Engineering Applications of Artificial Intelligence 93 (2020) 103682

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Auto-adaptive multi-scale Laplacian Pyramids for modeling non-uniform data

Ángela Fernández [a,*], Neta Rabin [c], Dalia Fishelov [d], José R. Dorronsoro [a,b]

[a] Departamento de Ingeniería Informática, Universidad Autónoma de Madrid, Spain
[b] Instituto de Ingeniería del Conocimiento, Madrid, Spain
[c] Department of Industrial Engineering, Tel-Aviv University, Israel
[d] Department of Mathematics, Afeka Academic College of Engineering, Israel

ARTICLE INFO

Keywords:
Laplacian Pyramids
Kernel methods
Overfitting
Multi-scale interpolation
Non-uniform data
Adaptive stopping

ABSTRACT

Kernel-based techniques have become a common way for describing the local and global relationships of data samples that are generated in real-world processes. In this research, we focus on a multi-scale kernel based technique named Auto-adaptive Laplacian Pyramids (ALP). This method can be useful for function approximation and interpolation. ALP is an extension of the standard Laplacian Pyramids model that incorporates a modified Leave-One-Out Cross Validation procedure, which makes the method stable and automatic in terms of parameters selection without extra cost. This paper introduces a new algorithm that extends ALP to fit datasets that are non-uniformly distributed. In particular, the optimal stopping criterion will be point-dependent with respect to the local noise level and the sample rate. Experimental results over real datasets highlight the advantages of the proposed multi-scale technique for modeling and learning complex, high dimensional data.

1. Introduction

An important challenge nowadays, when large amounts of data is collected, is the correct approximation of functions for modeling and analyzing data. These approximations have special interest in cases where the values of the functions are not known over the entire dataset. They can also be useful when the function may be too expensive to compute, or it is only represented on a finite expansion. There exist several methods for modeling and analyzing data, but when dealing with functions that depend on multiple variables, or that are defined over many scattered data points, the best way to tackle the general problem of approximation and interpolation is using an approach based on Radial Basis Functions (RBFs) (Buhmann, 2003).

An RBF is defined in terms of an univariate continuous function ϕ. Given a training sample $\{x_n\}_{n=1}^N$, a linear combination of RBFs approximates a real function f over a new data point x in the following way:

$$f(x) \approx s(x) = \sum_{\xi} w_\xi \phi(\|x - x_\xi\|).$$

Here, w_ξ represents the weight associated with the expansion points ξ, and $\|\cdot\|$ is an adequate norm. The most usual choice for the norm is the Euclidean distance, and one of the most common RBFs are Gaussians, where ϕ is defined as $\phi(r) = \exp(\epsilon r)^2$.

There exists a comprehensive literature on RBF methods and expansions (see for example Buhmann, 2003; Wang and Liu, 2002; Beatson and Light, 1997; Carozza and Rampone, 2001). In this work we will focus on Laplacian Pyramids (LP), a multi-scale model that generates a smoothed version of a function in an iterative manner, using Gaussian kernels of decreasing widths (Burt and Adelson, 1983). It is a simple method for learning functions from a general set of samples. The LP approximation algorithm works in the spirit of wavelets, as the reconstruction goes from coarser to finer scales and due to this fact, it is stable and convenient for working in the manifold learning context. It is also remarkable that this method can be seen as an iterative version of a Nadaraya–Watson estimator (Nadaraya, 1964; Watson, 1964). This classic estimator is typically defined as

$$g(x) = \frac{1}{n} \sum_{n} K(x, x_n) y_n,$$

where K is a kernel function and y_n are the objective function values on the training points x_n. As we shall see, the proposed LP procedure uses this type of construction at each iteration.

