Abstract

This paper presents a family of layered feed-forward networks that is able to uniformly approximate functions on any metric space, and also on a wide variety of non-metric spaces. Non-Euclidean input spaces are frequently encountered in practice, while usual approximation schemes are guaranteed to work only on Euclidean metric spaces. Theoretical foundations are provided, as well as practical algorithms and illustrative examples. This tool potentially constitutes a significant extension of the common notion of 'universal approximation capability'.

1. Introduction

Neural computation research, together with related areas in approximation theory, have developed powerful methods for approximating continuous mappings on compact subsets of \( \mathbb{R}^n \) as a Euclidean space. Most approximation schemes use three layered feed-forward neural architectures with scalar product based neurons (Cybenko, 1989; Funahashi, 1989; Hornik, 1993; Leshno, Lin, Pinkus, & Schocken, 1993; Rumelhart, Hinton, & Williams, 1986a,b), or Euclidean distance based neurons (Girosi & Poggio, 1990; Micchelli, 1986; Poggio & Girosi, 1990; Yoon, 2001), while more general feed-forward architectures have also sometimes been studied (Courrieu, 1993; Fahlman & Lebiere, 1990; Kreinovich, 1991). In such schemes, function approximation capabilities critically depend on the Euclidean metric nature of the input space. However, it is frequent in practical applications that one must approximate functions on data spaces that are not Euclidean, not metric, or even not numerical (symbol strings, graphs). In such cases, one usually attempts to empirically encode the input data in the form of numerical vectors, with the hope, but without any guarantee, that the performed encoding is relevant. Then one applies conventional neural methods on the space of codes, assuming that it is Euclidean and that it suitably preserves the topology of the original data space. Such an empirical encoding can result in a failure in approximating the objective function because the complexity of the approximation to be found not only depends on that of the objective function, but also on the suitability of the encoding. The data-encoding problem has been previously treated for special data types such as clusters, that have been shown to be suitably encodable in Euclidean spaces (Courrieu, 2001). It has also been shown that certain data sets can be monotonically embedded in a Euclidean space, provided that their topology is induced by a function that has at least some properties of a metric, except possibly the triangle inequality (Courrieu, 2002). In these cases, one can apply a conventional neural method for function approximation on the Euclidean embedding space. However, there are also data spaces that cannot be embedded in a Euclidean space without strongly modifying their topology. This is the case, for example, of data spaces where neighborhood relations are not symmetric, or whose topology is induced by a semi-metric (that can take a zero value for pairs of distinct elements). Such topological data spaces are not rare in practice, and it can even happen that their basically non-metric properties are in fact relevant to the function approximation problem to be solved. Consider, for example, objective functions invariant to certain regular transformations of the input, which are common in pattern recognition (see Section 8 of this paper for a meaningful
example). Non-Euclidean and non-metric spaces are also commonly generated by Dynamic Programming methods applied to time series or symbol strings (see Section 9). Thus, there is clearly a need for building function approximation schemes on data spaces with minimal requirements concerning their topology, which is the purpose of this paper. The model proposed hereafter has the form of a layered feed-forward neural network, with special neural basis functions, and it can approximate functions on a wide variety of non-metric data spaces, as well as on any metric space. Data spaces are usually probabilized by a (possibly unknown) sampling probability function on the input space. This property will be explicitly used hereafter.

2. Problem statement

2.1. Input space topology

Let $\Omega$ be any set, and $\delta$ be a real valued function on $\Omega \times \Omega$ such that, for any $X, Y \in \Omega$:
$$\delta(X, X) = 0,$$
$$\delta(X, Y) \geq 0,$$
$$\delta(X, Y) < \infty.$$

Then $(\Omega, \delta)$ is a topological space whose topology is induced by $\delta$. In particular, a closed ball of center $X \in \Omega$, and of radius $r \geq 0$, is the set defined by:
$$B(X, r) = \{ Y \in \Omega, \, \delta(X, Y) \leq r \}.$$

2.2. Sampling probability

Let $\mu$ be a sampling probability on $\Omega$ such that:
$$\mu(\Omega) = 1, \quad \text{for any } X \in \Omega, \quad \mu(B(X, r)) > 0.$$

Note that this last requirement implies some restriction concerning $\delta$. In particular, if $\Omega$ is a continuum and $\mu$ is a continuous probability function, then $\delta$ cannot be a trivial distance because any ball of radius less than 1 would have only one element (its center), and then would be a subset of zero measure of $\Omega$.

2.3. Objective function

Let $f$ be a real valued mapping from $\Omega$ to $\mathbb{R}^K$ such that, for any $X, Y \in \Omega$:
$$|f(X) - f(Y)| \leq \beta \delta(X, Y), \quad 0 < \beta < \infty, \quad 1 \leq i \leq K.$$

If $(\Omega, \delta)$ is a metric space, then the above requirement is simply a Lipschitz condition on $f$. The above property, together with the finiteness of $\delta$ (see Section 2.1), implies that $f$ is bounded on $\Omega$.

2.4. Approximation problem

Find an approximation of $f$, given a learning set of $M$ data points
$$\Xi = \{(X_i, f(X_i)), \, 1 \leq i \leq M\},$$
where points are sampled on $\Omega$ with the probability $\mu$.

One requires that the approximation be uniformly convergent as $M$ tends to infinity.

3. Approximation method

3.1. Prototypical examples

First, one must select a subset of $m$ data points from $\Xi$, with $m \leq M$, in such a way that
$$\min_{1 \leq i \neq j \leq m} \delta(X_i, X_j) = s > 0.$$

Each of these points is a ‘prototype’ that will be associated to a particular basis function. The strictly positive real number $s$ is the minimum ‘spacing’ of prototypes in $(\Omega, \delta)$, while non-prototypical examples are not subject to any spacing constraint. The set of input values of prototypes is denoted $\chi$.

