Phoneme Discrimination with Functional Multi-Layer Perceptron[0]

Brieuc Conan-Guez[1,2] and Fabrice Rossi[1,2,3]

[1] Projet AXIS, INRIA-Rocquencourt
Domaine De Voluceau,
BP 105 Bâtiment 18
78153 Le Chesnay Cedex, France
brieuc.conan-guez@inria.fr
[2] CEREMADE, UMR CNRS 7534, Université Paris Dauphine
rossi@ufrmd.dauphine.fr
[3] Up to date contact informations for Fabrice Rossi are avaible at
http://apiacoa.org/

1 Introduction

Functional Data Analysis (see Ramsay and Silverman (1997)) is a framework which aims at improving traditional data analysis techniques when individuals are described by functions. This approach, contrary to its multivariate counterpart, takes advantage of some internal data structures to produce in general more pertinent results. The most obvious structure, in the functional context, is of course the regularity of the observed individuals: for example, in some spectrometric applications from food industry, each individual is described by a spectrum (say 100 chanels). The shape of these spectra are very smooth, and is handled in a natural way by the functional techniques. Another example of structure which can not be treated without a functional prior-knowledge, is the periodicity of the studied functions : for example, in Ramsay and Silverman (1997), the authors study the evolution of the Canadian temperature over one year. These data present strong sinusoidal patterns, which are taken into account by the functional framework.

In this paper, we recall the properties of the Functional Multi-Layer Perceptron (FMLP) based on a projection step. This model is in fact a natural extension of standard MLPs to the functional context, and its properties were studied from a theoretical point of view in some earlier works (see for example Conan-Guez and Rossi (2002) and Rossi and Conan-Guez (2004)). In this paper, we focus at comparing our model to standard MLPs on a real application:

[0] Published in IFCS'2004 Proceedings.
Available at http://apiacoa.org/publications/2004/fmlp-ifcs04.pdf

our problem is a discrimination task on phoneme data which is very similar to the one studied in Hastie et al. (1995).

2 Functional Multi-Layer Perceptrons

2.1 Functional neurons

A n input MLP neuron is characterized by a fixed activation function, T, a function from $\mathbb{R}$ to $\mathbb{R}$, by a vector from $\mathbb{R}^n$ (the weight vector, w) and by a real valued threshold, b. Given a vectorial input $x \in \mathbb{R}^n$, the output of the neuron is $N(x) = T(w.x + b)$.

In the proposed approach, we restrict ourselves to the case where inputs belong to the Hilbertian space $L^2(\mu)$ (with μ a finite positive Borel measure). In this framework, the extension of numerical neurons to functional inputs is straightforward. Indeed ,we consider two functions f and g, elements of $L^2(\mu)$. f is the input function, and g is called "the weight function" (g has the same meaning as w for the numerical neuron). The output of the functional neuron is $N(f) = T(\int fgd\mu + b)$ (where T is a function from $\mathbb{R}$ to $\mathbb{R}$ and b is a real value).

2.2 Functional MLP

The MLP architecture can be decomposed in neuron layers: the output of each layer (i.e., the vector formed by the output of neurons belonging to this layer) is the input of the next layer. As a functional neuron gives a numerical output, we can define a functional MLP by combining numerical neurons with functional neurons. The first hidden layer of the network consists exclusively in functional neurons (defined thanks to weight functions g_i), whereas subsequent layers are constructed exclusively with numerical neurons. For instance, an one hidden layer functional MLP with real output computes the following function:

$$H(f) = \sum_{i=1}^{k} a_i T\left(\int g_i f \, d\mu + b_i\right) \tag{1}$$

where a_i, b_i are real values, and f, g_i are elements of $L^2(\mu)$.

3 Projection based approach

3.1 Parametric approach

As stated above, the FMLP evaluation relies on the computation of all the integrals $\int g_i f \, d\mu$ of the first hidden layer. Unfortunately, in practice these integrals can not be calculated exactly, as g_i and f are arbitrary functions of

$L^2(\mu)$. One way to deal with this problem is to use a regularized representation of f and g_i in place of the true functions.