The LP scheme has been used for several applications in diverse domains; most of them utilize LP for function approximation and its out-of-sample extension. When a model does not allow a direct out-of-sample solution, the extension of the model to new points is not

* Corresponding author.
E-mail addresses: a.fernandez@uam.es (Á. Fernández), netara@tauex.tau.ac.il (N. Rabin), daliaf@afeka.ac.il (D. Fishelov), jose.dorronsoro@uam.es (J.R. Dorronsoro).

https://doi.org/10.1016/j.engappai.2020.103682
Received 15 July 2019; Received in revised form 13 March 2020; Accepted 28 April 2020
Available online xxxx
0952-1976/© 2020 Elsevier Ltd. All rights reserved.

trivial (Duchateau et al., 2013; Bengio et al., 2004; Long and Ferguson, 2019), and the LP method offers one way for extension of such models. In Mishne and Cohen (2012), an multi-scale anomaly detection algorithm that is based on diffusion maps (Coifman and Lafon, 2006a) was proposed. The diffusion maps embedding was calculated on a subset of the points and extended to the rest of the dataset with the LP function extension scheme. LP based out-of-sample extension for target detection was presented in Mishne et al. (2014). Extensions of the anomaly detection algorithm (Mishne et al., 2014), which utilizes LP for extension, include anomaly detection in side-scan sonar images of sea-mines (Mishne and Cohen, 2014b) and detection of defects in wafers (Mishne and Cohen, 2014a). LP was utilized for function extension in problems related to voice activity detection. In N. Spingarn and Cohen (2014), the likelihood ratio function of unlabeled data was learned by extending the likelihood ratios obtained from the labeled data. An LP-based speech enhancement algorithm was proposed in M. Li and Mousazadeh (2014). The LP-based extension was compared with the Geometric harmonics extension scheme (Coifman and Lafon, 2006b), for which parameters need to be carefully tuned, and it was shown that the LP-scheme provides better results.

Another domain in which Laplacian Pyramids have been applied is for data lifting. This challenge arises in models that first reduce the dimension of the data to obtain a compact and reliable representation, and then need to estimate new points in the ambient space from the low-dimensional embedding. LP-based lifting was applied in Dsilva et al. (2013) and Chiavazzo et al. (2014) for reconstruction of data in molecular dynamics applications and for modeling chemical kinetics. Last, the LP method has been applied in the context of kernel based forecasting in dynamical systems such as prediction of the North Pacific climate variability (Comeau et al., 2017), prediction of regional and pan-Arctic sea ice anomalies (Comeau et al., 2019) and forecasting of tropical intraseasonal oscillations (Alexander et al., 2017). In all of the above applications, heuristic approaches were used in order to limit the number of iterations of the LP algorithm to avoid a risk of overfitting if too many iterations are executed.

As mentioned above, and as it is often the case in machine learning, when an LP model is applied, one may overfit the data by refining the prediction too much during the training phase (Chiavazzo et al., 2014). In fact, it is difficult to decide when to stop the training phase to maximize the generalization capabilities of the resultant model. A usual approach is to apply the Cross Validation (CV) method (Duda et al., 2001, chap. 9) to measure a validation error during the training in order to stop when this error starts to increase. An extreme form of CV is Leave-One-Out CV (LOOCV): a model is built using all the samples but one, which is then used as a single validation pattern; this is repeated for each sample in the dataset, and the validation error is the average of all the one-pattern errors. Although LOOCV has a theoretical support and often yields good results, it has the drawback of being a time-consuming process.

Auto-adaptive LP (ALP), proposed in Fernández et al. (2016), is a modification of the LP training algorithm that merges training and an approximate LOOCV in one single phase. The ALP algorithm results in a LOOCV approximation that does not add any cost during the training step. This reduces significantly the training complexity and provides an automatic global criterion to stop training. Thus, the risk of overfitting, which may appear in a standard LP, is avoided. Therefore, ALP prevents overfitting the data and, moreover, it requires essentially no parametrization or expert knowledge about the problem under study, while still achieving a good test error. Moreover, it adds no extra cost compared to other classical neighbor-based interpolation methods.