3.2. Low level layer(s)

Given an input $X \in \Omega$, one uses one or more low level layer(s) for computing $\delta$ values between the current input $X$ and the $m$ prototypical input points $X_i$, $1 \leq i \leq m$. The output of this processing is a set of $m$ real values $\delta(X, X_i)$, $1 \leq i \leq m$. Note that the order of the arguments of $\delta$ is relevant since we do not assume that $\delta$ is symmetric. The exact specification of this low level processing of course depends on each particular input space $(\Omega, \delta)$.

3.3. Basis functions

The output of the low level processing (Section 3.2) is used as the input of a layer of $m$ neural basis functions, that are defined as follows:
$$g_i(X; \beta) = \frac{e^{-\beta \delta(X, X_i)}}{\sum_{j=1}^m e^{-\beta \delta(X, X_j)}}, \quad 1 \leq i \leq m.$$

The determination of the real parameter $\beta > 0$ will be studied in the following sections. We note that the basis functions are all positive and that their sum is equal to 1, for any $X \in \Omega$.

3.4. Output layer

The output of the basis function layer (Section 3.3) is used as the input of an output layer of $K$ (linear) neurons.
The synaptic weight of the connection from the \( i \)-th basis function neuron to the \( j \)-th output neuron is denoted \( w_{ij} \) with \( 1 \leq i \leq m \), and \( 1 \leq j \leq K \). The output functions are given by:

\[
q_j(X) = \sum_{i=1}^{m} w_{ij} g_i(X; \beta), \quad 1 \leq j \leq K.
\]

In the following, we will refer to such approximators as ‘\( \varphi \) approximators’.

3.5. Computation of synaptic weights

Synaptic weights are simply computed by a usual Least Square method, which provides an exact interpolation of data points if \( m = M \), or which allows for filtering possible data noise if \( m < M \).

Let \( G \) be the \( M \times m \) real matrix of basis function values for the \( M \) learning examples:

\[
G = (g_i(X_p; \beta)), \quad 1 \leq p \leq M, \quad 1 \leq i \leq m.
\]

Let \( F \) be the \( M \times K \) real matrix of the objective function values for the learning examples.

Let \( W \) be the \( m \times K \) real matrix of synaptic weights to be computed.

The Least Square solution can be obtained, for example, by using the pseudo-inverse method

\[
W = (G^T G)^{-1} G^T F,
\]

where the ‘\( T \)’ denotes the transposition operator.

This assumes that the symmetric matrix \((G^T G)\) is invertible, which can be guaranteed by a suitable choice of the parameter \( \beta \), as we shall see in Section 5. In practice, very large systems can be solved using another approach, such as a Conjugate Gradient method. However, the existence of a solution is always guaranteed by the non-singularity of \( G^T G \).

4. Uniform approximation capability

The uniform convergence proof of the above approximator does not require that \( s > 0 \) (see Section 3.1), thus in this section we can simply consider the case \( m = M \). Before proving the uniform convergence of the approximator, we must generalize a result that is well-known for continuous function approximation on compact subsets of \( \mathbb{R}^n \). In order to simplify the writing, we state the results for \( K = 1 \), while the generalization to any \( K \) is immediate since all output components have similar properties.

Definition 1. The ‘nearest known neighborhood’ of any point \( X \in \Omega \) is the set defined by

\[
N(X) = \{ X_i \in \chi ; \delta(X, X_i) = \min_{1 \leq i \leq m} \delta(X, X_j) \}.
\]

The number of points in \( N(X) \) is denoted \( |N(X)| \), and one has necessarily \( |N(X)| \geq 1 \).

Definition 2. The ‘stepwise approximation’ of a function \( f \) on \( \Omega \) is defined by

\[
Q_m(X) = \sum_{j=1}^{m} q_j(X)f(X_j),
\]

with

\[
q_j(X) = \begin{cases} 1 & \text{if } X_j \in N(X), \\ 0 & \text{otherwise} \end{cases}
\]

Lemma 1. Under the general conditions stated in Section 2, one has, for any \( \varepsilon > 0 \):

\[
\lim_{m \to \infty} \mathop{\text{Prob}} \left[ \sup_{X \in \Omega} |f(X) - Q_m(X)| \geq \varepsilon \right] = 0.
\]

Proof.

(i) Using Section 2.2, for any \( X \in \Omega, r > 0 \Rightarrow \mu(B(X,r)) > 0 \), and thus: for any \( r > 0 \), \( \lim_{m \to \infty} \mathop{\text{Prob}} \left[ \min_{1 \leq i \leq m} \delta(X, X_i) > r \right] = \lim_{m \to \infty} \mu(B(X,r))^m = 0 \).

(ii) Using Section 2.3, \( \delta(X, X_i) \leq r \Rightarrow |f(X) - f(X_i)| \leq a.r, \) with \( a < \infty \), and thus:

\[
|f(X) - Q_m(X)| = \left| f(X) - \frac{1}{|N(X)|} \sum_{X_i \in N(X)} f(X_i) \right|
\]

\[
= \frac{1}{|N(X)|} \sum_{X_i \in N(X)} |f(X) - f(X_i)|
\]

\[
\leq \frac{1}{|N(X)|} \sum_{X_i \in N(X)} |f(X) - f(X_i)|
\]

\[
\leq \frac{1}{|N(X)|} N(X) a.r = a.r.
\]

(iii) One obtains Lemma 1 from (i) and (ii), with \( r = \varepsilon/a \). \( \square \)

We are now ready to prove the following theorem.

