High-Dimensional Function Approximation with Neural Networks for Large Volumes of Data

Peter Andras, *Senior Member, IEEE*

***Abstract*— Approximation of high-dimensional functions is a challenge for neural networks due to the curse of dimensionality. Often the data for which the approximated function is defined resides on a low dimensional manifold and in principle the approximation of the function over this manifold should improve the approximation performance. Projecting the data manifold into a lower dimensional space, followed by the neural network approximation of the function over the projection space have been shown to be more precise than the approximation of the function with neural networks in the original data space. However, if the data volume is very large, the projection into the low-dimensional space has to be based on a limited sample of the data. Here we investigate the nature of the approximation error of neural networks trained over the projection space. We show that such neural networks should have better approximation performance than neural networks trained on high-dimensional data even if the projection is based on a relatively sparse sample of the data manifold. We also find that it is preferable to use a uniformly distributed sparse sample of the data for the purpose of the generation of the low-dimensional projection. We illustrate these results considering the practical neural network approximation of a set of functions defined on high dimensional data including real world data as well.**

***Index Terms*—big data, function approximation, high-dimensional data, manifold mapping, neural networks.**

I. Introduction

OFTEN problems in engineering and science require the approximation of functions defined over high-dimensional data (e.g. data with more than 10 dimensions, and possibly with 100s or more dimensions) [1-3]. For example, consider the estimation of the likelihood of faults in complex engines equipped with a wide range of sensors, or the association of the likelihood of brain scale dysfunction with high resolution EEG or neuro-imaging data (e.g. fMRI data with millions of voxels). Neural networks are a commonly used tool to perform such approximation tasks [3-5]. However, the high-dimensionality of the data on which the approximated function is defined means that the sampling density of the data is usually low even in the case of large volumes of data and that the expected error of the approximation is also large, or in other words the curse of dimensionality applies [3, 6-9].

In many cases the high-dimensional data resides around a low-dimensional manifold [4, 9-11]. The usual reason for this is that the system which is characterized by the data has a state trajectory in a much lower dimensional space than the number of measurements or indicators that are used to record the behavior of the system. Thus there are many constraining relationships between the components of data vectors restraining the data vectors to low-dimensional manifolds, although these constraints are generally not known.

A key idea to deal with the curse of dimensionality in the context of neural network approximation of functions defined over such high-dimensional data residing on a low-dimensional manifold, is to approximate the function over the manifold and not over the whole high-dimensional space [9, 12]. In practice this can be realized by projecting first the data manifold onto a low-dimensional space [11, 13] and then approximating the function with a neural network over this low-dimensional projection space [9]. The reason for the projection step is that in general the formal equations defining the data manifold are not known and consequently there is no obvious way to define the function directly in a restricted manner over the data manifold only.

The above noted approach in principle works by projecting the whole data set onto the low-dimensional space and considering a sample of the approximated function as the set of function values associated with projected data points [9]. However, in the case of big data, when the volume of the data may be beyond the limits of storage of the data analysis system, e.g. very large volumes of astronomy data, high-frequency data from very many sensors, this approach would not work in this way. Thus, it is important to consider how to overcome the potential limitation of data storage for neural network approximation of functions defined over high-dimensional data residing on low-dimensional manifolds.

In this paper we investigate the nature of the approximation error in the case of neural networks function approximation using high-dimensional data projected onto a low-dimensional space. We show that the behavior of the approximation error is such that even if the projection space is defined using a relatively small sample of the data manifold, the approximation performance of the neural network defined

Paper re-submitted to review on 17th September 2016

P. Andras is with the School of Computing and Mathematics, Keele University, Keele, Staffordshire, ST5 5BG, UK (e-mail: p.andras@keele.ac.uk).

over the projection space is still better than the approximation performance of a neural network trained with the original high-dimensional data. This shows that in the case of big data scenarios a small sample of the data is sufficient to define the low-dimensional projection of the data manifold such that the neural network approximation of the function will be sufficiently good. Thus there is no need to retain all the data for the purpose of the definition of the low-dimensional projection of the data manifold.

The rest of the paper is structured as follows. First we briefly review the relevant related works. Next we describe briefly the projection based neural network function approximation and present the analysis of the approximation error of such neural networks. This is followed by the presentation of a set of application examples using selected functions defined over high-dimensional data and including a real world data set as well. The paper is closed by the conclusion section.

II. Related Works

Neural networks have the universal approximation property with respect to continuous functions, i.e. the ability to approximate arbitrarily correctly any continuous function, given that they have sufficiently many hidden neurons with activation functions belonging to an appropriate class of functions (e.g. sigmoidal or Gaussian functions) [4, 14, 15]. While this results holds in principle, in practice the required number of neurons may be excessively large and the existential results about the universal approximation property do not provide advice on how to find the appropriate parameters for the hidden neurons.

In general, neural network approximation of functions suffers from the curse of dimensionality, in the sense that the approximation error grows exponentially with the dimensionality of the data [7, 8, 16]. The approximation error of neural networks with a single hidden layer of neurons with nonlinear activation functions is inversely proportional with the square-root of the number of neurons and proportional with a factor which grows exponentially with the dimensionality of the data space, i.e. the error is proportional with $b^d / \sqrt{n}$, where n is the number of neurons, d is the dimensionality of the data and $b > 1$ is a constant depending on the class of the activation functions [8, 16]. We note that in particular cases, the approximation error bounds are smaller and do not grow exponentially with the dimensionality of the data [17], however these special cases do not apply very often.

The high-dimensional data, over which the approximated function is defined, often resides on a low-dimensional manifold [4, 9-11]. There are several methods for mapping of data manifolds onto low-dimensional spaces. One such method is the local-linear embedding (LLE) [11]. This method first finds a linear approximation of each data point by its closest neighbouring data points, then using the coefficients of these linear approximations maps the data points onto a low dimensional space such that the projections of the data points in this space are forced to satisfy the linear approximation relationships applied to the projection data points. In terms of equations, for each data point $\mathbf{x}^k, k = 1,...,m$, the r closest neighbouring data points are $\mathbf{x}^{j(k,i)}, i = 1,...,r$, then we find the best approximation of $\mathbf{x}^k$ in the form of

$$\sum_{i=1}^{r} a_{k,i} \cdot \mathbf{x}^{j(k,i)} \tag{1}$$

Following this we map the data points $\mathbf{x}^k$ onto $\mathbf{y}^k$ in the low-dimensional projection space, such that we minimize the expression

$$E = \sum_{k=1}^{m} \left\| \mathbf{y}^k - \sum_{i=1}^{r} a_{k,i} \cdot \mathbf{y}^{j(k,i)} \right\| \tag{2}$$

subject also to the following constraints $\sum_{k=1}^{m} \mathbf{y}^k = 0$ and $\left(1/m\right) \cdot \sum_{k=1}^{m} \mathbf{y}^k \mathbf{y}^{kT} = I^{d'}$ where $I^{d'}$ is the $d' \times d'$ identity matrix and d' is the dimensionality of the projection space. Alternative dimension reduction projections of data manifolds include the self-organising maps, ISOMAP, and other methods [13, 18].

