KI - Künstliche Intelligenz manuscript No.
(will be inserted by the editor)

Neural Networks for Complex Data

Marie Cottrell · Madalina Olteanu · Fabrice Rossi · Joseph Rynkiewicz · Nathalie Villa-Vialaneix

Received: date / Accepted: date

Abstract Artificial neural networks are simple and efficient machine learning tools. Defined originally in the traditional setting of simple vector data, neural network models have evolved to address more and more difficulties of complex real world problems, ranging from time evolving data to sophisticated data structures such as graphs and functions. This paper summarizes advances on those themes from the last decade, with a focus on results obtained by members of the SAMM team of Université Paris 1.

1 Introduction

In many real world applications of machine learning and related techniques, the raw data are not anymore in a standard and simple tabular format in which each object is described by a common and fixed set of numerical attributes. This standard vector model, while useful and efficient, has some obvious limitations: it is limited to numerical attributes, it cannot handle objects with non uniform descriptions (e.g., situations in which some objects have a richer description than others), relations between objects (e.g., persons involved in a social network), etc.

In addition, it is quite common for real world applications to have some dynamic aspect in the sense that the data under study are the results of a temporal process. Then, the traditional hypothesis of statistical independence between observations does not hold anymore: new hypothesis and theoretical analysis are needed to justify the mathematical soundness of the machine learning methods in this context.

Équipe SAMM, EA 4543
Université Paris I Panthéon-Sorbonne
90, rue de Tolbiac
75634 Paris cedex 13 France
E-mail: Firstname.Lastname@univ-paris1.fr

Artificial neural networks provide some of the most efficient techniques for machine learning and data mining [43]. As other solutions, they were mainly developed to handle vector data and analyzed theoretically in the context of statistically independent observations. However, the last decade has seen numerous efforts to overcome those two limitations [21]. We survey in this article some of the resulting solutions. We will focus our attention on the two major artificial neural network models: the Multi-Layer Perceptron (MLP) and the Self-Organizing Map (SOM).

2 Multi-Layer Perceptrons

The Multi-Layer Perceptron (MLP) is one the most well known artificial neural network model (see e.g., [5]). On a statistical point of view, MLP can be considered as a parametric family of regression functions. Technically, if the data set consists in vector observations in $\mathbb{R}^p$, that is if each object is described by a vector $x = (x_1, \cdots, x_p)^T$, the output of a one hidden layer perceptron with k hidden neurons is given by

$$F_\theta(x) = \beta + \sum_{i=1}^{k} a_i \psi\left(w_i^T x + b_i\right), \tag{1}$$

where the w_i are vectors of $\mathbb{R}^p$, and the β, a_i and b_i are real numbers (θ denote the vector of all parameters obtained by concatenating the w_i, a_i and b_i). In this equation, ψ is a bounded transfer function which introduces some non linearity in F_θ. Given a set of training examples, that is N pairs (X_i, Y_i), the learning process consists in minimizing over θ a distance between Y_i the target value and $F_\theta(X_i)$ the predicted value. Given an error criterion (such as the mean squared error), an optimal value for θ is determined by any optimization algorithm (such as quasi Newton methods see e.g. [7]), leveraging the well know backpropagation algorithm [62] which enables a fast computation of the derivatives of F with

respect to θ. The use of a one hidden layer perceptron model is motivated by approximation results such as [26] and by learnability results such as [63] (in statistical community, learning is called estimation and learnability consistency).

2.1 Model selection issues for MLP

It is well known since the seminal paper of [31] that MLP are an efficient solution for modeling time series whenever the linear model proves to be inadequate. The simplest approach consists in building a non linear auto-regressive model: given a real valued time series $(Y_t)_{t\in\mathbb{N}}$, one builds training pairs $Z_t = (U_t, Y_t)$, where U_t is a vector in $\mathbb{R}^p$ defined by $U_t = (Y_{t-1}, \cdots, Y_{t-p})$). Then a MLP is used to learn the mapping between the U_t (the past of the time series in a time window of length p) and Y_t (the current value of the time series), as in any regression problem.

In order to avoid overlearning and/or large computation time, the question of selecting the correct number of neurons or, more generally, the question of model selection arises immediately. Standard methods used by the neural-networks community are based on pruning: one trains a possibly too large MLP and then removes useless neurons and/or connection weights. Heuristic solutions include Optimal Brain Damage [32] and Optimal Brain Surgeon [24], but a statistically founded method, SSM (Statistical Stepwise Method), was introduced by [11]. The method relies on the minimization of the Bayesian Information Criterion (BIC). Shortly after, [64] and [55] proved the consistency (almost surely) of BIC in the case of MLPs with one hidden layer. These results, established for time series, allow to generalize the consistency results in [63] for the iid case.