Theorem 1. Let \( \beta = \beta(m) \) be a positive increasing function of \( m \) such that \( \lim_{m \to \infty} \beta(m) = \infty \). Then, under the general conditions stated in Section 2, there are synaptic weights \( w_i \), \( 1 \leq i \leq m \), for the approximator \( \varphi(X) \) defined in Section 3.4, such that, for any \( \varepsilon > 0 \):

\[
\lim_{m \to \infty} \mathop{\text{Prob}} \left[ \sup_{X \in \Omega} |f(X) - \varphi(X)| > \varepsilon \right] = 0.
\]

Proof. Given Lemma 1, it suffices to take \( w_i = f(X_i), \) \( 1 \leq i \leq m \), and to show that:

\[
\lim_{m \to \infty} g_i(\beta(m)) = q_i(X).
\]
We note that the $i$th neural basis function can also be written as:

$$g_i(X; \beta) = \frac{1}{\sum_{j=1}^m e^{\beta \delta(X,X_j)}}.$$ 

If $X_i \in N(X)$ then

$$g_i(X; \beta) = \frac{1}{|N(X)| + \sum_{X_j \notin N(X)} e^{\beta \delta(X,X_j)}},$$

where the arguments of the exponentials are all strictly negative, and thus the sum of these exponentials tends to zero as $\beta$ tends to infinity, which leads to the expected result:

$$g_i(X; \beta) \rightarrow \frac{1}{|N(X)|}, \quad \text{as } \beta \rightarrow \infty.$$ 

On the other hand, if $X_i \notin N(X)$ then

$$g_i(X; \beta) \leq \sum_{X_j \notin N(X)} g_i(X; \beta)$$

$$= 1 - \sum_{X_j \in N(X)} g_i(X; \beta) \rightarrow 1 - |N(X)| \frac{1}{|N(X)|}$$

$$= 0, \quad \text{as } \beta \rightarrow \infty,$$

which completes the proof. \hfill \Box

5. Interpolation and Least Square approximation

5.1. Interpolation problem

This is the case $m=M$, with a minimum spacing $s>0$ (see Sections 3.1 and 3.5). The computation of synaptic weights reduces to $W = G^{-1}F$, which requires that the $m \times m$ matrix $G$ be invertible.

**Theorem 2.** With $m > 1$ and $s > 0$, there is a real number $\beta_0$ such that $0 \leq \beta_0 \leq \ln(m-1)/s$, and if $\beta > \beta_0$, then the matrix $G = (g_{ij}) = (g_i(X_j; \beta))$, $1 \leq i,j \leq m$, is invertible.

**Proof.** After the well-known theorem of Gerschgorin–Hadamard, one knows that a sufficient (but not necessary) condition for a square matrix to be invertible is that the absolute value of each of its diagonal coefficients be greater than the sum of the absolute values of all non-diagonal coefficients in the same row. Given that all coefficients of $G$ are positive and that each row has a sum equal to 1, the theorem of Gerschgorin–Hadamard applies if $g_{ii} > 1/2$, $1 \leq i \leq m$.

One has:

$$g_{ii} = \frac{1}{1 + \sum_{j=1}^m \frac{1}{e^{\beta \delta(X,X_j)}}} \geq \frac{1}{2} \Longleftrightarrow \sum_{j=1}^m \frac{e^{-\beta \delta(X,X_j)}}{j \neq i} < 1.$$ 

On the other hand, one has:

$$\sum_{j=1}^m e^{-\beta \delta(X,X_j)} \leq (m-1)e^{-\beta s},$$

$$j = 1$$

$$j \neq 1$$

and finally:

$$(m-1)e^{-\beta s} < 1 \Leftrightarrow \beta > \frac{\ln(m-1)}{s}.$$ 

Thus the lower bound $\beta_0$ is at most equal to $\ln(m-1)/s$, which completes the proof. \hfill \Box

5.2. Least square approximation

This is the case $M > m$, which allows for approximating the objective function while filtering possible data noise. As stated in Section 3.5, one must inverse the symmetric matrix $G^T G$, while $G$ is a rectangular $M \times m$ matrix. The following evidence solves the problem.

**Lemma 2.** If the square $m \times m$ submatrix of $G$ corresponding to the prototypical examples is invertible, then $G^T G$ is invertible.

**Proof.** If the submatrix of $G$ corresponding to the prototypical examples is invertible, then its $m$ column vectors are linearly independent, which implies that the $m$ column vectors of $G$ are also linearly independent since there is no non-zero vector $u$ such that $Gu = 0$. As a consequence, there is no non-zero vector $u$ such that $u^T G^T Gu = 0$, which means that none of the eigenvalues of $G^T G$ is zero, and thus $G^T G$ is invertible. \hfill \Box

As one can see, it suffices to apply Theorem 2 to the set of prototypes to be sure that the Least Square approximation problem has a solution ($W = (G^T G)^{-1} G^T F$). Note however that the above proof assumes that all prototypes actually belong to the learning set.

6. Behavior of the approximator as a function of $\beta$

In order to visualize the behavior of the approximator $\varphi(X)$ as a function of $\beta$, we interpolated a fixed set of 20 distinct data points on the unit square of $R^2$, with $\delta(X,Y) = ||X-Y||^2$ (thus the support space is Euclidean for this visual example), using various values for $\beta$. In this case, each coefficient of the matrix $G$ is equal to a Gaussian divided by a sum of Gaussians, and the matrix $G$ is equal to the product of a non-singular diagonal matrix (inverse sums) by a symmetric matrix of Gaussians. This implies that any $\beta > 0$ can be used (that is $\beta_0 = 0$), since such a matrix $G$ is always invertible (Micchelli, 1986). We have $m=20$, $s=0.02$, and thus $\ln(m-1)/s = 147.22$. One can see, in Fig. 1, interpolation surfaces obtained for $\beta = 150$ (upper panel), and for $\beta = 50$ (lower panel). The interpolation surface obtained
with $\beta = 150$ looks like a ‘smoothed stepwise approximation’, with typical sigmoid profiles between data points. The corresponding minimum diagonal coefficient of $G$ is equal to 0.95, which is unnecessarily large. The interpolation surface obtained with $\beta = 50$ is more regular, although its variation range is wider. The corresponding minimum diagonal coefficient of $G$ is equal to 0.675. Gradually lowering $\beta$, one obtains larger and larger oscillations of the interpolator between data points, and clear symptoms of ill-conditioning (of $G$) for $\beta < 15$, while the minimum diagonal coefficient of $G$ is lower than 0.30.

7. Regularization

One knows that regularizing an approximator is important in order to obtain a good generalization capability from finite samples of data points. Regularizations of approximators on $\mathbb{R}^d$ are commonly obtained by minimizing norms of differential operators (Girosi & Poggio, 1990; Poggio & Girosi, 1990). Unfortunately, such operators are not defined for functions on non-metric spaces. So, we must first define some suitable stabilizer usable on such spaces, which requires some reasonably restrictive additional conditions concerning the space $(\Omega, \delta)$.