Recently the combination of low-dimensional mapping and neural network approximation has been suggested for the approximation of functions defined over high-dimensional data [9]. It has been shown that the approximation performance of neural networks built using data projected with self-organising maps onto the low-dimensional space, is much better than the approximation performance of neural networks trained with the original high-dimensional data [9].

It has been shown that the local linear approximation of data points by other data points in the high-dimensional space in the manner described above for the LLE can be used to build a linear approximation of the function defined over the high-dimensional data [12]. The error bounds provided for such linear approximations of the function in the high-dimensional space indicate that such linear approximation can be sufficiently precise [12].

III. Error Bounds for Neural Network Approximation Following Low-dimensional Projection of the Data

We assume that the data points $\mathbf{x}^k \in \mathbf{R}^d, k = 1,...,m$, reside on a data manifold of dimension $d' < d$. This implies that there is a mapping $\varphi^* : \mathbf{R}^d \rightarrow \mathbf{R}^{d'}$ that maps high-dimensional data points into the low-dimensional projection space that corresponds to the low-dimensional data manifold. We assume that we can estimate d' with an appropriate dimension determination method, for example by using a ball counting estimation of the data manifold's dimensionality [18]. (For this approach of dimension estimation non-overlapping balls of decreasing radius are used to cover the part of the space where the data points reside. The number of balls is considered as a power function of the radius, i.e.

$N(r)=r^{-H}$. The calculated power value (H) is an estimation of the Hausdorff dimension of the data manifold. The integer part of this is the estimated dimensionality of the data manifold $d'=[H]$.)

We use LLE to project the high-dimensional data points into a low-dimensional space of dimension d', we denote this mapping of data as φ. Using equations (1) and (2) we calculate the low-dimensional projections of the data points. Following the optimisation of the approximation of $\mathbf{x}^k$ by the expression in equation (1) we get the following linear coefficients

$$a_{k,i}=\frac{\sum_{j=1}^{r}c_{k,i,j}^{-1}}{\sum_{h=1}^{r}\sum_{j=1}^{r}c_{k,h,j}^{-1}} \tag{3}$$

where

$$c_{k,l,h}=<\mathbf{x}^k-\mathbf{x}^{j(k,l)},\mathbf{x}^k-\mathbf{x}^{j(k,h)}> \tag{4}$$

We assume that the LLE mapping of the data manifold onto the low-dimensional space, i.e. $\varphi(\mathbf{x}^k)=\mathbf{y}^k$, is calculated using a sample of the data manifold, i.e. $\mathbf{x}^k\in\mathbf{R}^d, k=1,...,m$. For other data points $\mathbf{x}$ not included into the sample used to calculate the low-dimensional mapping, we calculate their low dimensional mapping by first determining their closest r neighbors $\mathbf{x}^{j(k,i)}, i=1,...,r$, among $\mathbf{x}^k\in\mathbf{R}^d, k=1,...,m$. Then calculate their linear approximation coefficients a_i using equation (3) and having

$$c_{l,h}=<\mathbf{x}-\mathbf{x}^{j(l)},\mathbf{x}-\mathbf{x}^{j(h)}> \tag{5}$$

Finally we calculate their low dimensional projection as

$$\mathbf{y}=\sum_{i=1}^{r}a_i\cdot\mathbf{y}^{j(i)}=0 \tag{6}$$

where $\mathbf{y}^{j(i)}=\varphi(\mathbf{x}^{j(i)})$.

We use neural networks to approximate the target function f defined on the high dimensional data set. The neural networks are either trained using the high-dimensional data points or the low dimensional projections of the data points. We consider neural networks with one hidden layer, having neurons with Gaussian activation function, i.e. $g(\mathbf{x})=e^{-\frac{\|\mathbf{x}-\mathbf{q}\|^2}{\sigma^2}}$ with $\mathbf{q}$ and σ being parameters of the neuron. It is assumed that for the training of the neural networks all training data points are used, not only the ones that are included in the calculation of the low-dimensional mapping of the data manifold. This means that the training data can be much larger than the data sample used for the mapping calculation. We aim to show that the approximation performance of neural networks trained low-dimensional projected data is better than the approximation performance of neural networks trained with the high-dimensional original data.

In general, in the context of very large data sets, characteristic of big data problems, it is expected that the data that can be used for the calculation of the low dimensional mapping of the data manifold is a relatively small sample of the full data set. We aim to show that even in such conditions the neural networks trained with low-dimensional projection data approximate better the target function than neural networks trained with high-dimensional data. We also aim to show that the best approximation performance by neural networks trained with low-dimensional projection data is achieved if the LLE mapping of the high-dimensional data is generated using a uniformly distributed sample of data.

Let us briefly explain the meaning of fundamental assumptions required for the validity of our results. We assume that the data manifold is compact and smooth. These mean that the d' dimensional data manifold is such that the size of any ε-size covering of the manifold is bounded by $c\cdot\varepsilon^{-d'}$, where c is a constant depending on the data manifold and that its derivative at any point is defined and bounded. We also assume that the true mapping function φ^* is (α,β)-Lipschitz smooth with respect to the Euclidean norm, which means that

$$\left|\varphi^*(\mathbf{x})-\varphi^*(\mathbf{x}')\right|\le\alpha\cdot\|\mathbf{x}-\mathbf{x}'\| \tag{7}$$

and

$$\left|\varphi^*(\mathbf{x})-\varphi^*(\mathbf{x}')-\nabla\varphi^*(\mathbf{x})^T\cdot(\mathbf{x}'-\mathbf{x})\right|\le\beta\cdot\|\mathbf{x}-\mathbf{x}'\|^2 \tag{8}$$

for some $\alpha,\beta>0$. We further assume that the target function f is smooth, meaning that it has well defined and bounded derivatives everywhere.