The convergence properties of BIC may be generalized even further. A first extension is given in [53]. The noise is supposed to be Gaussian and the transfer function ψ is supposed to be bounded and three times derivable. Then [53] shows that under some mild hypothesis, the maximum of the likelihood-ratio test statistic (LRTS) converges toward the maximum of the square of a Gaussian process indexed by a class of limit score functions. The theorem establishes the tightness of the likelihood-ratio test statistic and, in particular, the consistency of penalized likelihood criteria such as BIC. Some practical applications of such methods can be found in [38]. The hypothesis on the noise was relaxed in [54]. The noise is no longer supposed to be Gaussian, but only to admit exponential moments. Under this more general assumption, BIC criterion is still consistent (in probability).

On the basis of the theoretical results above, a practical procedure for MLP identification is proposed. For a one hidden layer perceptron with k hidden units, we first introduce

$$T_n(k) = \min_{\theta} \left(E_n(\theta) + a_n(k,\theta) \right),$$

where $E_n(\theta)$ is the mean squared error of the MLP for parameter θ and $a_n(k,\theta)$ is a penalty term. Then we proceed as follows:

1. Determination of the right number of hidden units.
   (a) begin with one hidden unit, compute $T_n(1)$,
   (b) add one hidden unit if $T_n(k+1) \leq T_n(k)$,
   (c) if $T_n(k+1) > T_n(k)$ then stop and keep k hidden units for the model.
2. Prune the weights of the MLP using classical techniques like SSM [11].

Note that the choice of the penalty term $a_n(k,\theta)$ is very important. On simulated data, good results have been reported for $a_n(k,\theta)$ from $a_n(k,\theta) = \frac{E_n(\theta)\log(n)}{n}$ to $a_n(k,\theta) = \frac{E_n(\theta)\sqrt(n)}{n}$ (see [55], [52]).

Let us also mention that the tightness of the LRTS and, in particular, the consistency of the BIC criterion were recently established for more complex neural-networks models such as mixtures of MLPs [41] and mixtures of experts [42].

2.2 Modeling and forecasting nonstationary time series

As mentioned in the previous section, MLP are a useful tool for modeling time series. However, most of the results cited above are available for iid data or for stationary time series. In order to deal with highly nonlinear or nonstationary time series, a hybrid model involving hidden Markov models (HMM) and multilayer perceptrons (MLP hereafter) was proposed in [50]. Let us consider $(X_t)_{t\in\mathbb{N}}$ a homogeneous Markov chain valued in a finite state-space $\mathbb{E} = \{e_1, \cdots, e_N\}$ and $(Y_t)_{t\in\mathbb{N}}$ the observed time series. The hybrid HMM/MLP model can be written as follows:

$$Y_{t+1} = F_{X_{t+1}}(Y_t, \cdots, Y_{t-p+1}) + \sigma_{X_{t+1}} \varepsilon_{t+1}, \tag{2}$$

where $F_{X_{t+1}} \in \{F_{e_1}, \cdots, F_{e_N}\}$ is a regression function of order p. In this case, F_{e_i} is the i-th MLP of the model, parameterized by the weight vector w_i. $\sigma_{X_{t+1}} \in \{\sigma_{e_1}, \cdots, \sigma_{e_N}\}$ is a strictly positive number and $(\varepsilon_t)_{t\in\mathbb{N}}$ is a iid sequence of standard Gaussian variables.

The estimating procedure as well as the statistical properties of the parameter estimates were established in [51]. The proposed model was successfully applied in modeling difficult data sets such as ozone peaks [14] or financial shocks [37].

2.3 Functional data

The original MLP model is limited to vector data for an obvious reason: each neuron computes its output as a non linear transformation ψ applied to a (shifted) inner product $w^T x + b$ (see equation (1)). However, as first pointed out in

[56], this general formula applies to any data space on which linear forms can be defined: give a data space $\mathscr{X}$ and a set of linear functions $\mathscr{W}$ from $\mathscr{X}$ to $\mathbb{R}$, one can define a general neuron with the help of $w \in \mathscr{W}$, as calculating $\psi(w(x)+b)$.