7.1. Foundations

**Definition 3.** With $C(X,r) = \{Y \in \Omega, 0 < \delta(X,Y) \leq r\}$, the absolute local variation ratio of an approximator $\varphi$ at point $X \in \Omega$ is defined as:

$$
\Delta \varphi(X) = \lim_{r \to 0} \sup_{Y \in C(X,r)} \frac{|\varphi(X) - \varphi(Y)|}{\delta(X,Y)}.
$$

In the following, we will determine a positive function $V(\beta)$ such that:

$$
\sup_{X \in \Omega} \Delta \varphi(X) \leq V(\beta).
$$
Then there is a particular $\beta$, denoted $\beta^*$, such that

$$V(\beta^*) = \min_{\beta > \beta^*} V(\beta),$$

where $\beta^*$ is the $\beta$ value for which $\min_{1 \leq j \leq m} g_{ij} = 0.5$, and thus $\beta > \beta^*$ guarantees that the matrix $G$ is invertible since $\beta^* > \beta_0$ (see Theorem 2).

In other words, our regularization approach consists of minimizing an upper bound of the absolute local variation ratio of $\phi$ on $(\Omega, \delta)$.

In order to do this, we consider first the case of an exact interpolation ($m = M$, $s > 0$), and we require the following two additional conditions:

- For any $X \in \Omega$, for any $r > 0$, $C(X, r)$ is not empty (thus $\Delta \phi(X)$ is defined).
- There is a positive number $\gamma < \infty$ such that, for any $X, Y, Z \in \Omega$

$$|\delta(X, Z) - \delta(Y, Z)| \leq \gamma \cdot \delta(X, Y).$$

As we shall see, there is no need for knowing the value of $\gamma$, provided that one can assume that it is finite. Note also that in the special case where $\delta$ is a metric, one has $\gamma = 1$, and the condition is equivalent to the triangle inequality (which, of course, is not required here).

**Theorem 3.** Under the above specified conditions, for any $\beta > \beta^*$, one has:

$$\sup_{X \in \Omega} \Delta \phi(X) \leq V(\beta) = \max_{1 \leq j \leq m} |f(X_i)| \cdot \gamma \cdot \frac{\beta}{\min_{1 \leq j \leq m} g_{jj} - 0.5}.$$

**Proof. Step 1**

$$|\phi(X) - \phi(Y)| = \left| \sum_{i=1}^{m} w_i g_i(X; \beta) - \sum_{i=1}^{m} w_i g_i(Y; \beta) \right| = \left| \sum_{i=1}^{m} w_i (g_i(X; \beta) - g_i(Y; \beta)) \right| \leq \max_{1 \leq j \leq m} |w_j| \sum_{j=1}^{m} |g_j(X; \beta) - g_j(Y; \beta)|,$$

and thus, for any $X \in \Omega$:

$$\Delta \phi(X) \leq \left( \max_{1 \leq j \leq m} |w_j| \right) \cdot \left( \lim_{r \to 0} \sup_{Y \in C(X, r)} \sum_{j=1}^{m} |g_j(X; \beta) - g_j(Y; \beta)| / \delta(X, Y) \right).$$

**Step 2**

Given that all prototypes have non-zero spacing, the $m \times m$ matrix $G$ tends to the identity matrix $I$ as $\beta$ tends to infinity, and thus $W = G^{-1} F$ tends to $F$. Now, for $\beta < \infty$, one can use a well-known theorem on linear system conditioning (Ciarlet, 1982, pp. 30–31), which gives

$$||F - W||_\infty \leq \frac{||I - G||_\infty}{1 - ||I - G||_\infty} ||F||_\infty,$$

that is

$$\max_{1 \leq i \leq m} |f(X_i) - w_i| \leq \frac{2(1 - \min_{1 \leq j \leq m} g_{jj})}{1 - 2(1 - \min_{1 \leq j \leq m} g_{jj})} \cdot \max_{1 \leq i \leq m} |f(X_i)|,$$

which implies that

$$\max_{1 \leq i \leq m} |w_i| \leq \left( 1 + \frac{2(1 - \min_{1 \leq j \leq m} g_{jj})}{1 - 2(1 - \min_{1 \leq j \leq m} g_{jj})} \right) \cdot \max_{1 \leq i \leq m} |f(X_i)| = \frac{\max_{1 \leq i \leq m} |f(X_i)|}{2 \cdot \min_{1 \leq j \leq m} g_{jj} - 1}.$$

**Step 3**

First, we note that

$$|\delta(X, Z) - \delta(Y, Z)| \leq \gamma \cdot \delta(X, Y) \Rightarrow e^{-\beta \cdot \delta(X, Z)} \
\in [e^{-\beta \cdot \delta(Y, Z)}, e^{-\gamma \cdot \delta(X, Y)}] \cdot e^{-\beta \cdot \delta(Y, Z)}.$$

Here, we use some interval calculation rules (numbered I1–I12, see Appendix):

$$\sum_{i=1}^{m} |g_i(X; \beta) - g_i(Y; \beta)| \leq \frac{\sum_{i=1}^{m} e^{-\beta \cdot \delta(X, Y)} - \sum_{i=1}^{m} e^{-\beta \cdot \delta(Y, X)}}{e^{-\beta \cdot \delta(X, Y)} - e^{-\beta \cdot \delta(Y, X)}} = \frac{[1, 1] \cdot e^{-\beta \cdot \delta(Y, X)} - [1, 1] \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)} \cdot e^{-\beta \cdot \delta(Y, X)}}{[1, 1] \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)} - [1, 1] \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)}} \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)}$$

$$= \sum_{i=1}^{m} \left[ e^{-2\beta \gamma \cdot \delta(Y, X)} - 1 \right] \sum_{i=1}^{m} e^{-\beta \cdot \delta(Y, X)} \cdot [0, e^{2\beta \gamma \cdot \delta(Y, X)} - 1] \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)}.$$