THEOREM 1. Let us consider φ as the function representing the LLE mapping of high dimensional data onto the low dimensional data manifold. Let us assume the high-dimensional data manifold is compact and smooth. Let us further assume that the true low-dimensional mapping function of the high-dimensional data φ^* is (α,β)-Lipschitz smooth with respect to the Euclidean norm. Then there exist constants Δ,ε_D and C such that

$$\left\|\varphi^*(\mathbf{x})-\varphi(\mathbf{x})\right\|\le\Delta=\sqrt{d'}\cdot C\cdot\varepsilon_D^2 \tag{9}$$

for any $\mathbf{x}$ on the high-dimensional data manifold.

Proof: Following from the results of Yu et al. [12] and considering the assumptions about the manifold and considering the mapping functions φ and φ^* for each coordinate of the low-dimensional space separately, i.e., φ_j and φ_j^* for $j=1,...,d'$, we have that for any $\boldsymbol{x}$ on the data manifold we have that

$$\left|\varphi_j^*(\mathbf{x})-\varphi_j(\mathbf{x})\right|\le C\cdot\varepsilon_D^2 \tag{10}$$

where ε_D is a constant depending on the distances between pairs of points within the set of data points in the sample chosen for defining the mapping of the data manifold onto the low-dimensional space and C is a constant that depends on the data manifold itself or equivalently on the true mapping of the data manifold onto the low dimensional space φ^*.

This implies that

$$\left\|\varphi^*(\mathbf{x})-\varphi(\mathbf{x})\right\| \le \sqrt{d'}\cdot C\cdot \varepsilon_D^2 = \Delta \tag{11}$$

for any $\mathbf{x}$ on the high dimensional data manifold. **Q.e.d.**

THEOREM 2. Let us denote as $f(\mathbf{x})$ the target function defined on the high-dimensional data and let us assume that this is a smooth function. Let us denote as φ and φ^* the LLE mapping and the unknown true mapping of the high-dimensional data onto the low-dimensional manifold and assume that the assumptions of Theorem 1 are satisfied. Let us define the functions $f^{\#}(\mathbf{y})$ and $f^*(\mathbf{y})$ over the low-dimensional projection space as $f^{\#}(\mathbf{y}) = f^{\#}(\varphi(\mathbf{x})) = f(\mathbf{x})$ and $f^*(\mathbf{y}) = f^*(\varphi^*(\mathbf{x})) = f(\mathbf{x})$. Let us denote as $g(\mathbf{x})$ and $g^{\#}(\mathbf{y})$ the functions representing the single hidden layer neural network approximations of $f(\mathbf{x})$ and $f^{\#}(\mathbf{y})$. Then

$$\left|f^*(\mathbf{y})-g^{\#}(\mathbf{y})\right| \le \left|f(\mathbf{x})-g(\mathbf{x})\right| \tag{12}$$

for any $\mathbf{x}$ on the high-dimensional data manifold and $\mathbf{y}=\varphi^*(\mathbf{x})$, if the constant ε_D defined in Theorem 1 is sufficiently small.

Proof: First we note that φ is an approximation of φ^*. Ideally we would like to approximate the mapped target function $f^*(\mathbf{y}) = f^*(\varphi^*(\mathbf{x})) = f(\mathbf{x})$ in the low-dimensional projection space, but not knowing φ^* we can only approximate $f^{\#}(\mathbf{y}) = f^{\#}(\varphi(\mathbf{x})) = f(\mathbf{x})$ in this space.

The error bounds for neural network approximation of the target function in high- and low-dimensional spaces are $b^d/\sqrt{n}$ and $b^{d'}/\sqrt{n}$, respectively, where n is the number of neurons in both cases, and $b>1$ is a constant depending on the class of the activation functions [8, 16]. Thus we have that

$$\left|f(\mathbf{x})-g(\mathbf{x})\right| \le \frac{b^d}{\sqrt{n}} \tag{13}$$

and

$$\left|f^{\#}(\varphi(\mathbf{x}))-g^{\#}(\varphi(\mathbf{x}))\right| \le \frac{b^{d'}}{\sqrt{n}} \tag{14}$$

From the last two inequalities and Theorem 1 we can derive the bound for the approximation of the unknown f^*, which is

$$\left|f^*(\mathbf{y})-g^{\#}(\mathbf{y})\right| \le \left|f^*(\mathbf{y})-f^{\#}(\mathbf{y})\right| + \left|f^{\#}(\mathbf{y})-g^{\#}(\mathbf{y})\right| \tag{15}$$
$$\le \frac{b^{d'}}{\sqrt{n}} + \max_{|\boldsymbol{\delta}|\le\Delta}\left|f^*(\mathbf{y})-f^*(\mathbf{y}+\boldsymbol{\delta})\right|$$

Considering the assumptions about the target function f we can write further that

$$\max_{|\boldsymbol{\delta}|\le\Delta}\left|f^*(\mathbf{y})-f^*(\mathbf{y}+\boldsymbol{\delta})\right| \le |\boldsymbol{\delta}|\cdot \max_{|\boldsymbol{\delta}|\le\Delta}\left|f^{*'}(\mathbf{y}+\boldsymbol{\delta})\right| \tag{16}$$
$$\le \Delta\cdot\max_{\mathbf{y}\in A}\left|f^{*'}(\mathbf{y})\right| = \sqrt{d'}\cdot C\cdot\varepsilon_D^2\cdot M$$

where A is the bounded part of the low-dimensional space into which the data points are mapped and M is the maximal value of $\left|f^{*'}(\mathbf{y})\right|$ over A.

Thus we have that

$$\left|f^*(\mathbf{y})-g^{\#}(\mathbf{y})\right| \le \frac{b^{d'}}{\sqrt{n}} + \sqrt{d'}\cdot C\cdot\varepsilon_D^2\cdot M \tag{17}$$

where n is the number of hidden neurons in the neural networks, d' is the dimensionality of the low-dimensional projected data, b depends on the nature of the activation functions of the hidden neurons of the neural network, M depends on the approximated target function, C and ε_D are according to Theorem 1: C depends on the true mapping of the data manifold onto the low dimensional space φ^*, and ε_D depends on the distances between the data points selected for the definition of the LLE mapping of the data manifold onto the low-dimensional space.

Comparing the approximation errors for the neural networks defined and trained on the high- and low-dimensional data having the same number of neurons in their single hidden layer we find that the neural networks with low-dimensional data have better approximation performance if

$$\frac{b^{d'}}{\sqrt{n}} + \sqrt{d'}\cdot C\cdot\varepsilon_D^2\cdot M \le \frac{b^d}{\sqrt{n}} \tag{18}$$

This inequality is satisfied if

$$\varepsilon_D \le b^{\frac{d'}{2}}\cdot\sqrt{(b^{d-d'}-1)}\cdot n^{-\frac{1}{4}}\cdot d'^{-\frac{1}{4}}\cdot C^{-\frac{1}{2}}\cdot M^{-\frac{1}{2}} \tag{19}$$

Given that $b>1$ the last inequality is satisfied even for relatively large ε_D values. This is especially true if d is sufficiently large. **Q.e.d.**

Theorem 2 implies that even for relatively sparse samples of the data points that cover the whole data manifold the approximation performance of neural networks trained with the projected low-dimensional data will be better than the approximation performance of neural networks trained with the original high-dimensional data.