Let $(\phi_p)_{p\in\mathbb{N}^*}$ be a topological basis of $L^2(\mu)$, and let Π_P be the projection operator on the subspace spanned by the P first elements of the basis (denoted $span(\phi_1, ..., \phi_P)$), i.e. $\Pi_P(f) = \sum_{p=1}^{P}(\int f\phi_p \, d\mu)\phi_p$. Thanks to this projection step, a first simplification occurs in the FMLP evaluation: it is no more necessary to deal with the real input function f as well as to compute $H(f)$. We just consider the projection $\Pi_P(f)$ as the FMLP input, and our only concern is the evaluation of $H(\Pi_P(f))$. A second simplification can be applied to our model, we restrict the choice of weight functions to $span(\phi_1, ..., \phi_P)$. Therefore, for the weight function $g = \sum_{q=1}^{P} \alpha_q\phi_q$, the integral can be rewritten in the following form:

$$\int g\Pi_P(f) \, d\mu = \sum_{q=1}^{P}\sum_{p=1}^{P}(\int \phi_p f \, d\mu)\alpha_q \int \phi_p\phi_q \, d\mu = \alpha^T \Lambda\beta \tag{2}$$

where $\Lambda = (\int \phi_p\phi_q d\mu)_{p,q}$, and $\beta = (\int \phi_p f \, d\mu)_p$.

In this expression, each $\int \phi_p f \, d\mu$ is computed during the projection step (more precisely, an approximate value as explained in 3.2). The $\int \phi_p\phi_q \, d\mu$ are independant of the α_q as well as the input functions, therefore their evaluation can be done once and for all. Depending on the basis used to represent weight functions and input functions, this evaluation can be performed either exactly, or approximately.

Using linear models to represent weight functions allows the FMLP to be parameterized by a finite number of numerical parameters. Hence, the FMLP training can be performed with traditional optimization algorithms.

3.2 Approximation of the projection

As explained in 3.1, the proposed method aims at computing $H(\Pi_P(f))$. Unfortunaltely, due to our limited knowledge of f (f is known thanks to a finite number of input/output pairs), the computation of $\Pi_P(f)$ is not possible in practice. To overcome this problem, we substitute in the FMLP evaluation the real projection $\Pi_P(f)$ by an empirical one defined as follows.

Each studied function f is described by a list of observations $(x_j, f(x_j) + \varepsilon_j)_{0\leq j\leq m}$, where ε_j is the evaluation error on f at the observation point x_j (see Conan-Guez and Rossi (2002) for a detailed probabilistic description). It should be noted that the number of evaluation points m is free to vary from one function to another.

We define $\Pi_P(f)_m$, element of $L^2(\mu)$, as the unique minimizer $\sum_{p=1}^{P} \beta_p\phi_p$ of $\sum_{j=1}^{m}(f(x_j) + \varepsilon_j - \sum_{p=1}^{P} \beta_p\phi_p(x_j))^2$ defined thanks to the Moore-Penrose inverse. Thanks to this empirical projection, a second simplification occurs in the FMLP evaluation: after the substitution of the input function f by its projection $\Pi(f)$ in the section 3.1, we now focus on computing $H(\Pi_P(f)_m)$, rather than $H(\Pi_P(f))$.

3.3 Link with classical MLPs

In section 3.1, we have shown that the evaluation of the integral of a functional neuron could be reduce to some algebraic computation: $\int g \Pi_P(f)_m \, d\mu = \alpha^T \Lambda \beta_m$ where β_m is the coordinates of $\Pi_P(f)_m$. In fact, as we shall see now, the calculation of a functional neuron is equivalent to the one of its numerical counterpart. Indeed, as $(\phi_p)_{0 \leq p \leq P}$ is a free system, Λ is a full rank matrix. Therefore if we choose an arbitrary vector of coefficients c, we can define a function t by:

$$t = \sum_{q=1}^{P} d_q \phi_q$$

with $d = \Lambda^{-1} c$ such that

$$\int t \, \Pi_m(f) d\mu = \sum_{q=1}^{P} c_q \beta_{m,q}$$

Therefore, a linear combination of the (approximate) coordinates of f on $span(\phi_1, ..., \phi_P)$ is always equal to the scalar product of $\Pi_m(f)$ with a well chosen weight function t. From a practical point of view, we can submit the coordinates of $\Pi_m(f)$ to a standard MLP, and the prediction made by this model will be totally equivalent to the one of a FMLP. Consequently we see that our functional approach doesn't require specific software developement: an existing neural network library can be easily used to implement FMPs.

3.4 Theoretical properties

We studied from a theoretical point of view the proposed model in some earlier works (see Conan-Guez and Rossi (2002) and Rossi and Conan-Guez (2004)). We showed that FMLPs possess two important properties:

- the FMLP based on a projection step is a universal approximator, in the sense that for any real valued continuous function F defined on a compact K of $L^2(\mu)$, it exists P, the size of the truncated basis (the basis is fixed), and H a FMLP, such that F is approximated by $Ho\Pi_P$ to a given precision (the set of functions $Ho\Pi_P$ is dense in $C(K, \mathbb{R})$ for the uniform norm);
- the FMLP based on a projection step is consistent, in the sense that if we estimate the model parameters on a finite number of input functions, each one known thanks to a finite list of observations, these estimators converge to the theoretical parameters, when the number of functions as well as the number of observation points tend to infinity (more precisely, the number of observations needed to achieve a given precision depends on the number of functions).