This generalization is particularly suitable for functional data, that is for data in which each object is described by one or several functions [45]. This type of data is quite common for instance in multiple time series setting (where each object under study evolves through time and is described by the temporal evolutions of its characteristics) or in spectrometry. A functional neuron [46] can then be defined as calculating $\psi\left(b+\int fw\,\mathrm{d}\mu\right)$, where f is the observed function and w is a parameter function. Results in [46] show that MLP based on this type of neurons share many of the interesting properties of classical MLP, from the universal approximation to statistical consistency (see also [47] for an alternative functional neuron with similar properties). In addition, the parameter functions w can be represented by standard numerical MLP, leading to a hierarchical solution in which a top level MLP for functional data is obtained by using a numerical MLP in each of its functional neurons. Experimental results in [46, 48] show the practical relevance of this technique.

3 Self-Organizing Maps

As the MLP, Kohonen's Self-Organizing Map (SOM) is one of the most well known artificial neural network model [29]. The SOM is a clustering and visualization model in which a set of vector observations in $\mathbb{R}^p$ is mapped to set of M neurons organized in a low dimensional prior structure, mainly a two dimensional grid or a one dimensional string. Each neuron c is associated to a codebook vector p_c in $\mathbb{R}^p$ (p_c is also called a prototype). As in all prototype based clustering methods, each p_c represents the data points that have been assigned to the corresponding neuron, in the sense that p_c is close to those points (according to the Euclidean distance in $\mathbb{R}^p$). The distinctive feature of the SOM is that each prototype p_c is also somewhat representative of data points assigned to other neurons, based on the geometry of the prior structure: if neurons c and d are neighbours in the prior structure, then p_c will be close to data points assigned to neuron d (and vice versa). On the contrary, if c and d are far away from each other in the prior structure, the data points assigned to one neuron will not influence the prototype of the other neuron. This has some very important consequences in terms of visualization capabilities, as illustrated in [60] for instance.

The original SOM algorithm has been designed for vector data, but numerous adaptations to more complex data have been proposed. We survey here three specific extensions, respectively to time series, functional data and categorical data. Another important extension not covered here is proposed in [22] which is built upon processing of multiple time series with recursive versions of the SOM. The authors show that trees and graphs can be clustered by those versions of the SOM, using a temporal coding of the structure. Recent advances in this line of research include e.g. [19]. Other specific adaptation include the symbol strings SOM described in [59].

3.1 Time series with metadata

While the SOM is a clustering algorithm, it has been used frequently in supervised context as a component of a complex model. We described briefly here one such model as an example of complex time series processing with the SOM. Let us consider a time series with two time scales, i.e., that can be written down with two subscripts. The date is denoted by (j,h) where j represents the slow time scale and corresponds for instance to the day (or month or year) while $h = 1,\ldots,H$ corresponds to the observed values (e.g. the hours or half-hours of the day, the days of the month, the months of the year, etc.). Then the time series is denoted $(c_j)_{j\geq 0} = \left((c_{j,1},\ldots,c_{j,H})\right)_{j\geq 0}$. We assume in addition that the slow time scale is associated with metadata. For instance, if each j corresponds to a day in a year and one knows the day of the week, the month, etc. Metadata are supposed to be available prior a prediction.

The original time series $c_{j,h}$ takes value in $\mathbb{R}$, but the dual time scale leads naturally to a vector valued time series representation, that is to the $c_j \in \mathbb{R}^H$. In this point of view, given the past of the vector valued time series, one has to predict a future vector value, that is a complete vector of H values. This could be seen as a long term forecasting problem for which a usual solution would be to iterate one-step ahead forecasts. However, this leads generally to unsatisfactory solutions either because of a squashing behaviour (convergence of the forecasting to the mean value of series) or to a chaotic behaviour (for nonlinear methods).

An alternative solution is explored in [12]. It consists in forecasting separately, on the one hand, the mean and variance of the time series on next slow time scale step (that is, on the next j), and on the other hand, the *profile* of the fast time scale. The prediction of the mean and of the variance is done by any classical technique. For the profile, a SOM is used as follows. The vector values of the time series, i.e., the $(c_j)_{j\geq 0}$, are centred and normalized with respect to the fast time scale, that is are transformed into profiles defined by

$$q_j = \frac{1}{\sigma_j}\left((c_{j,1}-\mu_j,\ldots,c_{j,H}-\mu_j)\right), \tag{3}$$

where $\mu_j = \frac{1}{H}\sum_{h=1}^{H} c_{j,h}$ and $\sigma_j^2 = \frac{1}{H}\sum_{h=1}^{H}(c_{j,h}-\mu_j)^2$ are respectively the mean and the variance of c_j. The profiles are clustered with a SOM leading to some prototype profiles p_c. Each prototype is associated to the metadata of the profiles that has been assigned to the corresponding neuron.