This result means that in the context of big data, when using the whole data set to find the low dimensional mapping of the data might be unfeasible due to the size of the data, a relatively small sample of the data should be sufficient to calculate the low-dimensional LLE mapping of data in order to train a neural network approximation of the target function using the projected data. This may be considerably important in the context of very high volume astronomy or medical imaging or industrial sensor network data.

THEOREM 3. The assumptions of Theorem 1 and Theorem 2 are satisfied and $\mathbf{z}_l, l=1,\dots,p$ are a sample of data points from the high dimensional data space that is used to generate the low-dimensional LLE mapping of the original high-dimensional data. Let us consider $g(\mathbf{x})$ the function representing the neural network approximation of the target function following training with low-dimensional projected data. The average approximation error of $g(\mathbf{x})$ has the tightest bound if the data points $\mathbf{z}_l, l = 1,..., p$ are uniformly distributed.

Proof: The error bound in equation (16) can be improved by using a lower estimate instead of $\Delta \cdot \max_{|\boldsymbol{\delta}|\le\Delta} \left| f^{*'}(\mathbf{y}+\boldsymbol{\delta}) \right|$ by considering the distribution of the sample data points that is used for the generation of the low-dimensional mapping. Let us define $\Delta(\mathbf{y})$ as the local bound on $\left\| \varphi^*(\mathbf{x}) - \varphi(\mathbf{x}) \right\|$ around $\mathbf{y}$ in the part of projection space defined by the projection of the projection generating sample data point that is closest to $\mathbf{y}$ and the projections of the neighbouring projection generating sample data points – note that $\Delta(\mathbf{y}) \le \Delta$. Now we can replace the $\Delta \cdot \max_{|\boldsymbol{\delta}|\le\Delta} \left| f^{*'}(\mathbf{y}+\boldsymbol{\delta}) \right|$ term by the term $\Delta(\mathbf{y}) \cdot \max_{|\boldsymbol{\delta}|\le\Delta(\mathbf{y})} \left| f^{*'}(\mathbf{y}+\boldsymbol{\delta}) \right|$ where $\Delta(\mathbf{y})$ depends on $\mathbf{y}$.

We note that if $\mathbf{y}, \mathbf{y}'$ share their closest projection generating data point then $\Delta(\mathbf{y}) = \Delta(\mathbf{y}')$ and if $\mathbf{z}$ is this closest projection of a projection generating data point then $\Delta(\mathbf{y}) = \Delta(\mathbf{y}') = \Delta(\mathbf{z})$.

Let us consider a set of low-dimensional projections of data points $\mathbf{y}_l, l = 1,..., p$ and the corresponding closest projections of projection generating high-dimensional data points $\mathbf{z}_l, l = 1,..., p$. The average error Err of the approximation of f over $\mathbf{y}_l, l = 1,..., p$ is bounded as follows

$$Err \le \frac{1}{p} \cdot \sum_{l=1}^{p} \left(\frac{b^{d'}}{\sqrt{n}} + \Delta(\mathbf{z}_l) \cdot \max_{\boldsymbol{\delta}\le\Delta(\mathbf{z}_l)} \left| f^{*'}(\mathbf{y}_l+\boldsymbol{\delta}) \right| \right) \quad (20)$$

Denoting $\max_{\boldsymbol{\delta}\le\Delta(\mathbf{z}_l)} \left| f^{*'}(\mathbf{y}_l+\boldsymbol{\delta}) \right|$ as $M(\mathbf{y}_l)$ we get that

$$Err \le \frac{b^{d'}}{\sqrt{n}} + \frac{1}{p} \cdot \sum_{l=1}^{p} \Delta(\mathbf{z}_l) \cdot M(\mathbf{y}_l) \quad (21)$$

Further we have that

$$\left(\sum_{l=1}^{p} \Delta(\mathbf{z}_l) \cdot M(\mathbf{y}_l) \right)^2 \le \left(\sum_{l=1}^{p} \Delta(\mathbf{z}_l)^2 \right) \cdot \left(\sum_{l=1}^{p} M(\mathbf{y}_l)^2 \right) \quad (22)$$

and

$$\sum_{l=1}^{p} \Delta(\mathbf{z}_l)^2 \ge \frac{1}{p} \cdot \left(\sum_{l=1}^{p} \Delta(\mathbf{z}_l) \right)^2 \quad (23)$$

We note that $\sum_{l=1}^{p} \Delta(\mathbf{z}_l)$ is effectively a weighted sum of the distances between the projections of the sample data points used to generate the low dimensional projection of the data manifold. The value of $\sum_{l=1}^{p} \Delta(\mathbf{z}_l)$ is approximately constant for any set of p data points $\mathbf{y}_l, l = 1,..., p$ and it is a multiple of the approximate volume of the part of the data manifold spanned by the data points and the multiplier is proportional to the number of data points for which the approximation error is calculated. Let us denote this value as Δ_0, then we have that

$$\sum_{l=1}^{p} \Delta(\mathbf{z}_l)^2 \ge \frac{1}{p} \cdot \Delta_0^2 \quad (24)$$

and the minimum value of the sum on the left-hand side is achieved for the case when $\Delta(\mathbf{z}_l)$ are equal. This is achieved approximately if the data sample used to generate the low-dimensional projection of the data manifold is uniformly distributed over the manifold ($\Delta(\mathbf{z}_l)$ are equal if the sample projection is perfectly uniform in the sense of being equally spaced).

Thus the error bound for the approximation is the tightest if projection of the data manifold is based on a uniform sample of the data points over the high-dimensional data manifold. In this case the error is bounded as follows

$$Err \le \frac{b^{d'}}{\sqrt{n}} + \left(\frac{1}{p\sqrt{p}} \cdot \Delta_0 + \varepsilon_\Delta \right) \cdot \sqrt{\sum_{l=1}^{p} M(\mathbf{y}_l)^2} \quad (25)$$

where ε_Δ is a small number and $\varepsilon_\Delta = 0$ in the case of equally spaced projected data. **Q.e.d.**

This suggests that low-dimensional manifold projections based on uniform samples of the data manifold allow the best approximation performance by neural networks approximating the target function f over projections of the data points into the low-dimensional space.

