ESANN'2004 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 313-318

Functional Radial Basis Function Networks (FRBFN)

N. Delannay*[1], F. Rossi[2], B. Conan-Guez[2], M. Verleysen†[1]
[1]Université catholique de Louvain (UCL)
DICE - Machine Learning Group
Place du levant 3, 1348 Louvain-la-Neuve, Belgium
{delannay, verleysen}@dice.ucl.ac.be
[2]INRIA - Projet AxIS - Université Paris-Dauphine
Fabrice.Rossi@dauphine.fr, Brieuc.Conan-Guez@inria.fr

Abstract. There has been recently a lot of interest for functional data analysis [1] and extensions of well-known methods to functional inputs (clustering algorithm [2], non-parametric models [3], MLP [4]). The main motivation of these methods is to benefit from the enforced inner structure of the data. This paper presents how functional data can be used with RBFN, and how the inner structure of the former can help designing the network.

1 Introduction

In a variety of problems, the data at our disposal are samplings of a functional entity. Working with functional data instead of vectorial data can be an efficient way to account for this within-data structure and could help resolving the modeling problem.

In addition, radial basis function networks (RBFN) have shown to be powerful tools to construct non-linear regression models. In this paper, we shall see how the RBFN model can be extended to accept functional data on the input side. Furthermore, we will explain how the functional features can direct the design of the network.

In section 2, functional data are presented and some examples are given. Section 3 recalls what the RBFN is and how it can be designed. The extension to functional data input is shown in section 4. Section 5 applies ideas expressed in this paper on spectrometric data. Section 6 gives concluding comments.

*N.D is Research Fellow of the Belgian National Fund for Scientific Research.
†M.V. is Senior Research Associate of the Belgian National Fund for Scientific Research.

2 Functional data

Consider a set of N observations $\{\mathbf{x_i}\}_{i=1}^{N}$, where the ith observation is itself composed of p_i pairs $(t_i^{(j)}, x_i^{(j)})_{j=1}^{p_i}$. The $t_i^{(j)} \in \mathcal{T}$ may be considered as the argument of corresponding value $x_i^{(j)} \in \Re$, with $\mathcal{T}$ an open set in $\Re^D$. For simplicity, we will assume $\mathcal{T} \subset \Re$. Each observation is a functional entity:

$$x_i : t \in \mathcal{T} \rightarrow x_i(t) \in \Re. \tag{1}$$

The idea behind functional data is to compute the underlying function of each observation and continue to work with the functional expression, instead of the p_i-dimensional vector $\mathbf{x_i}$.

The usual way to handle functional data is to use a basis expansion such that

$$\tilde{x}_i(t) = \sum_{k=1}^{K} \xi_i^{(k)} h_k(t) = \xi_i{}^{\mathrm{T}} \mathbf{h}(\mathbf{t}) \tag{2}$$

where $\mathbf{h}(\mathbf{t}) = [h_1(t) \ldots h_K(t)]^{\mathrm{T}}$ is a set of basis functions defined over $\mathcal{T}$, ξ_i is the vector of coeficients characterizing observation i and $\tilde{x}_i(t)$ is a good approximation of the true function $x_i(t)$. Well-known basis functions are Fourier series, B-splines and wavelets.

Figure 1 shows three examples for which functional data should be used. Original observations are sets of points, but it is obvious that there is a functional entity behind each observation. Using a functional smooth representation is of particular interest when the data are sampled from a smooth function, when the sampling is different between two observations, or when a smooth function is polluted by noise, as illustrated in figures 1(a), 1(b) and 1(c) respectively.