$$\sum_{i=1}^{m} |g_i(X; \beta) - g_i(Y; \beta)| \leq \frac{\sum_{i=1}^{m} e^{-\beta \cdot \delta(X, Y)} - \sum_{i=1}^{m} e^{-\beta \cdot \delta(Y, X)}}{e^{-\beta \cdot \delta(X, Y)} - e^{-\beta \cdot \delta(Y, X)}} = \frac{[1, 1] \cdot e^{-\beta \cdot \delta(Y, X)} - [1, 1] \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)} \cdot e^{-\beta \cdot \delta(Y, X)}}{[1, 1] \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)} - [1, 1] \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)}} \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)}$$

$$= \sum_{i=1}^{m} \left[ e^{-2\beta \gamma \cdot \delta(Y, X)} - 1 \right] \sum_{i=1}^{m} e^{-\beta \cdot \delta(Y, X)} \cdot [0, e^{2\beta \gamma \cdot \delta(Y, X)} - 1] \cdot \sum_{j=1}^{m} e^{-\beta \cdot \delta(Y, X)}.$$
Since $\sum_{i=1}^{m} g_i(Y; \beta) = 1$, one obtains:

$$\sum_{i=1}^{m} |g_i(X; \beta) - g_i(Y; \beta)| \leq e^{2\beta Y - \delta(X,Y)} - 1.$$ 

Remembering that $(e^u-1)$ is equivalent to $u$ on the neighborhood of 0, we obtain:

$$\lim_{r \to 0} \sup_{Y \in C(X,r)} \sum_{i=1}^{m} |g_i(X; \beta) - g_i(Y; \beta)| \delta(X,Y) \leq \lim_{r \to 0} \sup_{Y \in C(X,r)} \frac{e^{2\beta Y - \delta(X,Y)} - 1}{\delta(X,Y)} \approx 2\beta \cdot \beta.$$ 

Following Step 1, this last result time the result of Step 2 provides an upper bound of $\Delta \phi(X)$ on $\Omega$, which completes the proof of Theorem 3.

We note that the location of a minimizer $\beta^*$ of $V(\beta)$ depends only on the ratio $\beta(\min_{1 \leq i \leq m} g_i(0) - 0.5)$ since the remaining factors of $V(\beta)$ are constant with respect to $\beta$.

7.2. Computation of $\beta^*$

In order to examine the behavior of the function $V(\beta)$ defined in Theorem 3, we generated a large set of square matrices with zero diagonal coefficients and strictly positive random off-diagonal coefficients. These coefficients were used as random $\delta$ values, and we plotted the corresponding $V(\beta)$ functions for $\beta \geq 0$. It turned out that $V(\beta)$ was always unimimimal on its positive part (that is for $\beta \geq \beta^\circ$). A typical profile of $V(\beta)$ can be seen in Fig. 2. Unfortunately, we failed to state a formal proof that $V(\beta)$ is necessarily unimimimal on its positive part, so there is a small doubt that could justify the use of a global optimization method in order to minimize $V(\beta)$. As verification, we applied well-known random walk type global optimization algorithms, whose convergence is guaranteed and that usually provide accurate results (Courrieu, 1997; Ingber & Rosen, 1992). This always provided the same result as the following simple local search procedure, where the matrix $G$ is explicitly expressed as a function of $\beta$ (i.e. $G = G(\beta)$), and the output $\beta^*$ is a minimizer of $V(\beta)$ on its positive part.

Procedure 1

```plaintext
function V(b) = b(\min_{1 \leq i \leq m} g_i(b) - 0.5).
b1 := ln(m)/2; b2 := 1.1*b1; b3 := 1.2*b1;
while V(b1) < V(b2) do
    b3 := b2; b2 := b1; b1 := 2*b2 - b3;
while \min_{1 \leq i \leq m} g_i(b1) \leq 0.5 do b1 := (b1 + b2)/2;
end
while V(b3) < V(b2) do
    b1 := b2; b2 := b3; b3 := 2*b2 - b1;
end
while (b3 - b1) > precision*b2 do
    c1 := (b1 + b2)/2; c2 := (b2 + b3)/2;
    if V(c1) \leq V(b2) then
        b3 := b2; b2 := c1;
    else if V(b2) \leq V(c2) then
        b1 := c1; b3 := c2;
        b1 := b2; b2 := c2;
    end
end
\beta^* := b2.
```

Note that the above procedure is written to be easily readable, however, in practical implementations, one must of course avoid repeated calls to the $V(\beta)$ function with the same argument, and reusable values must be stored. This procedure has been applied to the illustrative interpolation problem of Section 6. The obtained value was $\beta^* \approx 70$, while the corresponding minimum diagonal coefficient of $G$ was 0.78. The obtained interpolation surface is shown in Fig. 3, where the regularization provided by $\beta^*$ seems effective. We note that the interpolation surface is close to that obtained with $\beta = 50$ (Fig. 1, lower panel), but that surface oscillations between data points are a bit smaller.

Finally, we note that $\beta^*$ can be computed in all cases, provided that $s > 0$. In the case of a Least Square approximation ($m < M$), $\beta^*$ must be computed for the square $m \times m$ submatrix of $G$ corresponding to the prototypical examples. $\beta^*$ can also be computed whether we know that $\gamma$ is finite or not, since $\gamma$ is not actually used in practical computation. Similarly, the condition that $C(X,r)$ is never empty, which excludes discrete spaces, is not necessary for $\beta^*$ computation. Thus we conclude that $\beta^*$ can always be used at least as a reasonable default parameter, while its full theoretical justification of course requires the conditions stated in (Section 7.1).

8. The example of affinely invariant pattern functions

There are many kinds of data spaces on which the above function approximation scheme can be used because the
requirements concerning \((\mathcal{O}, \delta)\) are very weak. As an example, we chose a problem that, at first glance, can seem to be of metric nature, but that in fact is not. This is the problem of approximating a function on a space of patterns, while the function is invariant to affine transformations of the input. Of course, one can ignore this last property and build a metric input space in which each possible pattern is considered as independent of others. Another approach consists of taking into account the invariance property in order to improve the generalization and to reduce the required amount of learning, but this leads to build some non-metric input space, as we shall see hereafter.