4 Experiment

4.1 The data

The problem we are adressing in this paper is a discrimination task of phoneme data. This dataset can be found in the TIMIT database and was studied by Hastie et al. (1995) as well as Ferraty and Vieu (2003). The data are log-periodograms corresponding to recording phonemes of 32 ms duration. The goal of this experiment is to discriminate 5 different patterns corresponding to 5 different phonemes ("sh" as in "she", "dcl" as in "dark", "iy" as in "she", "aa" as in "dark", and "ao" as in "water"). These phonemes are part of the first sentence of the speach corpus. Each speaker (325 in the training set and 112 in the test set) is recorded at a 16-kHz sampling rate; and we retain only the first 256 frequencies. Finally, the training set contains 3340 spectra, whereas the test set contains 1169 spectra.

classes	aa	ao	dcl	iy	sh
training	519	759	562	852	648
test	176	263	195	311	224

Table 1. number of phonemes in the training/test set

The table 1 describes the distribution of each phoneme in the training set as well as in the test set. In figure 1, we draw 5 spectra of the phoneme "sh". We can see from this figure that spectra are very noisy.

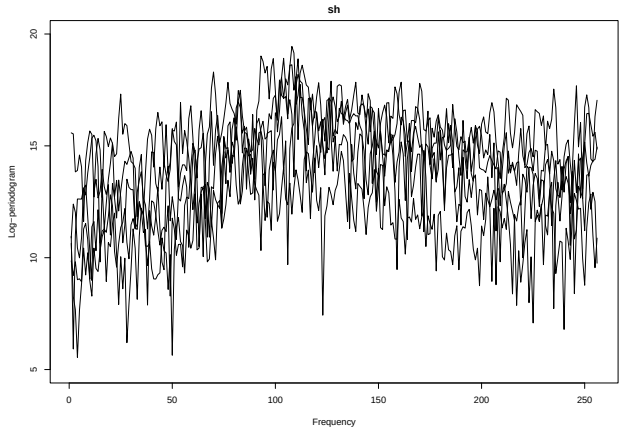

Fig. 1. 5 Log-Periodograms for the "sh" phoneme

4.2 Multivariate and functional approach

In this experiment, we aim at comparing the projection based FMP to standard MLPs. As it can be seen in the following part, both models are in many ways very similar. The main difference lies in the data pre-processing: the FMLP uses a functional pre-treatement, whereas the standard MLP relies on a raw pre-processing. It should be noted that apart from this practical difference, the functional model brings a theoretical framework which doesn't exist for the multivariate model (parameters consistency in 3.4).

Each spectrum is a vector of 256 components. In the case of the standard MLP, we compute the principal component analysis on these spectra and we retain a fixed number of eigenvectors according to the explained variance criterion. More precisely, each eigenvector wich explains more than 0.5% of the total variance is retained. The criterion imposes 11 eigenvectors. The projection components are then centered and scaled to unit variance. We finally submit these vectors to a standard MLP.

In the case of the FMLP, functions are first smoothed by a standard roughness penalty technique: the estimation of each curve is done by a spline with a penalty on the second derivative of the estimate (function `smooth.spline` in R). After this fitting step, each estimate is then sampled on 256 equally spaced points. The vector is then submitted to a principal component analysis, and the number of retained eigenfunctions is done according to the same explained variance criterion as above: we retain 10 eigenfunctions. This technique of smoothing the data before the PCA is well described in Ramsay and Silverman (1997). This functional approach allows to "let the data speak" while adding a functional constraint which get rid of the noise of the observed functions. In figure 2, we can see spline estimates of 5 spectra of the phoneme "sh".

After a centering and scaling stage, we submit the sample of each spline to a standard MLP (which is equivalent to the functional approach as explained in 3.3). In figure 3 and 4, we show the PCA for both approaches.

The MLP training is done thanks to a weight decay technique: we add to the error function a penalty term which constraints model parameters to be small (the penalty is the L^2-norm of parameter vector). In order to find the best model, we therefore have to choose the best architecture (number of hidden neurons in the MLP) as well as the smoothing parameter (which controls the weight decay). This can be done by a k-fold cross-validation technique. In our case, we choose k equal to 4, mainly for computational reasons. For the multivariate approach, 4 neurons is chosen, whereas for the functional one, 5 neurons is chosen (the range of hidden neurons for the k-fold cross-validation is from 3 to 7). The results obtained are summarized in table 2.