The value of Δ_0 depends on the approximation of the volume of the data manifold by the polyhedra determined by the data point sample used to generate the low-dimensional mapping of the manifold and the boundary of the data set within the manifold. For larger projection generation samples this volume approximation gets better and the polyhedral components of approximation get smaller. Consequently, the difference between the two sides of the inequality (24) gets smaller. Thus, we expect that the effect of the sampling distribution is more significant if the projection generating sample of data points is a coarse sample of the data set.

We note that if the measurement of function values is noisy that affects both the high and low dimensional approximation of the function and the impact is higher on the high dimensional approximation (see equations (13) and (14)). Thus, noisy measurement of the function values does not change the above derived theoretical conclusions about the comparison of the approximation performance of neural networks that use the high dimensional data and those that use low dimensional projected data to learn the approximated function.

IV. Application Examples

To test the applicability of the theoretical results we considered high-dimensional data arranged on a 5-dimensional data manifold embedded within a 60-dimensional space. The data manifold is defined as multi-dimensional multiple Swiss roll according to the equations below. The 60-dimensional $\mathbf{x}$ vectors are defined component-wise using 5-dimensional $\mathbf{y}$ vectors as follows

$$\begin{aligned}
\mathbf{x}_{3\cdot(j-1)\cdot(10-j)+6\cdot(k-j-1)+1} &= \mu\cdot\mathbf{y}_j\cdot\cos(\mathbf{y}_j) \\
\mathbf{x}_{3\cdot(j-1)\cdot(10-j)+6\cdot(k-j-1)+2} &= \mu\cdot\mathbf{y}_k \\
\mathbf{x}_{3\cdot(j-1)\cdot(10-j)+6\cdot(k-j-1)+3} &= \mu\cdot\mathbf{y}_j\cdot\sin(\mathbf{y}_j) \\
\mathbf{x}_{3\cdot(j-1)\cdot(10-j)+6\cdot(k-j-1)+4} &= \mu\cdot\mathbf{y}_k\cdot\cos(\mathbf{y}_k) \\
\mathbf{x}_{3\cdot(j-1)\cdot(10-j)+6\cdot(k-j-1)+5} &= \mu\cdot\mathbf{y}_j \\
\mathbf{x}_{3\cdot(j-1)\cdot(10-j)+6\cdot(k-j-1)+6} &= \mu\cdot\mathbf{y}_k\cdot\sin(\mathbf{y}_k)
\end{aligned} \tag{26}$$

where $j < k, j = 1,\ldots,5$ and $k = j+1,\ldots,5$, and

$$\mu = \frac{1}{\sqrt{5}\cdot\|\mathbf{y}\|}\cdot\left(\frac{2}{1+e^{-\|\mathbf{y}\|^2}}-1\right) \tag{27}$$

In fact the equations (26) and (27) above define the function $\varphi^{*-1}(\mathbf{y}) = \mathbf{x}$. Note that the values of $3\cdot(j-1)\cdot(10-j)+6\cdot(k-j-1)+p$ go from 1 to 60 as j goes from 1 to 5, k goes from $j+1$ to 5 and p goes from 1 to 6.

We considered the approximation of ten functions defined on the high-dimensional data (adapted from [9]). The considered functions are defined below using $\mathbf{y} = \varphi^*(\mathbf{x})$.

1) Squared modulus:

$$f_1(\mathbf{x}) = \|\mathbf{y}\|^2 \tag{28}$$

2) Second degree polynomial:

$$f_2(\mathbf{x}) = \frac{1}{500}\cdot\sum_{j=1}^{4}\mathbf{y}_j^2\cdot\mathbf{y}_{j+1} \tag{29}$$

3) Exponential square sum:

$$f_3(\mathbf{x}) = \frac{1}{500}\cdot\sum_{j=1}^{5}e^{\frac{1}{50}\cdot\mathbf{y}_j^2} \tag{30}$$

4) Exponential-sinusoid sum:

$$f_4(\mathbf{x}) = \frac{1}{500}\cdot\sum_{j=1}^{4}e^{\frac{1}{50}\cdot\mathbf{y}_j^2}\cdot\sin(\mathbf{y}_{j+1}) + e^{\frac{1}{50}\cdot\mathbf{y}_5^2}\cdot\sin(\mathbf{y}_1) \tag{31}$$

5) Polynomial-sinusoid sum:

$$f_5(\mathbf{x}) = \frac{1}{50000}\cdot\sum_{j=1}^{5}\mathbf{y}_j^2\cdot\cos(j\cdot\mathbf{y}_j) \tag{32}$$

6) Inverse exponential square sum:

$$f_6(\mathbf{x}) = \frac{10}{\sum_{j=1}^{5}e^{\frac{1}{25}\cdot\mathbf{y}_j^2}} \tag{33}$$

7) Sigmoidal:

$$f_7(\mathbf{x}) = \frac{10}{1+e^{-\frac{1}{5}\cdot\sum_{j=1}^{5}\mathbf{y}_j}} \tag{34}$$

8) Gaussian:

$$f_8(\mathbf{x}) = 10\cdot e^{-\frac{1}{100}\cdot\sum_{j=1}^{5}\mathbf{y}_j^2} \tag{35}$$

9) Linear:

$$f_9(\mathbf{x}) = \sum_{j=1}^{5} j\cdot\mathbf{y}_j \tag{36}$$

10) Constant:

$$f_{10}(\mathbf{x}) = 1 \tag{37}$$

In all cases, both for 60- and 5-dimensional data we constructed neural networks with 20 hidden units having Gaussian activation functions with fixed and randomly set parameters (the number of hidden units was chosen to be sufficiently large, but not too large, assuming that we approximate a moderately complicated function). For each approximated function we trained 20 neural networks using 20 different data sets (i.e. the sampling of the data was repeated 20 times resulting in 20 independent samples of the data).

Each data set consisted of 5000 uniformly randomly chosen 60-dimensional data points for training of the neural networks (note that the samples were chosen from an infinite size complete data set, i.e. the full data set is the complete manifold defined by equations (26) and (27)). In addition to these for each data set we considered 400 additional test data

TABLE I
COMPARISON OF THE APPROXIMATION PERFORMANCES OF NEURAL NETWORKS TRAINED WITH LOW- AND HIGH-DIMENSIONAL DATA – AVERAGE VALUES AND STANDARD DEVIATIONS IN BRACKETS. THE LEVEL OF STATISTICAL SIGNIFICANCE OF THE DIFFERENCES BETWEEN AVERAGE VALUES IS INDICATES AS
* - SIGNIFICANT AT P=0.01 LEVEL, ** - SIGNIFICANT AT P=0.001 LEVEL.