8.1. Input patterns

We consider here \(\mathcal{O}\) as the set of sequences of \(L\) points of a bounded subset of \(\mathbb{R}^d\), with \(L > n\) Any sequence is represented in the form of a \(L \times (n+1)\) real matrix \(X\) whose first column coefficients are equal to 1, and the remaining \(n\) columns correspond to the coordinates of points. A sequence \(X\) belongs to \(\mathcal{O}\) if \(\det(X'X) > 0\), which means that \(X\) is of rank \(n+1\), or equivalently that the set of \(L\) points is actually of dimension \(n\).

8.2. The \(\delta\) function

We must define a \(\delta\) function such that for any \(X,Y \in \mathcal{O}\), \(\delta(X,Y) = 0\) if there is an affine transformation \(T\) of the \(n\) coordinates such that \(T(X) = Y\). Given that we added a constant unit coordinate to each point in order to compute translations, it is equivalent to say that \(\delta(X,Y) = 0\) if there is a square \((n+1) \times (n+1)\) matrix \(T\) such that \(XT = Y\). The following lemma will be useful.

**Lemma 3.** If \(X\) and \(Y\) are both of rank \(n+1\) and there is \(T\) such that \(XT = Y\), then \(T\) is invertible.

**Proof.** If \(T\) is not invertible, then there is a non-zero vector \(u\) such that \(Tu = 0\). Then \(XT = Y\) implies that \(XTu = Yu = 0\), and thus \(Y\) is not of rank \(n+1\), which contradicts the hypothesis.

Now, one can define \(\delta(X,Y)\), for example, as the least square error function:

\[
\delta(X,Y) = \inf_T ||XT - Y||^2 = ||(X(X'X)^{-1}X' - I)Y||^2.
\]

This \(\delta\) function obviously satisfies the requirements of Section 2.1, however one can have \(\delta(X,Y) = 0\) while \(X \neq Y\), and if \(\delta(X,Y) \neq 0\), then one has in general \(\delta(X,Y) \neq \delta(Y,X)\). Thus \(\delta(X,Y)\) is certainly not a metric and it cannot by monotonically transformed into a metric (Courrieu, 2002). The first additional requirement of Section 7.1 is satisfied since \((\mathcal{O}, \delta)\) is in fact a continuum. Now, we must verify that \(\delta\) satisfies the second additional requirement of (Section 7.1) that \(\gamma\) is finite, which allows for applying Theorem 3, and thus theoretically justifies the use of \(\beta^*\).

**Theorem 4.** With \(\mathcal{O}\) and \(\delta\) defined as above, there is a positive number \(\gamma < \infty\) such that, for any \(X,Y,Z \in \mathcal{O}\),

\[
|\delta(X,Z) - \delta(Y,Z)| \leq \gamma \cdot \delta(X,Y).
\]

**Proof.** Given that all data points belong to a bounded subset of \(\mathbb{R}^d\), and that for any \(X \in \mathcal{O}\), the matrix \(X'X\) is invertible, we have that for any \(X,Y \in \mathcal{O}\), \(\delta(X,Y) < \infty\), and thus for any \(X, Y, Z \in \mathcal{O}\), \(\|\delta(X, Z) - \delta(Y, Z)\| < \infty\). This implies that, if \(\delta(X,Y) > 0\), then \(\|\delta(X,Z) - \delta(Y,Z)\|/\delta(X,Y) < \infty\). On the other hand, if \(\delta(X,Y) = 0\), then there is a transformation matrix \(T\) such that \(XT = Y\), and after Lemma 3, this matrix is invertible, thus \(YT^{-1} = X\). Assume that \(\delta(X,Z) = ||XU - Z||^2\), and \(\delta(Y,Z) = ||YV - Z||^2\). If \(\delta(X,Z) < \delta(Y,Z)\), then \(\delta(X,Z)\) is not the least square solution since \(||YT^{-1}U - Z||^2 < \delta(Y,Z)\). Similarly, if \(\delta(X,Z) > \delta(Y,Z)\), then \(\delta(X,Z)\) is not the least square solution since \(||XTV - Z||^2 < \delta(X,Z)\). Since \(\delta\) is in all cases the least square error function, we can conclude that if \(\delta(X,Y) = 0\) then \(\|\delta(X,Z) - \delta(Y,Z)\| = 0\), for any \(Z \in \mathcal{O}\). □
8.3. Computational test

For this test, we used as input patterns sequences of 4 points of [0,1]², and sequences of 12 points of [0,1]², while patterns were encoded as described in Section 8.1. Learning example patterns were randomly generated, while generalization test patterns were generated in the following way. For each generalization input Z, a learning example X was selected, a random affine transformation matrix T and a random pattern R were generated, and finally:

\[ Z = (1 - \eta)XT + \eta R, \quad 0 \leq \eta \leq 1. \]

Whenever \( \eta = 0 \), Z is a random affine transform of a learning example. Whenever \( \eta = 1 \), Z is completely independent of the learning set. Five \( \eta \) values were used for the test: 0, 0.25, 0.5, 0.75, and 1. The size \( m \) of the learning set was varied from 15 to 240, and for each \((m,\eta)\) combination, 60 generalization patterns were generated. An artificial objective function invariant to affine transformations of the input was built in the following way

\[ f(X) = 100/(1 + c \cdot \delta(X, X_0)), \quad c = 28/(nL), \]

where the \( \delta \) function is defined as in Section 8.2, and \( X_0 \) is a fixed reference pattern that does not belong to the learning set.