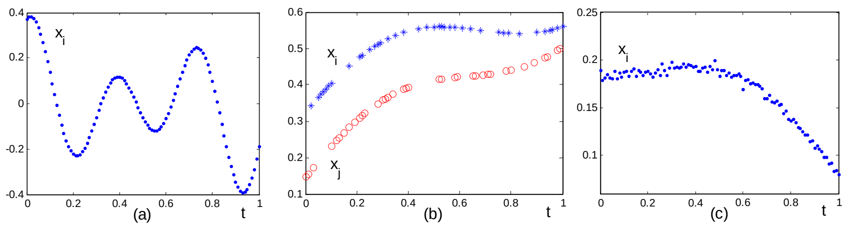

Figure 1: (a) the observation is obviously a smooth function, (b) two observations $\mathbf{x_i}$ and $\mathbf{x_l}$ having different sampling arguments $t_i^{(j)}$ and $t_l^{(j)}$, (c) a sampled curve with additional noise.

3 RBFN model

The general problem of modelling is to estimate the relation between an input variable $\mathbf{x} \in \Re^p$ and an output variable $y \in \Re$ from a set of observed data,

$(\mathbf{x_i}, y_i)_{i=1}^{N}$. With RBFN, the relation has the following form:

$$\tilde{y} = \sum_{m=1}^{M} \lambda_m \Phi_m \left(d_m(\mathbf{x}, \mathbf{c_m}) \right), \tag{3}$$

where $\Phi_m(\cdot)$ are radial basis functions, $\mathbf{c_m}$ are centers belonging to the space of $\mathbf{x}$, $d_m(\cdot,\cdot)$ are associated distances and λ_m are weighting coefficients. According to the universal approximation property of RBFN [5], an adequate choice of these parameters leads to a good approximation $\tilde{y}$ of y. As such, RBFN may be applied to functional data instead of vectorial ones. One has just to define a functional measure of distance $d_m(\cdot,\cdot)$, leading to so-called *Functional RBFN*.

There is too much freedom in such network to tackle directly its design. A way to proceed can be to start by choosing a family of RBFN and testing different definitions of the distance $d_m(.,.)$. Next, the other parameters are computed with a fast algorithm even if it is suboptimal with respect to a global optimization procedure [6]. It should be mentioned that one of the main advantages of RBFN is their simplified training, compared to MLP (Multi Layers Perceptrons) for example. Indeed, centers $\mathbf{c_m}$ and scaling factors σ_m may be computed by an unsupervised learning or a deterministic algorithm. Model learning then becomes a linear problem in coefficients λ_m, leading to low computational cost and avoiding local minima issues. Despite this, the RBFN performs very efficiently if well designed.

A classical choice for the distance is

$$d_m(\mathbf{x}, \mathbf{c_m}) = \frac{1}{\sigma_m} \left((\mathbf{x} - \mathbf{c_m})^{\mathrm{T}} \mathbf{W} (\mathbf{x} - \mathbf{c_m}) \right)^{1/2}, \tag{4}$$

where σ_m is a scaling factor adjusted with the learning algorithm and $\mathbf{W}$ is a positive semi-definite matrix. The usual Euclidian distance is obtained with $\sigma_m = 1$ and $\mathbf{W} = \mathbf{I}$ the identity matrix. It is interesting to note that working with $\mathbf{W} = \mathbf{I}$ on linearly preprocessed vectors, $\mathbf{x}' = \mathbf{A}\mathbf{x}$, $\mathbf{c}'_\mathbf{m} = \mathbf{A}\mathbf{c_m}$, is the same as working on original vectors $\mathbf{x}$ and $\mathbf{c_m}$ with $\mathbf{W} = \mathbf{A}^{\mathrm{T}}\mathbf{A}$. This remark shows that it can be equivalent to think in terms of distance definition or in terms of preprocessing of the original data.

4 RBFN with functional data

Let us consider now that the input variable $\mathbf{x}$ is a function on domain $\mathcal{T}$. As mentioned earlier, it is straightforward to extend the RBFN model (3). The functional extension of (4) is

$$d_m(x(t), c_m(t)) = \frac{1}{\sigma_m} \left(\int_{\mathcal{T}} w(t)(x(t) - c_m(t))^2 dt \right)^{1/2} \tag{5}$$

where $w(t)$ is the functional version of $\mathbf{W}$ when this matrix is diagonal. The extension of non-diagonal weighting matrix $\mathbf{W}$ to functional case would be a convolution integral.