Then a vector value is predicted as follows: the mean μ and variance σ are obtained by a standard forecasting model

for the slow time scale. Then the metadata of the vector to predict is matched against the metadata associated to neurons: assume for instance, that metadata are days of the week, and the we try to predict a Sunday. Then one collects all the neurons to which Sunday profiles have been assigned. Finally, a weighted average of the matching prototypes is computed and rescaled according to μ and σ. As shown in [12] this technique enables both some stable and meaningful full day predictions, while integrating non numerical metadata.

3.2 Functional data

The dual time scale approach described in the previous section has become a standard way of dealing with time series in a functional way, as shown in e.g. [4]. But as pointed out in Section 2.3, functional data arise naturally in other contexts such as spectrometry. Then, the SOM has been naturally adapted to functional data in other contexts than time series. In those contexts, in addition to the normalization technique described above that produces profiles, one can use functional transformation such as derivative calculations in order to drive the clustering process by the shapes of the functions rather than mainly by their average values [49].

Another adaptation consists in integrating the SOM with optimal segmentation techniques that represent functions or time series with simple models, such as piecewise constant functions for instance. The main idea it to a apply a SOM to functional data using any functional distance (from the L^2 norm to more advanced Sobolev norms [61]) with an additional constraint that prototypes must be simple, e.g., piecewise constant. This leads to interesting visualization capabilities in which the complexity of the display is automatically globally adjusted [25].

3.3 Categorical data

In surveys, it is quite standard that the collected answers are categorical variables with a finite number of possible values. In this case, a specific adaptation of the SOM algorithm can be defined, in the same way that Multiple Correspondence Analysis is related to Principal Component Analysis. More precisely, useful encoding methods for categorical data are the Burt Table (BT), which is the full contingency table between all pairs of categories of the variables, or the Complete Disjunctive Table (CDT), that contains the answers of each individual coded as 0/1 against dummy variables that correspond to all the categories of all variables. Then, a Multiple Correspondence Analysis of the BT or of the CDT is nothing else than a Principal Component Analysis on BT or CDT, previously transformed to take into account a specific distance between the rows and a weighting of the individuals [33]. The SOM can be adapted to categorical data using this approach, as described in [10] and [13]. The same transformation on BT or CDT is achieved and a SOM using the rows of the transformed tables can thus be trained. This training provides an organized clustering of all the possible values of the categorical variables on a prior structure such as a two dimensional grid. Moreover, if a simultaneous representation of the individuals and of the values is needed, two coupled SOM can be trained and superimposed. The aforementioned articles present various real-world use cases from socio-economic field.

4 Kernel and dissimilarity SOM

The extensions of artificial neural networks model described in the previous sections are *ad hoc* in the sense that they are constructed using specific features of the data at hand. This is a strength but also a limitation as they are not universal: given a new data type, one has to design a new adaptation of the general technique. In the present section, we present more general versions of the SOM that are based on a dissimilarity or a kernel on the input data. Assuming the existence of such a measure is far weaker than assuming the data are in a vector format. For instance, it is simple to define a dissimilarity/similarity between the vertices of a graph, a data structure that is very frequent in real world problems [40], while representing directly those vertices as vectors is generally difficult.

4.1 Dissimilarity SOM

Let us assume that the data under study belong to a set $\mathscr{X}$ on which a dissimilarity d is defined: d is a function from $\mathscr{X} \times \mathscr{X}$ to $\mathbb{R}^+$ that maps a pair of objects x and y to a non negative real number which measures how different x and y are. Hypothesis on d are minimal: it has to by symmetric ($d(x,y) = d(y,x)$) and such that $d(x,x) = 0$.

As pointed out above, dissimilarities are readily available on sets of non vector data. A classical example is the string edit distance [34] which defines a distance[1] on symbol strings. More general edit distances can be defined, such as for instance the graph edit distance which measure distances between graphs [8].