For comparison, we tested three types of interpolators. The first one, referred to as ‘NN-invar’, is the \( \varphi \) approximator with the \( \delta \) function invariant to affine transformations defined in Section 8.2. The second one, referred to as ‘NN-metric’, is the \( \varphi \) approximator with a \( \delta \) function, say \( \delta' \), that is a squared Euclidean metric on \( R^nL \), namely \( \delta'(X,Y) = \|X - Y\|^2 \). In all cases, \( \varphi \) approximators were tested with \( \beta = \beta^* \), where \( \beta^* \) was computed by Procedure 1. Now, \( \delta' \) can also be used with usual radial basis function approximators, since it is a squared Euclidean metric. We chose Radial Splines as the third type of interpolator. Radial Spline interpolators are of common use, they have well-known uniform approximation and regularization capabilities on Euclidean spaces (Girosi & Poggio, 1990; Poggio & Girosi, 1990), and they do not require any free parameter tuning. Given that \( nL \) is always even here, we used Radial Spline basis functions of the form:

\[ S(X, Y) = \ln(r) \cdot r^2, \quad \text{with} \quad r^2 = \delta'(X, Y) = ||X - Y||^2. \]

Comparing the generalization performance of NN-invar to that of NN-metric allows for evaluating the interest of using non-metric input spaces in this type of problem. Comparing the generalization performance of NN-metric to that of Spline interpolators (necessarily on a Euclidean space) provides an evaluation of general capabilities of \( \varphi \) approximators with respect to a well-known reference.

Results of the test are reported in Table 1, for \( L = 4 \) and \( n = 2 \), and in Table 2, for \( L = 12 \) and \( n = 4 \). Tables show the mean absolute generalization error (for 60 test items), and the corresponding standard deviation in parenthesis, in the various \((m,\eta)\) conditions for the three types of interpolators. In addition, Student \( t \)-tests were performed in order to test the difference of performance between NN-invar and NN-metric interpolators, and between NN-metric and Spline interpolators. The notation ‘n.s’ means that the difference between the mean just above and the mean just below ‘n.s’ is statistically non-significant. The notation \( \cong \) (or \( \approx \)) means that the difference between means is marginally significant (\( p < 0.10 \)). The notation \( \cong \) (or \( \approx \)) means that the difference is significant (\( p < 0.05 \)), according to usual decision criterions, while the notation \( \cong \) (or \( \approx \)) means that the difference is highly significant (\( p < 0.01 \)).

Although there are some visible differences between the processing of the smallest patterns (Table 1) and that of the largest patterns (Table 2), we can make the following general observations. First, the NN-invar interpolator always provides zero generalization error for \( \eta = 0 \), which simply confirms that this type of network ‘recognizes’ known patterns independently of affine transformations. For \( \eta > 0 \), the advantage of NN-invar over NN-metric is not systematic with small learning sets, howevever, NN-invar

<table>
<thead>
<tr>
<th>( L = 4 ), ( n = 2 )</th>
<th>( \eta = 0 )</th>
<th>( \eta = 0.25 )</th>
<th>( \eta = 0.50 )</th>
<th>( \eta = 0.75 )</th>
<th>( \eta = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m = 15 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-invar</td>
<td>0 (0)</td>
<td>22 (17)</td>
<td>26 (21)</td>
<td>19 (14)</td>
<td>28 (22)</td>
</tr>
<tr>
<td>NN-metric</td>
<td>15 (11)</td>
<td>24 (15)</td>
<td>23 (15)</td>
<td>19 (13)</td>
<td>25 (13)</td>
</tr>
<tr>
<td>Spline</td>
<td>777 (519)</td>
<td>676 (394)</td>
<td>755 (359)</td>
<td>712 (371)</td>
<td>880 (750)</td>
</tr>
<tr>
<td>( m = 30 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-invar</td>
<td>0 (0)</td>
<td>12 (14)</td>
<td>11 (11)</td>
<td>12 (10)</td>
<td>12 (12)</td>
</tr>
<tr>
<td>NN-metric</td>
<td>18 (13)</td>
<td>22 (11)</td>
<td>22 (13)</td>
<td>19 (13)</td>
<td>20 (14)</td>
</tr>
<tr>
<td>Spline</td>
<td>85 (54)</td>
<td>64 (40)</td>
<td>62 (41)</td>
<td>60 (48)</td>
<td>102 (81)</td>
</tr>
<tr>
<td>( m = 60 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-invar</td>
<td>0 (0)</td>
<td>12 (15)</td>
<td>15 (17)</td>
<td>11 (14)</td>
<td>13 (14)</td>
</tr>
<tr>
<td>NN-metric</td>
<td>18 (13)</td>
<td>24 (13)</td>
<td>20 (13)</td>
<td>21 (12)</td>
<td>21 (13)</td>
</tr>
<tr>
<td>Spline</td>
<td>19 (14)</td>
<td>22 (13)</td>
<td>21 (12)</td>
<td>22 (14)</td>
<td>18 (15)</td>
</tr>
<tr>
<td>( m = 120 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-invar</td>
<td>0 (0)</td>
<td>6.9 (8.8)</td>
<td>7.8 (9.0)</td>
<td>7.6 (8.7)</td>
<td>8.2 (11)</td>
</tr>
<tr>
<td>NN-metric</td>
<td>18 (15)</td>
<td>20 (14)</td>
<td>20 (15)</td>
<td>17 (13)</td>
<td>15 (12)</td>
</tr>
<tr>
<td>Spline</td>
<td>21 (13)</td>
<td>21 (13)</td>
<td>18 (13)</td>
<td>19 (13)</td>
<td>18 (14)</td>
</tr>
<tr>
<td>( m = 240 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-invar</td>
<td>0 (0)</td>
<td>5.3 (6.3)</td>
<td>5.1 (6.4)</td>
<td>4.6 (7.2)</td>
<td>3.3 (4.5)</td>
</tr>
<tr>
<td>NN-metric</td>
<td>21 (16)</td>
<td>20 (13)</td>
<td>18 (17)</td>
<td>17 (14)</td>
<td>17 (13)</td>
</tr>
<tr>
<td>Spline</td>
<td>22 (14)</td>
<td>19 (13)</td>
<td>22 (15)</td>
<td>15 (12)</td>
<td>16 (11)</td>
</tr>
</tbody>
</table>

The \( \varphi \) approximator ‘NN-invar’ uses a non-metric input space with affine invariance properties.
9. Non-metric $\delta$ functions from dynamic programming

Dynamic Programming methods provide another source of non-metric $\delta$ functions. These methods allow for comparing numerical sequences or symbol strings of different lengths, and they have been extensively applied in speech recognition (Sakoe & Chiba, 1978). Although the resulting $\delta$ functions can have some properties of distances under reasonably restrictive conditions, most of these functions are not metrics, since they usually break the triangle inequality (Okochi & Sakai, 1982). Whenever the triangle inequality is the only distance property that is broken, one can monotonically embed the data set into a Euclidean space (Courrieu, 2002), and then apply a usual neural algorithm on the embedding space. However, it can be more convenient to directly use a $\varphi$ approximator on the original data space, which avoids prior input data embedding, or possibly undesirable restrictions concerning $\delta$, and does not require prior knowledge of all properties of $\delta$, except those listed in Section 2.1.