Target Function	High-dim data	Low-dim data, LLE with 3000 samples	Low-dim data, LLE with 1000 samples	Low-dim data, LLE with 300 samples
Squared modulus	22,905.28 (2,441.403)	6,665.98 (1,178.058) **	6,117.09 (689.5801) **	9,357.61 (1,443.37) **
Polynomial	107.7742 (19.02001)	8.8760 (0.919657) **	8.4338 (0.654946) **	7.3792 (0.263297) **
Exponential square sum	0.008188 (0.001503)	7.07E-5 (7.06E-6) **	7.33E-5 (7.21E-6) **	0.000185 (3.65E-5) **
Exponential-sinusoid sum	0.009063 (0.002145)	0.000107 (9.23E-6) **	0.000138 (2.46E-5) **	8.66E-5 (4.22E-6) **
Polynomial-sinusoid sum	0.010596 (0.001471)	3.3E-6 (5.62E-7) **	4.5E-6 (6.63E-7) **	2.7E-6 (1.93E-7) **
Inverse exponential square sum	0.375867 (0.04593)	0.129347 (0.016776) **	0.135788 (0.030936) **	0.122131 (0.008952) **
Sigmoidal	200.9935 (37.4462)	16.5594 (1.449868) **	12.09924 (0.84997) **	15.63517 (1.735107) **
Gaussian	9.311568 (1.77583)	2.902362 (0.28782) **	3.33429 (0.727295) *	3.264746 (0.253456) *
Linear	50,244.24 (6,624.208)	2,139.939 (250.5473) **	1,608.612 (112.818) **	1,570.015 (112.0867) **
Constant	0.319902 (0.030275)	0.001936 (0.001386) **	0.0033 (0.002557) **	0.225078 (0.061538)

points that were randomly picked with uniform distribution over the data manifold. For each data set we selected three samples of the training data set for the calculation of the LLE mapping of the data manifold into the 5-dimensional space. The samples had 3000, 1000 and 300 randomly selected points in them. For each data set and for each calculated LLE mapping we trained and tested one neural network for all 10 considered target functions, i.e. we used the same training and testing data and LLE mapping for all target functions for each data set. For the data points not included into the sample used for the calculation of the LLE mapping we used the equations (5) and (6) to calculate the corresponding projected data.

According to our theoretical results it is expected that the neural networks trained with the low-dimensional projected data perform better than neural networks trained with high-dimensional data even if the sample used to calculate the low-dimensional mapping is small. To compare the performances of neural networks we calculated their average performance for each target function over the 20 data sets and also the standard deviations of their performance values. The performance of each network was assessed as their average squared error over the appropriate test data set. To test the statistical significance of the difference between the average performances we used the t-test. The results are presented in Table I.

The results show that the neural networks trained with the low-dimensional data are statistically significantly better than the neural networks trained with high-dimensional data in terms of their approximation performance in all considered cases with the exception of the approximation of the constant function following the calculation of the LLE projection based on 300 data points. The results do not show in general a systematic difference between the approximation performances of the neural networks trained with low-dimensional data as a function of the size of the data sample used to calculate the LLE projection of the data manifold. These together confirm our expectation that even small sample based LLE projections of the data manifold allow much better neural network approximation of the target function using the projected data than the direct neural network approximation of this target function in the original high-dimensional data space.

To assess the role of the distribution of the data points used to generate the low-dimensional mapping of the data manifold we considered normally distributed data over the 60-dimensional manifold defined by equations (26) and (27). We selected from this data first a normally distributed sample and then a uniformly distributed sample to generate the 5-dimensional mapping of the manifold. We repeated this 20 times and we used the same 10 functions that we used previously (equations (28) to (37)). In all cases we used 5000 data points for training, from which we selected the projection generation data point sample, and we used 400 independently generated data points for the test set.

To generate the normally distributed data we used the Box-Müller transform of uniformly distributed data and we set the component-wise standard deviation to be 0.5 in order to generate a relatively peaked normal distribution for the data points. To create the normally distributed sample from the normally distributed training data set, we picked a random selection of the data points. To create the uniformly distributed sample of the data points first we created a set of uniformly distributed points on the data manifold independently from the training data set and then selected the data points from the training data set that were the closest to these uniformly distributed points on the data manifold. For both cases for each training data set we selected a coarse sample of 100 data points for the generation of the low-dimensional projection of the data manifold.

The results are presented in Table II – note that approximation error performances (mean squared error over the test set) are different from the result reported in Table I as both the training and test sample are from a relatively narrow normal distribution over the data manifold. The results show that in all cases, for all approximated functions, the low-dimensional neural network approximation based on the manifold projection using the uniformly distributed selection of the data points performs statistically significantly better than the low-dimensional neural network approximation based on the manifold projection generated using the normally

TABLE II
COMPARISON OF THE APPROXIMATION PERFORMANCES OF NEURAL NETWORKS TRAINED WITH LOW-DIMENSIONAL DATA GENERATED WITH MANIFOLD PROJECTIONS BASED ON NORMAL AND UNIFORMLY DISTRIBUTED DATA SAMPLES (100 DATA POINTS IN BOTH CASES) – 1000 TIMES AVERAGE VALUES AND STANDARD DEVIATIONS IN BRACKETS. THE LEVEL OF STATISTICAL SIGNIFICANCE OF THE DIFFERENCES BETWEEN AVERAGE VALUES IS INDICATES AS * - SIGNIFICANT AT P=0.01 LEVEL, ** - SIGNIFICANT AT P=0.001 LEVEL.

Target Function	Low-dim data, LLE normal sample	Low-dim data, LLE uniform sample
Squared modulus	5,340.236 (469.2243) **	722.8871 (137.7073) **
Polynomial	0.404295 (0.055482) **	0.011856 (0.00202) **
Exponential square sum	0.45004 (0.092735) **	0.002446 (0.000397) **
Exponential-sinusoid sum	0.430052 (0.046063) **	0.005054 (0.000554) **
Polynomial-sinusoid sum	0.411339 (0.078397) **	0.00267 (0.000683) **
Inverse exponential square sum	5.695947 (1.298511) **	0.312307 (0.071051) **
Sigmoidal	1,428.417 (91.00933) **	201.0943 (19.76358) **
Gaussian	132.0175 (26.31923) **	71.25482 (56.58054) **
Linear	68,235.93 (7,675.206) **	11,037.84 (1,152.381) **
Constant	3.359855 (1.582305) **	0.031211 (0.012912) **

TABLE III
COMPARISON OF THE APPROXIMATION PERFORMANCES OF NEURAL NETWORKS TRAINED WITH LOW- AND HIGH-DIMENSIONAL DATA WITH NOISE – AVERAGE VALUES AND STANDARD DEVIATIONS IN BRACKETS. THE LEVEL OF STATISTICAL SIGNIFICANCE OF THE DIFFERENCES BETWEEN AVERAGE VALUES IS INDICATES AS * - SIGNIFICANT AT P=0.05 LEVEL, ** - SIGNIFICANT AT P=0.01 LEVEL.