As the hypothesis on $\mathscr{X}$ are minimal, one cannot assume anymore that vector calculation are possible in this set. Then, the learning rules of the SOM do not apply as they are based on linear combination of the prototypes with the data points. To circumvent this difficulty, [28] suggest to chose the values of the prototypes p_c in the set of observations $(X_i)_i$. This leads to a batch version of the SOM which proceeds

[1] A distance is a dissimilarity that satisfies in addition the strong hypothesis of the triangle inequality: $d(x,y) \leq d(x,z) + d(z,y)$.

as follows. After a random initialization of the prototypes, each observation is assigned to the neuron with the closest propotype (according to the dissimilarity measure) and the prototypes are then updated. For each neuron, the updated p_c is chosen among the observations as the minimizer the following distortion

$$\sum_i \Gamma(N(X_i), c) d(X_i, p) \tag{4}$$

where $N(X_i)$ is X_i's neuron and Γ is a decreasing function of the distance between neurons in the prior structure. This modification of the SOM algorithm is known as the *median SOM* and is closely related to the earlier median version of the standard k-means algorithm [27].

In the case where $(X_i)_i$ is a small sample, the constraint to chose the prototypes in the data can be seen as too strong. Then, [15] suggests to associate several prototypes (a given number q) to each neuron. A neuron is represented by a subset of size q from (X_i) and the different steps of the SOM algorithm are modified accordingly. A fast implementation is described in [9].

A successful application of the dissimilarity SOM on real world data concerns school-to-work transitions. In [39], we were interested in identifying career-path typologies, which is a challenging topic for the economists working on the labor market. The data was issued from the "Generation'98" survey by the CEREQ. The data sample contained information about 16040 young people having graduated in 1998 and monitored during 94 months after having left school. The labor-market statuses had nine categories, from permanent contracts to unemployed and including military service, inactivity or higher education.

The dissimilarity matrix was computed using optimal matching distances [1], which are currently the main stream in economy and sociology. The most striking opposition appeared between the career-paths leading to stable-employment situations and the "chaotic" ones. The stable positions were mainly situated in the west region of the map. However, the north and south regions were quite different: in the north-west region, the access to a permanent contract (red) was achieved after a fixed-term contract (orange), while the south-west classes were only subject to transitions through military service (purple) or education (pink). The stability of the career paths was getting worse as we moved to the east of the map. In the north-east region, the initial fixed-term contract was getting longer until becoming precarious, while the south-east region was characterized by the excluding trajectories: unemployment (light blue) and inactivity (dark blue).

Two other extensions of the SOM to dissimilarity data have been proposed; they both avoid the use of constrained prototypes. The oldest one is based on deterministic annealing [18] while a more recent one uses the so-called relational approach that relies on pseudo-Euclidean spaces [20, 23]. Both approaches lead to better results for datasets where the ratio between the number of observations and the number of neurons is small.

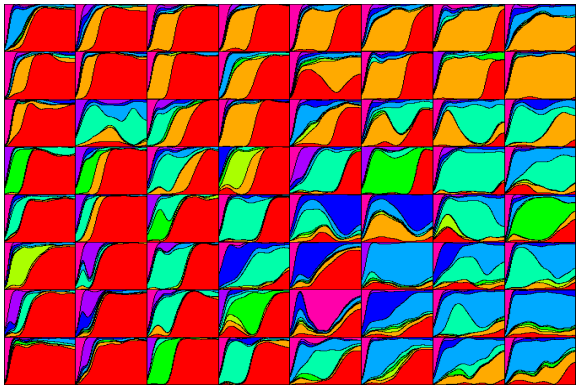

Fig. 1 Career-path visualization with the dissimilarity SOM [39]: colors correspond to the nine different categories

4.2 Kernel SOM

An alternative approach to dissimilarities is to rely on kernels. Kernels can be seen as a generalization of the notion of similarity. More precisely, a kernel on a set $\mathscr{X}$ is a symmetric function K from $\mathscr{X} \times \mathscr{X}$ to $\mathbb{R}$ that satisfies a positivity property:

$$\forall N \in \mathbb{N}^*,\ \forall (x_i)_{1\leq i\leq N} \in \mathscr{X}^N,\ \forall (\alpha_i)_{1\leq i\leq N} \in \mathbb{R}^N,$$
$$\sum_{i,j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) \geq 0.$$

For such a kernel, there is a Hilbert space $\mathscr{H}$ (called the feature space of the kernel) and a mapping ϕ from $\mathscr{X}$, such that the inner product in $\mathscr{H}$ corresponds to the kernel via the mapping, that is [3] :

$$\langle \phi(x), \phi(x') \rangle_{\mathscr{H}} = K(x, x'). \tag{5}$$

Then K can be interpreted as a similarity on $\mathscr{X}$ (values close to zero correspond to unrelated objects) and defines indirectly a distance between objects in $\mathscr{X}$ as follows:

$$\begin{aligned} d_K(x, x') &= \|\phi(x) - \phi(x')\|_{\mathscr{H}} \\ &= \sqrt{K(x,x) + K(x',x') - 2K(x,x')}. \end{aligned} \tag{6}$$