As it can be seen, the functional approach perfoms better than the multivariate one. This can be explained by the fact that spectra are very noisy, which penalizes the multivariate approach. The functional approach, thanks to its smoothing step, is able to eliminate some part of this noise, which leads

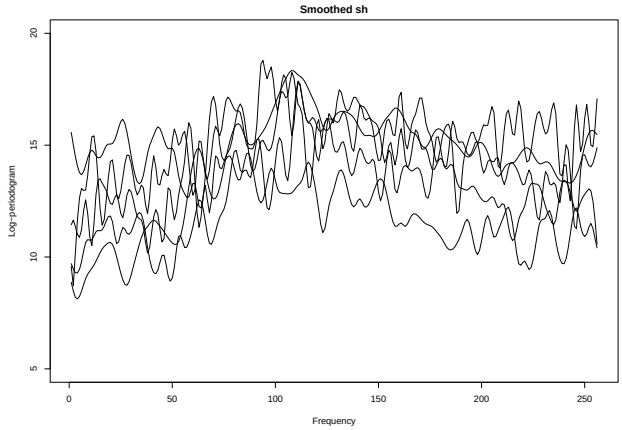

Fig. 2. 5 Log-Periodograms for the smooth "sh" phoneme

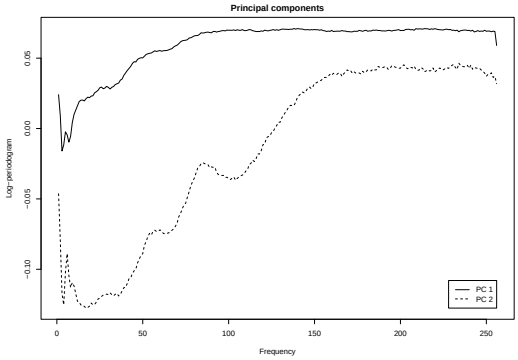

Fig. 3. the 2 first eigenvectors for the multivariate approach

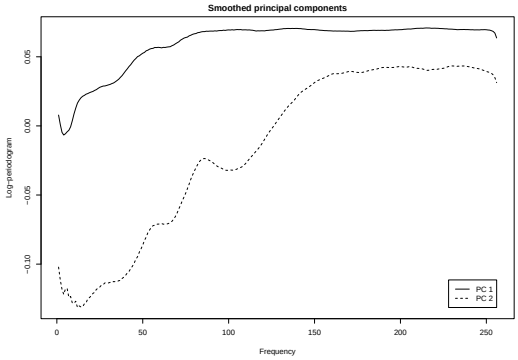

Fig. 4. the 2 first eigenfunctions for the functional approach

set	training	test
FMLP	7 %	7.7%
MLP	7.1%	8.3%

Table 2. Classification error rates of FMLPs and MLPs

to improved results. Moreover, the time needed by the smoothing phase of the functional approach is very small compared to the (F)MLP training time.

5 Conclusion

In this paper, we showed that a functional approach can be valuable compare to the multivariate approach. Indeed, in the proposed example, the FMLP is able to handle the noisy data in a more robust way than the standard MLP. Although this result is satisfactory, it should be noticed that the proposed approach doesn't use as much functional prior-knowledge as it could. According to this remark, one future way of investigation would be for example to add some functional penalties on the high frequencies of the eigenfunctions: indeed human ears are less sensitive to frequencies above 1 kHz. Moreover, as the k-fold cross-validation associated with neural networks is a very expensive technique, we were unable to use it in order to choose the number of retained eigenvectors (we used only a variance criterion). It would be interesting to assess this problem.

References

Conan-Guez, B., Rossi, F., August 2002. Multilayer perceptrons for functional data analysis: a projection based approach. In: Dorronsoro, J. R. (Ed.), Artificial Neural Networks – ICANN 2002. Springer, Madrid, pp. 667–672.

Ferraty, F., Vieu, P., 2003. Curves discriminations: a nonparametric functional approach. Computational Statistics and Data Analysis 44 (1–2), 161–173.

Hastie, T., Buja, A., Tibshirani, R., 1995. Penalized discriminant analysis. Annals of Statistics 23, 73–102.

Ramsay, J., Silverman, B., June 1997. Functional Data Analysis. Springer Series in Statistics. Springer Verlag.

Rossi, F., Conan-Guez, B., 2004. Functional multi-layer perceptron: a nonlinear tool for functional data analysis. Neural Networks (to be published).