In theory, the scaling function $w(t)$ could be almost any function. Nevertheless, in order to keep the functional character of data in (5), it seems natural to constrain $w(t)$ to exhibit about the same complexity[1] as $x(t)$. Hence, working with functional data can help reducing the excessive freedom of the parameters of the RBFN model.

One could also work with a functional semi-metric [3]. For example,

$$d_m(x(t), c_m(t)) = \frac{1}{\sigma_m}\left(\int_{\mathcal{T}} D\left(x(t) - c_m(t)\right)^2 dt\right)^{1/2} \tag{6}$$

where $D(\cdot)$ is a differential operator.

There is no obvious way to design efficiently this extended network. Once again, the choice of the distance is crucial. As it is equivalent to think in terms of distance measure or data preprocessing, the functional approach may help to find an adequate preprocessing accounting for the inner structure of the data.

An usual way to handle functions is to work with basis expansions. With $x(t) = \xi^{\mathrm{T}}\mathbf{h}(t)$ and $c_m(t) = \gamma_{\mathbf{m}}{}^{\mathrm{T}}\mathbf{h}(t)$ and if $w(t) = 1$, (5) becomes

$$d_m(x(t), c_m(t)) = \frac{1}{\sigma_m}\left((\xi - \gamma_{\mathbf{m}})^{\mathrm{T}}\mathbf{H}(\xi - \gamma_{\mathbf{m}})\right)^{1/2}, \tag{7}$$

where $\mathbf{H} = \int_{\mathcal{T}} \mathbf{h}(t)^{\mathrm{T}}\mathbf{h}(t)dt$. Hence, handling functional RBFN can be done as with any vectorial RBFN on coefficients ξ and $\gamma_{\mathbf{m}}$ if the appropriate matrix $\mathbf{H}$ is used.

5 Application on spectrometric data

In this section, we apply on spectrometric data[2] the ideas developped in this paper. Each observation is a 100-channel spectrum ($\lambda \in [850 - 1050]$ nm) of a meat sample. Note that we used symbol λ for the argument instead of the generic t. The output to be predicted is the sample level of fat.

Spectra are smooth functions. The original vectorial data are transformed to be expressed on a B-spline basis expansion with 30 splines. On Figure 2, four different choices of preprocessing are illustrated and will be used below. These were selected in an exploratory way. Fig. 2(a) shows five original spectra. It is assumed that the mean of each observation does not account much for the output and thus the means of the observations are substracted on Fig. 2(b). To enhance the correlation between inputs and outputs, spectra preprocessed in (b) are scaled with function $a(\lambda)$ (Fig. 2(c)). This function $a(\lambda)$ comes from the mean of the two first components of the partial least square analysis (PLS). The last preprocessing we present here is simply the use of the first derivative of the original spectra (Fig. 2(d)).

The RBFN are constructed with the OLS algorithm [7] and with the distance (7) applied on preprocessed functions. The additional scaling factors

[1]The complexity of a function is often characterized by its second derivative.
[2]Tecator data set: http://lib.stat.cmu.edu/datasets/tecator

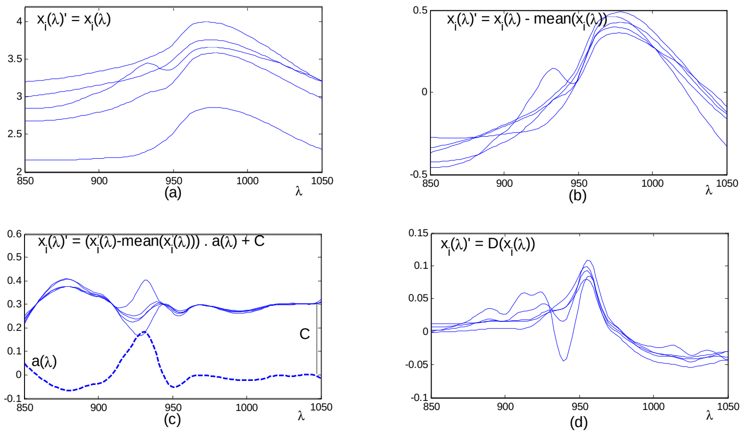

Figure 2: (a) 5 original spectra, (b) the same spectra translated to have a null mean, (c) the translated spectra scaled by function $a(\lambda)$ and (d) the first derivative of original spectra