As shown in e.g. [57], kernels are a very convenient way to extend standard machine learning methods to arbitrary spaces. Indeed, the feature space $\mathscr{H}$ comes with the same elementary operations as $\mathbb{R}^p$: linear combination, inner product, norm and distance. Then, one has just to work in the feature space as if it were the original data space. The only difficulty comes from the fact that ϕ and $\mathscr{H}$ are not explicit in general, mainly

because $\mathscr{H}$ is an infinite dimensional functional space. Then one has to rely on equation (5) to implement a machine learning algorithm in $\mathscr{H}$ completely indirectly using only K. This is the so called *Kernel trick*.

In the case of the batch version of the SOM, this is quite simple [6]. Indeed, assignments of data points to neurons are based on the Euclidean distance in the classical numerical case: this translates directly into the distance in the feature space, which is calculated solely using the kernel (see equation (6)). Prototypes update is performed as weighted averages of all data points: weights are computed with the Γ function introduced in equation (4) as a proxy for the prior structure. It can be shown that those weights, which are computed using the assignments only, are sufficient to define the prototypes and that they can be plugged into the distance calculation, without needing an explicit calculation of ϕ. Variants of this scheme, especially stochastic ones, have been studied in [2, 36]. It should also be noted that the relational approach mentioned in the previous section [20, 23] can be seen a relaxed kernel SOM, that is an application of a similar algorithm in situations where the function K is not positive.

While kernels are very convenient, the positivity conditions might seem very strong at first. It is indeed much stronger than the conditions imposed to a dissimilarity, for instance. Nevertheless, numerous kernels have been defined on complex data [16], ranging from kernels on strings based on substrings [35] to kernel between the vertices of a graph such as the heat kernel [30, 58] (see [6] for a SOM based application of this kernel to a medieval data set of notarial acts). Two graphs can also be compared via a kernel based on random walks [17] or on subtrees comparisons [44].

5 Conclusion

Present days data are becoming more and more complex, according to several criteria: structure (from simple vector data to relational data mixing a network structure with categorical and numerical descriptions), time evolution (from a fixed snapshot of the data to ever changing dynamical data) and volume (from small datasets with a handful of variables and one thousand of objects to terabytes and more datasets). Adapting artificial neural networks to those new data is a continuous challenge which can be solved only by mixing different strategies as outlined in this paper: adding complexity to the models enable to tackle non standard behavior (such as non-stationarity), theoretical guarantees limit the risk of overfitting, new models can be tailor made for some specific data structures such as graph or functions, while generic kernel/dissimilarity models can handle almost any type of data. The ability to combine all those strategies demonstrates once again the flexibility of the artificial neural network paradigm.

