix Q_n, using the generalization set instead of the training set. Equation 6 is solved, replacing matrix Q_n with Q'_n, and using coefficients a_n obtained with matrix Q_n inversion. The neural network associated mean squared error is determined to choose the network to select.

The algorithm is iterated until N layers are constructed. The validation error of the final neural network is determined using validation set D_V.

4.4. Results

We have applied Igelnik's algorithm to gray level images. Figure 11 represents two layers: first ($N = 1$) and last ($N = 10$) layer. The ten layers are summed to obtain the final approximation. Figure 12 represents the approximation obtained with Igelnik's algorithm with $N = 10$ layers. Two cases are considered: the sum using optimized weights a_n and the sum of every layer (without weight). The validation error is 684.273 with optimized weights a_n and 192.692 for $a_n = 1, n \in [\![1, N]\!]$. These first results show that every layer should have the same weight.

5. Discussion and comparison

Sprecher and *al.* have demonstrated in [9] that we obtain a space-filling curve by inverting function ξ. Figure 13 represents the scanning curve obtained with the function ξ inversion, that connects real couples. For a 2D space, each couple $(d_{k_r 1}, d_{k_r 2})$ ($k_r = 2$ in figure 13) is associated to a

Figure 11. Two decomposition layers: (a) First layer. (b) Last layer.

real value in $[0, 1]$, that are sorted and then connected by the space filling curve.

We can generate space filling curves for Igelnik internal functions as well. We obtain a different curve for each layer (a new function ξ is defined for each layer). Moreover, each function ξ has constant values over tilage blocks, which introduces indetermination in space filling curves: different neighbor couples (d_{k_r1}, d_{k_r2}) have the same image by function ξ. Figure 14 and figure 15 are two different views of a space filling curve defined by a function ξ of Igelnik's algorithm. Neighbor points connections can be seen: horizontal squares correspond to a function ξ plateau image by function ξ.

Sprecher's algorithm generates an exact decomposition. The function constructions are explicitly defined, which simplifies implementation. Unfortunately, external monovariate function constructions are related to internal function definitions and *vice versa*, which implies that monovariate function modifications require large algorithm redefinitions. Igelnik's algorithm, on the other hand, generates larger approximation error, but internal and external function definitions are independent.

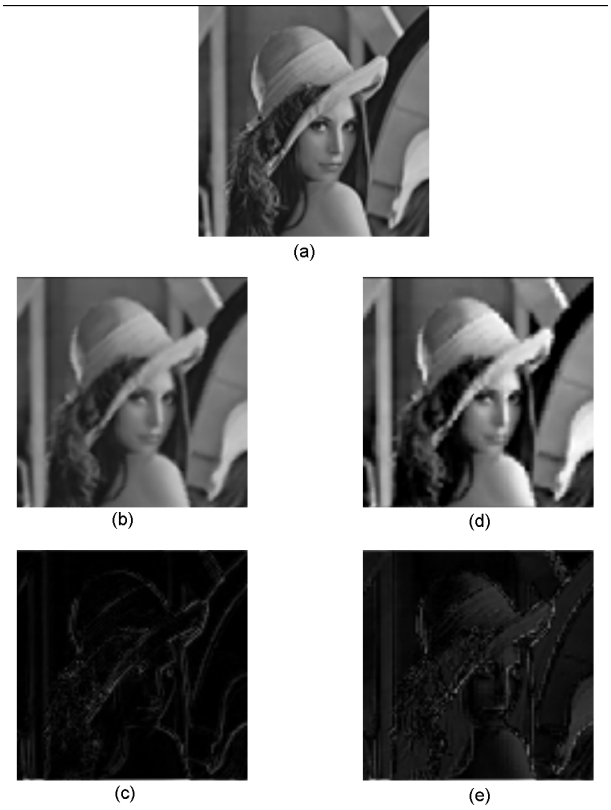

Figure 12. (a) Original image. (b) Igelnik's approximation for $N = 10$ and identical weight for every layer. (c) Absolute difference between Igelnik's approximation (b) and original image. (d) Igelnik's approximation for $N = 10$ and optimized weights a_n. (e) Absolute difference between Igelnik's approximation (d) and original image

6. Conclusion and perspectives

We have dealt with multivariate function decomposition using Kolmogorov superposition theorem. We have presented Sprecher's algorithm that creates internal and external functions, following theorem statement: for every function f to decompose, an inner function is used, and several external functions are created. Then, we have applied the algorithm to bivariate functions, illustrated on gray level images. The results show that the algorithm rapidly converges to the original image. Finally, we have presented Igelnik's algorithm and its application to gray level images.