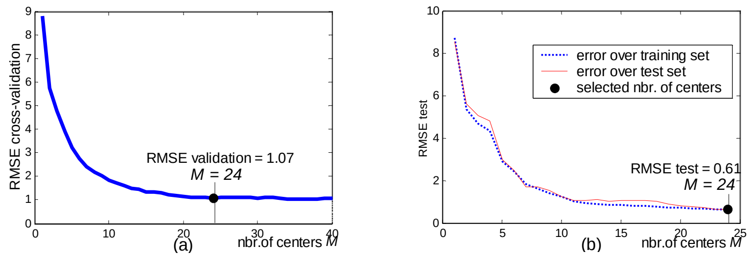

Figure 3: (a) cross-validation scheme to select the number of centers, (b) test error with the selected number of centers.

σ_m are determined heuristically according to the mean of the distances to the neighboring centers. The original data set holds 215 spectra. It is partitioned in two parts: a training set and a test set with respectively 172 and 43 spectra. A cross-validation scheme is applied on the training set[3] to select the number of centers for each network. Finally, the root mean squared error (RMSE) over the test set is calculated to give an idea of the overall perfomances of the network. This procedure is illustrated on Figure 3 for preprocessing (d).

Table 1 summaries the results obtained with the four different preprocessing. We see that:

- Working direcly with the original spectra gives very poor results.
- Applying the scaling function $a(\lambda)$ slightly improves the performances.

[3]each validation set contains 43 of the 172 spectra

- The best performances are obtained with the first derivative of the spectra.

The corresponding test error is not the smallest found in the literature. For example in [8], a test error of 0.36 is reported. But the major advantages of our procedure are its low computational cost and its ability to take the functional character of observations into account in a natural way.

Preprocessing	nbr. of centers	mean validation RMSE	test RMSE
(a)	6	10.2	11.7
(b)	23	2.06	1.64
(c)	18	1.95	1.25
(d)	24	1.07	0.61

Table 1: Results of constructed *F-RBFN*.

6 Conclusion

This paper shows that there are three levels at which functional data can help designing a RBFN. Firstly, it is a convenient way to express data having a functional structure. Secondly, thinking in terms of data preprocessing rather than distance measures, the functional approach helps designing a RBFN taking into account the functional nature of data. And finally, it may be a way to constrain parameters of the general RBFN model.

Further work should try to incorporate these advantages in a more automatic learning algorithm and thus reduce the exploratory part of the current scheme.

References

[1] J.O. Ramsay and B.W. Silverman. *Functional Data Analysis.* Springer-Verlag, Inc., New York, 1997.

[2] C. Abraham and al. Unsupervised curve clustering using b-splines. Technical report, ENSAM-INRIA-UM II-Montpellier, June 2002.

[3] F. Ferraty and P. Vieu. The functional nonparametric model and application to spectrometric data. *Computational Statistics*, 17(4):545–564, 2002.

[4] F. Rossi, B. Conan-Guez, and F. Fleuret. Theoretical properties of functional multi layer perceptrons. In *Proceedings of ESANN 2002*, pages 7–12, Bruges, Belgium, April 2002.

[5] J. Park and I. W. Sandberg. Universal approximation using radial basis function networks. *Neural Computation*, 3(2):246–257, 1991.

[6] F. Schwenker, H. A. Kestler, and G. Palm. Three learning phases for radial-basis-function networks. *Neural Networks*, 14(4-5):439–458, May 2001.

[7] S. Chen, F.N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. on Neural Networks*, 2(2):302–309, 1991.

[8] H. H. Thodberg. A review of bayesian neural networks with an application to near infrared spectroscopy. *IEEE Trans. on Neural Networks*, 7(1):56–72, 1996.