As an example, we briefly consider hereafter the well-known algorithm of Sakoe and Chiba (1978). Let $\Omega$ be a set of sequences of points of $\mathbb{R}^d$, and let $d$ be some metric associated to $\mathbb{R}^d$. Let $X=(x_1,x_2,...,x_m)$, and $Y=(y_1,y_2,...,y_n)$ be two sequences, of length $m>1$ and $n>1$, respectively, belonging to $\Omega$. The algorithm statement is:

$$D(1,1)=2d(x_1,y_1);$$
for $j=2...n$ $D(1,j)=D(1,j-1)+d(x_1,y_j);$  
for $i=2...m$ $D(i,1)=D(i-1,1)+d(x_i,y_1);$  
for $i=2...m$ $D(i,j)=\min[D(i,j-1)+d(x_i,y_j),$ 
$D(i-1,j)+2d(x_i,y_j), D(i-1,j)+d(x_i,y_j)]$;  
$\delta(X,Y)=D(m,n)/(m+n).$

The above $\delta$ function is insensitive to repetition (e.g. $\delta((1,1,2,3,3),(1,2,2,3,3))=0$), which makes it suitable for comparing sequences of regularly sampled acoustical parameters in speech recognition, given that speech speed is naturally variable. As an example of triangle inequality breaking, consider $X=(1,2,3)$, $Y=(4,5,6)$, $Z=(2,5)$. Then one obtains $\delta(X,Y)=2.333...$, $\delta(X,Z)=1$, $\delta(Y,Z)=1$, thus $\delta(X,Y)>\delta(X,Z)+\delta(Y,Z)$. The above algorithm can as well be used to compare symbol strings provided that one can define some natural distance $d$ between the elements of the used alphabet (Courrieu, Farioli, & Grainger, in press). Its behavior is quite different from that of well-known ‘edition distances’ for character strings (Lowrance & Wagner, 1975; Wagner & Fischer, 1974).

10. Conclusion

We have defined a function approximation scheme that can be expressed as a layered feed-forward neural network, and that is able to uniformly approximate functions on a wide
variety of non-metric spaces as well as on any metric space, while usual approximators are guaranteed to work only on real Euclidean spaces. Non-Euclidean metric or non-metric data spaces are commonly encountered in practical applications. For example, time series can be compared using Dynamic Programming methods (elastic matching), but the resulting space is not metric, in general. The same is true for spaces of symbol strings, or for spaces of graphs. Another example is that of spaces of real patterns invariant to some class of transformations, while the particular case of affine invariance has been detailed and illustrated above. Hence, the proposed tool clearly responds to practical needs that have been poorly investigated previously. Theoretical foundations are provided concerning the uniform approximation capability of the approximator, the solution of interpolation and least square approximation problems, and an approach of regularization suitable to the considered data spaces. All required practical algorithms are simple, and computational examples are provided that clearly show the suitability of the tool. As a final remark, we note that ϕ approximators have conventional architectures, and that their specificity simply resides in their special basis functions, although these basis functions themselves do not have an extremely ‘exotic’ form, except that they accept very weakly constrained δ functions as arguments, where Radial Basis Functions, for example, require Euclidean metrics. This leads us to suspect that other approximators with similar capabilities could exist, however, their theoretical investigation remains to do. This is of interest because the result is, in fact, a significant extension of the concept of ‘universal approximation capability’.

Appendix. Interval calculation

Interval calculation has been developed by Moore (1966) and Ratschek and Rokra (1984). Interval arithmetic is also reported in Zhiigljavsky (1991). We list hereafter some useful rules (I1–I12) whose consistency is easy to verify. We consider here closed real intervals of the form Z = (a, b], with a ≤ b.

I1. For x ∈ R, x = [x, x] = [1, 1]·x.

Interval arithmetic

I2. Z1 + Z2 = [a1 + a2, b1 + b2]
I3. Z1 − Z2 = [a1 − b2, b1 − a2]
I4. Z1·Z2 = [min(a1a2, a1b2, a1b1, a2b2, b1b2), max(a1a2, a1b2, a1b2, b2b2)]
I5. Z1/Z2 = Z1 · [1/b2, 1/a2], if 0 ∉ Z2.

Interval functions

I6. If h is a monotonic increasing function, then h(Z) = [h(a), h(b)].

I7. If h is a monotonic decreasing function, then h(Z) = [h(b), h(a)].

I8. The absolute value function of an interval is given by:

\[ |[a, b]| = \begin{cases} 0, & \text{if } ab \leq 0 \\ \max(|a|, |b|), & \text{otherwise} \end{cases} \]

Intervals and numbers

I9. |[a, b]·x| = |[a, b]|·|x|
I10. \[ \frac{[a1, b1]}{[a2, b2]}\cdot x1 = \frac{[a1, b1]}{[a2, b2]}\cdot x2, \quad \text{if } a1, b1 \]
\[ \geq 0, \quad \text{and } a2, b2, x2 > 0 \]
I11. \[ [a, b]·x - [1, 1]·x = [a - b - 1]·x \]
I12. \[ \sum_{i=1}^{m}[a, b]x_i = [a, b]·\sum_{i=1}^{m}x_i, \quad \text{if } x_i \geq 0, \quad l \leq i \leq m \]

References