Target Function	High-dim data with low noise	Low dim data with low noise	High-dim data with high noise	Low dim data with high noise
Squared modulus	14,556.92 (2,556.108)	6,993.992 (645.6014) **	16,685.87 (3083.226)	8289.357 (710.392) *
Polynomial	75.74326 (22.91223)	9.124691 (1.196528) **	90.25484 (24.16676)	9.301827 (1.095887) **
Exponential square sum	0.009796 (0.002308)	9.05E-5 (1.28E-5) **	0.004792 (0.001901)	0.000109 (1.41E-5) *
Exponential-sinusoid sum	0.008187 (0.00195)	0.000132 (3.59E-5) **	0.012149 (0.002561)	0.000128 (2.88E-5) **
Polynomial-sinusoid sum	0.01445 (0.003686)	4.6E-6 (6.3E-7) **	0.009811 0.003173	4.15E-6 (3.5E-7) **
Inverse exponential square sum	0.681158 (0.139765)	0.169217 (0.036676) **	0.4979 (0.136227)	0.156298 (0.022124) *
Sigmoidal	377.2754 (105.752)	10.77917 (1.015859) **	370.0072 (95.63658)	12.92726 (1.570246) **
Gaussian	9.62391 (2.411471)	3.88002 (0.59562) *	7.173484 (1.860447)	4.06306 (0.647155)
Linear	42,620.46 (13,968.38)	1,470.214 (107.9499) **	36888.25 (10,584.89)	1,502.074 (116.2073) **
Constant	0.253313 (0.05263)	0.02914 (0.021538) **	0.305002 (0.053929)	0.061983 (0.022482) **

distributed selection of data points. This confirms our expectation that for the purpose of low-dimensional neural network approximation of the target function it is preferred to use manifold projections based on uniformly distributed sample of the training data.

We also considered the addition of noise to the sampled values of the target functions. We added low and high level of noise to the function values (i.e. the noise was set to be 10% and 30% of the function values, respectively). In all cases we used LLE projections calculated with 1,000 data points. The results are presented in Table III. The results show that the neural networks trained with low-dimensional projected data statistically significantly outperform the neural networks trained with high dimensional data in all cases in the presence of noise, with the exception of the approximation of the Gaussian function in the presence of high noise.

Finally, to show the application of our results to real world data we considered the MNIST hand-written digits data set. In this case the function is defined on a 784-dimensional space and the sample data points from this data space are images with 784 pixels, each pixel having an integer value between 0 and 255. The function values are defined as the value of the digit corresponding to each image divided by 10 (i.e. the function values are 0, 0.1,..., 0.9). If the approximated function values are completely random the expected squared error of the approximation is 0.165.

We calculated low-dimensional LLE mappings of the data manifold for a range of dimension values: 4 – 8, 10, 12, 15, 20, 30, 50, 70 and 90. We found that the higher dimension values for the low-dimensional mappings (i.e. above 20) did not lead to the improved approximation performance. In general, we found that some low-dimensional projections of the data manifold lead consistently to high approximation error. After analysing the approximation results for neural networks working with low dimensional projected data we concluded to eliminate all cases of projections where a validation step with unseen data leads to an excessively high error (i.e. above 0.25 for dimensions above 4 and 0.35 for projection dimension 4 – the values were determined by analysing the distribution of the validation errors – see Figure 1). We note that the approximation error of the neural networks working with high dimensional data was below 0.22 with the exception of a single case.

We considered for approximation performance evaluation neural networks working with low dimensional data that use low dimensional projections that passed the validation phase (i.e. validation error is below 0.25 or 0.35 in the case of 4-dimensional projections). We found that these neural networks have better approximation performance in general than neural networks working with high dimensional MNIST data. However, the results show that difference in performance is statistically significant only in the case of neural networks working with 5-, 7- and 12-dimensional projection data and it

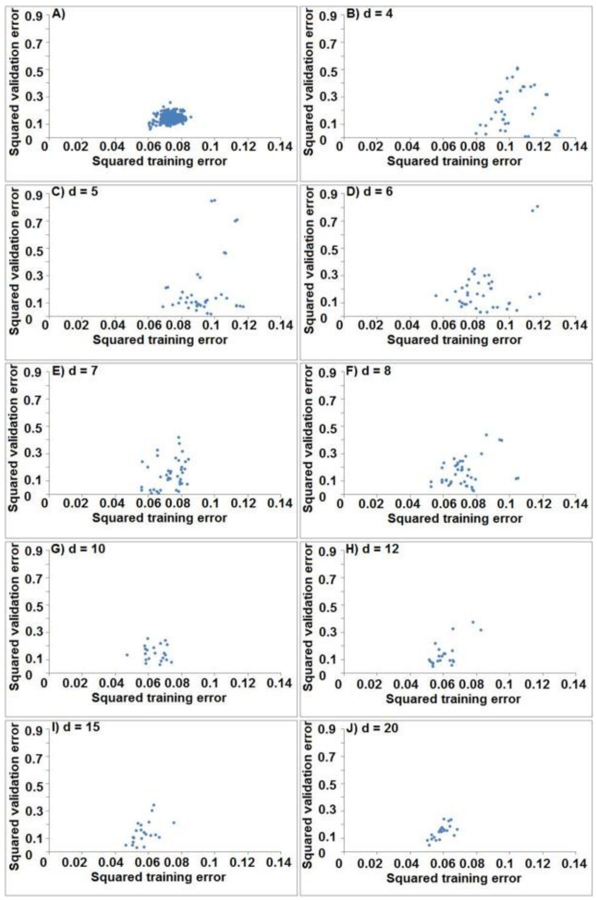

Fig. 1. The distribution of paired values of corresponding training and validation errors for neural networks trained to approximate the function based on the MNIST data using low-dimensional projected data. The dimensionality of the projected data is shown in the figure panels A) – J).