This preliminary work shows several interesting properties of the decomposition. An image can be converted into a 1D signal: with bijectivity of function ψ, every pixel of the image is associated with a value in $[0, 1]$. Several questions remain open: How can space-filling curves be controlled? Can we generate different curves for some specific areas of the image? The main drawback of Sprecher's algorithm is its lack of flexibility: inner functions ψ cannot be modified; space-filling curve, *i.e.* images scanning, is always the same function. Igelnik's algorithm constructs several inner functions (one per layer), that are not strictly increasing: we obtain several space-filling curves, at the cost of indeterminations for particular areas (on tilage blocks). Due to Igelnik's algorithm construction, we can adapt inner function definition. Few scanning functions could be tested (scanning of areas with the same gray levels for example), and resulting external functions and approximations observed. Several questions remain open: which part (internal or external) does contain most of information? Can we describe images with only inner or external functions? Optimal func-

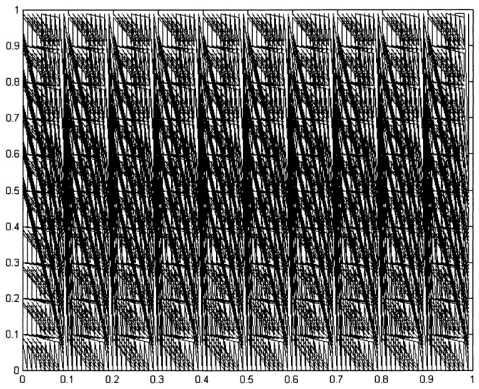

Figure 13. Sprecher's space filling curve.

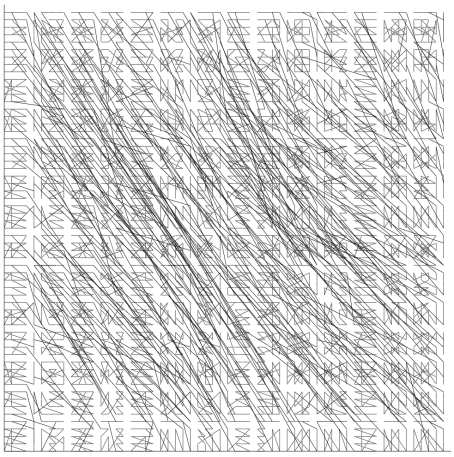

Figure 14. Igelnik's space filling curve, top view.

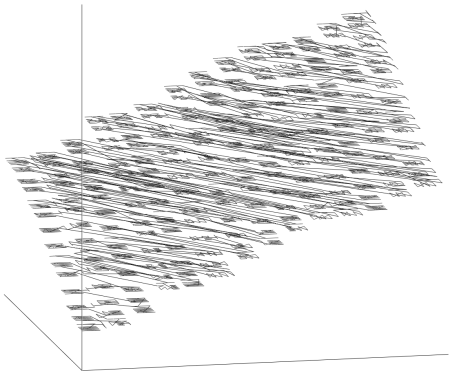

Figure 15. Igelnik's space filling curve, 3D view.

tion constructions induce data compression. An other possible application is watermarking. A mark could be included using several methods: information contained into a layer could be modified to define a mark, or define a specific space-filling curve.

References

[1] V. Brattka. Du 13-ième problème de Hilbert à la théorie des réseaux de neurones : aspects constructifs du théorème de superposition de Kolmogorov. *L'héritage de Kolmogorov en mathématiques. Éditions Belin, Paris.*, pages 241–268, 2004.

[2] J. Braun and M. Griebel. On a constructive proof of Kolmogorov's superposition theorem. *Constructive approximation*, 2007.

[3] R. Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. *Proceedings of the IEEE International Conference on Neural Networks III, New York*, pages 11–13, 1987.

[4] B. Igelnik, Y.-H. Pao, S. R. LeClair, and C. Y. Shen. The ensemble approach to neural-network learning and generalization. *IEEE Transactions on Neural Networks*, 10:19–30, 1999.

[5] B. Igelnik and N. Parikh. Kolmogorov's spline network. *IEEE transactions on neural networks*, 14(4):725–733, 2003.

[6] D. A. Sprecher. An improvement in the superposition theorem of Kolmogorov. *Journal of Mathematical Analysis and Applications*, (38):208–213, 1972.

[7] D. A. Sprecher. A numerical implementation of Kolmogorov's superpositions. *Neural Networks*, 9(5):765–772, 1996.

[8] D. A. Sprecher. A numerical implementation of Kolmogorov's superpositions II. *Neural Networks*, 10(3):447–457, 1997.

[9] D. A. Sprecher and S. Draghici. Space-filling curves and Kolmogorov superposition-based neural networks. *Neural Networks*, 15(1):57–67, 2002.