References

1. Abbott A, Tsay A (2000) Sequence analysis and optimal matching methods in sociology. Sociological Methods and Research 29(1):3–33
2. Andras P (2002) Kernel-Kohonen networks. International Journal of Neural Systems 12:117–135
3. Aronszajn N (1950) Theory of reproducing kernels. Transactions of the American Mathematical Society 68(3):337–404
4. Besse P, Cardot H, Stephenson D (2000) Autoregressive forecasting of some functional climatic variations. Scandinavian Journal of Statistics 4:673–688
5. Bishop C (1995) Neural Networks for Pattern Recognition. Oxford University Press, New York, USA
6. Boulet R, Jouve B, Rossi F, Villa N (2008) Batch kernel SOM and related laplacian methods for social network analysis. Neurocomputing 71(7–9):1257–1273
7. Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press
8. Bunke H (2003) Graph-based tools for data mining and machine learning. In: Proc. of the International Conference on Machine learning and Data Mining in Pattern Recognition, Springer-Verlag, pp 7–19
9. Conan-Guez B, Rossi F, El Golli A (2006) Fast algorithm and implementation of dissimilarity self-organizing maps. Neural Networks 19(6–7):855–863
10. Cottrell M, Letrémy P (2005) How to use the Kohonen algorithm to simultaneously analyse individuals in a survey. Neurocomputing 63:193–207
11. Cottrell M, Girard B, Girard Y, Mangeas M, Muller C (1995) Neural modeling for time series: a statistical stepwise method for weight elimination. IEEE Transactions on Neural Networks 6(6):1355–1364
12. Cottrell M, Girard B, Rousset P (1998) Forecasting of Curves Using Kohonen Classification. Journal of Forecasting 17(5–6):429–439
13. Cottrell M, Ibbou S, Letrémy P (2004) SOM-based algorithms for qualitative variables. Neural Networks 17:1149–1167
14. Dutot AL, Rynkiewicz J, Steiner F, Rude J (2007) A 24-h forecast of ozone peaks and exceedance levels using neural classifiers and weather predictions. Environmental Modellig and Software 22(9):1261–1269
15. El Golli A, Rossi F, Conan-Guez B, Lechevallier Y (2006) Une adaptation des cartes auto-organisatrices pour des données décrites par un tableau de dissimilarités. Revue de Statistique Appliquée LIV(3):33–64
16. Gärtner T (2008) Kernel for Structured Data. World Scientific
17. Gärtner T, Flach A, Wrobel S (2003) On graph kernels: Hardness a results and efficient alternatives. In: Proc. of the Annual Conference on Computational Learning Theory, pp 129–143
18. Graepel T, Burger M, Obermayer K (1998) Self-organizing maps: generalizations and new optimization techniques. Neurocomputing 21:173–190
19. Hagenbuchner M, Sperduti A, Tsoi AC (2009) Graph self-organizing maps for cyclic and unbounded graphs. Neurocomputing 72(7–9):1419 – 1430
20. Hammer B, Hasenfuss A (2010) Topographic mapping of large dissimilarity data sets. Neural Computation 22(9):2229–2284
21. Hammer B, Jain BJ (2004) Neural methods for non-standard data. In: Proc. of XIIth European Symposium on Artificial Neural Networks (ESANN 2004), Bruges (Belgium), pp 281–292
22. Hammer B, Micheli A, Strickert M, Sperduti A (2004) A general framework for unsupervised processing of structured data. Neurocomputing 57:3–35
23. Hammer B, Hasenfuss A, Rossi F, Strickert M (2007) Topographic processing of relational data. In: Proc. of the 6th Workshop on Self-Organizing Maps (WSOM 07), Bielefeld, Germany