TABLE IV
APPROXIMATION PERFORMANCE OF NEURAL NETWORKS TRAINED WITH HIGH- AND LOW-DIMENSIONAL DATA TO APPROXIMATE THE FUNCTION DEFINED USING THE MNIST DATA SET – AVERAGE VALUES AND STANDARD DEVIATIONS IN BRACKETS. THE LEVEL OF STATISTICAL SIGNIFICANCE OF THE DIFFERENCES BETWEEN AVERAGE VALUES IS INDICATES AS * - SIGNIFICANT AT P=0.05 LEVEL, ** - SIGNIFICANT AT P=0.01 LEVEL.

Projection dimension	Squared test error	p-value
d = 784 (no projection)	0.143939 (0.0027007)	NA
d = 4	0.161158 (0.116984)	0.397788
d = 5	0.103136 (0.04615)	0.000135**
d = 6	0.131836 (0.063457)	0.148549
d = 7	0.113415 (0.072498)	0.018475*
d = 8	0.131477 (0.063865)	0.246844
d = 10	0.140792 (0.053852)	0.422166
d = 12	0.106413 (0.046649)	0.001647**
d = 15	0.130754 (0.072505)	0.980903
d = 20	0.14672 (0.054586)	0.321735

is most significantly different for 5-dimensional data. The results of the comparison of neural network approximations of the real data function defined using the MNIST data are shown in Table IV. We note that the approximation error of neural networks using low-dimensional data with increasing dimensionality got lowered, while their validation and test error did not improve in general (an exception is the 12-dimensional projection). This suggests that increasing the dimensionality of the projected data leads to capturing more noise through the learning process of the neural networks.

V. CONCLUSIONS

We analysed in this paper the approximation error of neural networks built to approximate target functions defined on high-dimensional data, but using low-dimensional projected data derived by projecting the high-dimensional data manifold onto a low-dimensional space. Our analysis shows that the approximation error is dominated by the term that depends exponentially on the dimensionality of the data. This implies that even small samples of the data are sufficient to construct a sufficiently good low-dimensional LLE mapping of the data manifold in order to get much better neural network approximation performance using the projected data than the performance of neural networks trained to approximate the target function using the original high-dimensional data.

This result is important in the context large volumes of high-dimensional data that characterise ‘big data’ problems. In such cases a sufficiently good low-dimensional mapping should be obtainable using a relatively sparse sample of the full data set in order to get good low-dimensional neural network approximations of functions defined over the original high-dimensional data. The application examples presented in the paper provide strong support for this expectation.

Our work also shows that it is important to have as much as possible a uniformly distributed sample of the data manifold for the generation of the low-dimensional mapping of the manifold. This is especially true in the case when coarse samples of the data manifold are used. Using of coarse samples for the manifold mapping is very likely in the case of ‘big data’ data sets for which the storage of the data set in itself may represent a technical problem due to the volume of the data. The examples presented in the paper in this respect support strongly this claim.

Further work is planned to analyse the extent of preservation of properties of the target function by its neural network approximation built using the projected low-dimensional data. For example, the extent to which local maxima and minima are preserved and the accuracy of preservation of these and also the extent of approximation of derivatives and integrals of the target function by the low-dimensional neural network approximation of the target function.

Future work is also planned to compare a range of dimension reduction techniques in the context of approximation of functions on low dimensional projections of the original high dimensional data.

REFERENCES

[1] A.-M. Zhou, K.D. Kumar, Z.-G. Hou and X. Liu “Finite-time altitude tracking control for spacecraft using terminal sliding mode and Chebyshev neural network”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.41, pp.950-963, 2011.
[2] A.Y. Chervonenkis, Problems of machine learning. *LNCS 6744*, pp.21-23. Springer, 2011.
[3] J.H. Friedman, “An overview of predictive learning and function approximation”, *NATO ASI Series F Computer and System Science* vol.136, 1994.
[4] S. Haykin, *Neural Networks and Learning Machines*. Prentice Hall, 2008.
[5] G.-B. Huang, P. Saratchandran and N. Sundararajan “A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation”, *IEEE Transactions on Neural Networks*, vol.16, pp.57-67, 2005.
[6] I.M. Johnstone and D.M. Titterington, “Statistical challenges of high-dimensional data”, *Philosophical Transactions of The Royal Society A*, vol.367, pp.4237-4253, 2009.
[7] J.H. Friedman, “On bias, variance, 0/1 – loss and the curse-of-dimensionality”, *Data Mining and Knowledge Discovery*, vol.1, pp.55-77, 1997.
[8] P. Niyogi and F. Girosi, “Generalization bounds for function approximation from scattered noisy data”, *Advances in Computational Mathematics*, vol.10, pp.51-80, 1999.
[9] P. Andras, “Function approximation using combined unsupervised and supervised learning”, *IEEE Transactions on Neural Networks and Learning Systems*, vol.25,pp.495-505, 2014.
[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.
[11] S. Roweis and L. Saul, “Nonlinear dimensionality reduction by locally linear embedding”, *Science*, vol.290, pp.2323-2326, 2000.
[12] K. Yu, T. Zhang and Y. Gong, “Nonlinear learning using local coordinate coding”, *Advances in Neural Information Processing Systems – NIPS 22*, pp.2223-2231.
[13] H. Yin, “On multidimensional scaling and the embedding of the self-organising maps”, *Neural Networks*, vol.21, pp.160-169, 2008.
[14] K. Hornik, “Multilayer feedforward networks are universal approximators”, *Neural Networks*, vol.2, pp.183-192, 1989.

[15] M.B. Stinchcombe, "Neural networks approximation of continuous functional and continuous functions on compactifications", *Neural Networks*, vol.12, pp.467-477, 1999.

[16] A.R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function", *IEEE Transactions on Information Theory*,vol. 39, pp.930-945, 1993.

[17] G. Gnecco, V. Kurkova, and M. Sanguineti, "Some comparisons of complexity in dictionary-based and linear computational models", *Neural Networks*, vol.24, pp.171-182, 2011.

[18] F. Camastra, "Data dimensionality estimation methods: a survey", *Pattern Recognition*, vol.36, pp.2945-2954, 2003.

Peter Andras (M'95–SM'10) has a BSc in computer science (1995), an MSc in artificial intelligence (1996) and a PhD in mathematical analysis of neural networks (2000), all from the Babes-Bolyai University, Cluj, Romania.

He is a Professor in the School of Computing and Mathematics, Keele University, UK. He has published 2 books and over 100 papers. He works in the areas of complex systems, computational intelligence and computational neuroscience.

Dr. Andras is member of the International Neural Network Society, of the Society for Artificial Intelligence and Simulation of Behaviour, and fellow of the Royal Society of Biology.