24. Hassibi B, Stork DG, Wolff GJ (1993) Optimal brain surgeon and general network pruning. In: Proc. of the International Conference on Neural Networks, IEEE, pp 293–299
25. Hébrail G, Hugueney B, Lechevallier Y, Rossi F (2010) Exploratory analysis of functional data via clustering and optimal segmentation. Neurocomputing 73(7–9):1125–1141
26. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Networks 2:359–366
27. Kaufman L, Rousseeuw P (1987) Clustering by means of medoids. In: Dodge Y (ed) Statistical Data Analysis Based on the L1-Norm and Related Methods, North-Holland, pp 405–416
28. Kohohen T, Somervuo P (1998) Self-Organizing maps of symbol strings. Neurocomputing 21:19–30
29. Kohonen T (2001) Self-Organizing Maps, Springer Series in Information Sciences, vol 30, 3rd edn. Springer, Berlin, Heidelberg, New York
30. Kondor R, Lafferty J (2002) Diffusion kernels on graphs and other discrete structures. In: Proc. of the 19th International Conference on Machine Learning, pp 315–322
31. Lapedes A, Farber R (1987) Nonlinear signal processing using neural networks: Prediction and system modeling. Signal Processing
32. Le Cun Y, Denker J, Solla S (1990) Optimal brain damage. In: Touretzky DS (ed) Advances in Neural Information Processing Systems (NIPS 1989), vol 2, Morgan Kaufmann, pp 598–605
33. Le Roux B, Rouanet H (2004) Geometric Data Analysis: From Correspondence Analysis to Structured Data Analysis. Springer, Dordrecht
34. Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 6:707–710
35. Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins C (2002) Text classification using string kernels. The Journal of Machine Learning Research 444
36. Mac Donald D, Fyfe C (2000) The kernel self organising map. In: Proc. of 4th International Conference on knowledge-based intelligence engineering systems and applied technologies, pp 317–320
37. Maillet B, Olteanu M, Rynkiewicz J (2004) Nonlinear analysis of shocks when financial markets are subject to changes in regime. In: Proc. of XIIth European Symposium on Artificial Neural Networks (ESANN 2004), pp 87–92
38. Mangeas M (1997) Neural model selection: how to determine the fittest criterion. In: Proc. of ICANN 1997 (artificial neural networks), no. 1327 in Lecture Notes in Computer Science, Springer, Lausanne, Switzerland, pp 987–992
39. Massoni S, Olteanu M, Rynkiewicz J (2009) Career-path analysis using optimal matching and self-organizing maps. In: Advances in Self-Organizing Maps: 7th International Workshop, WSOM 2009, Lecture Notes in Computer Science, Springer, pp 154–162
40. Newman MEJ (2003) The structure and function of complex networks. SIAM Review 45:167–256
41. Olteanu M, Rynkiewicz J (2008) Estimating the number of components in a mixture of multilayer perceptrons. Neurocomputing 71(7–9):1321–1329
42. Olteanu M, Rynkiewicz J (2011) Asymptotic properties of mixture-of-experts models. Neurocomputing 74(9):1444–1449
43. Osowski S, Siwek K, Markiewicz T (2004) Mlp and svm networks - a comparative study. In: proceedings of the 6^{th} Nordic Signal Processing Symposium (NSPS 2004), pp 37–40
44. Ramon J, Gärtner T (2003) Expressivity versus efficiency of graph kernels. In: Proc. of First International Workshop on Mining Graphs, Trees and Sequences (held with ECML/PKDD'03)
45. Ramsay J, Silverman B (1997) Functional Data Analysis. Springer Verlag, New York
46. Rossi F, Conan-Guez B (2005) Functional multi-layer perceptron: a nonlinear tool for functional data anlysis. Neural Networks 18(1):45–60
47. Rossi F, Conan-Guez B (2006) Theoretical properties of projection based multilayer perceptrons with functional inputs. Neural Processing Letters 23(1):55–70
48. Rossi F, Conan-Guez B, Fleuret F (2002) Functional data analysis with multi layer perceptrons. In: IJCNN 2002 (part of WCCI) proceedings, pp 2843–2848
49. Rossi F, Conan-Guez B, El Golli A (2004) Clustering functional data with the SOM algorithm. In: Proc. of XIIth European Symposium on Artificial Neural Networks (ESANN 2004), Bruges (Belgium), pp 305–312
50. Rynkiewicz J (1999) Hybrid hmm/mlp models for time series prediction. In: Proc. of VIIth European Symposium on Artificial Neural Networks (ESANN 1999), Bruges (Belgium), pp 455–462
51. Rynkiewicz J (2001) Estimation of hybrid hmm/mlp models. In: Proc. of IXth European Symposium on Artificial Neural Networks (ESANN 2001), Bruges (Belgium), pp 383–390
52. Rynkiewicz J (2006) Consistent estimation of the architecture of multilayer perceptrons. In: Proc. of the 14^{th} European Symposium on Artificial Neural Networks (ESANN 2006), pp 149–154
53. Rynkiewicz J (2008) Asymptotic law of likelihood ratio for multilayer perceptron models. In: Advances in Neural Networks (5th International Symposium on Neural Networks, ISNN 2008), no. 5263 in Lecture Notes in Computer Science, Springer, Beijing, China, pp 186–195
54. Rynkiewicz J (2012) General bound of overfitting for mlp regression models. Neurocomputing 90:106–110
55. Rynkiewicz J, Cottrell M, Mangeas M, Yao J (2001) Modèles de réseaux de neurones pour l'analyse des séries temporelles ou la régression. Revue d'intelligence artificielle 15(3–4):317–332
56. Sandberg I (1996) Notes on weighted norms and network approximation of functionals. IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications 43(7):600–601
57. Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, Cambridge, UK
58. Smola A, Kondor R (2003) Kernels and regularization on graphs. In: Warmuth M, Schölkopf B (eds) Proc. of the Conference on Learning Theory (COLT) and Kernel Workshop, Lecture Notes in Computer Science, pp 144–158
59. Somervuo P (2004) Online algorithm for the self-organizing map of symbol strings. Neural Network 17:1231–1239
60. Vesanto J (1999) Som-based data visualization methods. Intelligent Data Analysis 3(2):111–126
61. Villmann T (2007) Sobolev metrics for learning of functional data - mathematical and theoretical aspects. In: Machine Learning Reports 1 (MLR-03-2007), pp 1–15, ISSN:1865-3960
62. Werbos PJ (1974) Beyond regession: New tools for prediction and analysis in the behavior sciences. PhD thesis, Harvard University, Cambridge, MA
63. White H (1990) Connectionist nonparametric regression: mutilayer feedforward networks can learn arbitrary mappings. Neural Networks 3:535–549
64. Yao J (2000) On least square estimation for stable nonlinear ar processes. The Annals of Institut of Mathematical Statistics 52:316–331