In this paper we propose a new implementation of the ALP algorithm, providing a natural improvement of it. The main idea is to work with a local (point-wise) kernel scale that better suited to the density of the data and, consequently, to perform the optimal number of training iterations around each point. This modification allows us to deal with datasets where sample densities vary in different regions, which may require a different resolution. The algorithm will be automatically adapted to each of these areas when necessary. The proposed method can also be seen as a variant of the iterative Nadaraya–Watson regression with L_2 boosting (Bühlmann and Yu, 2003).

To sum up, the contribution of this paper is twofold. On one hand, it presents a self-contained explanation about Laplacian Pyramids, including a complete analysis of the error bounds and decay rates. Moreover we review in detail the automatic stopping criteria which is integrated into the algorithm without extra computational cost, prevents overfitting and bypasses the need for heuristic approaches to set the parameters of the method. On the other hand, the second contribution is an extension of the ALP algorithm to a local resolution setting, taking advantage of the different sample statistics that might appear in the data.

This paper is organized as follows. In Section 2 we briefly review the LP model and present a detailed analysis of its training error. We describe classical ALP and its LOOCV estimation in Section 3, and an improved ALP version with local resolution is presented. The algorithm description is accompanied by a synthetic example to illustrate its behavior. Results over several datasets are shown in Section 4, and the paper ends with some conclusions in Section 5.

2. Laplacian pyramids

Laplacian Pyramids (LP) is an iterative model introduced by Burt and Adelson (1983) for image processing applications. In its traditional form, the LP algorithm decomposes the input image into a series of sub-images, and each of them captures a different frequency band of the original one. This process is carried out by constructing Gaussian kernel-based smoothing masks of different widths, followed by a down-sampling (quantization) step. LP was later proved to be a tight frame (see Do and Vetterli, 2003) and used for signal processing applications, for example as a reconstruction scheme in Liu et al. (2008). In Rabin and Coifman (2012), a multi-scale algorithm was introduced in the spirit of LP to be applied in the setting of high-dimensional data analysis. In particular, it was proposed as a simple method for extending low-dimensional embedding coordinates, that result from the application of a non-linear dimensionality reduction technique, to a high-dimensional dataset (this has been recently applied in Mishne and Cohen, 2013).

2.1. The basic LP procedure

Next, we review the LP procedure as described in Rabin and Coifman (2012) (note that the down-sampling step, which is part of Burt and Adelson's algorithm is skipped here). Let $S = \{(x_i, f_i = f(x_i))\}_{i=1}^N, x_i \in \mathbb{R}^M$ be the sample dataset where f is a function which is only known on the sample points. For simplicity we assume that it belongs to a Sobolev space (Adams and Fournier, 2003) $\|f\|_{m,2}$ for a certain m. The algorithm approximates the function f by constructing a series of functions $\{\tilde{f}^{(\ell)}\}$ obtained by several refinements $d^{(\ell)}$ over the approximation errors. In a slight abuse of notation we will use the same name f for both the general function $f(x)$ and also for the vector of its sample values $f = (f_1 = f(x_1), \ldots, f_N = f(x_N))$. The end result of this process yields a function approximation to f in the form

$$f \simeq \tilde{f} = \tilde{f}^{(0)} + d^{(1)} + d^{(2)} + d^{(3)} + \cdots$$

In more detail, a first level kernel $K^{(0)}(x, x') = \Phi\left(\mathrm{dist}(x, x')/\sigma\right)$ is chosen using a one dimensional positive function $\Phi(z)$ and a wide, initial scale σ; $\mathrm{dist}(x, x')$ denotes some distance function between points in the ambient space. As mentioned before, the Gaussian kernel with Euclidean distances is applied here, i.e., we take $\mathrm{dist}(x, x') = \|x - x'\|$. Then, we define

$$K^{(0)}(x, x') = \kappa^{(0)} e^{-\frac{\|x-x'\|^2}{\sigma^2}},$$

where $\kappa^{(0)}$ is the Gaussian kernel normalizing constant which depends on σ.

The notation $K^{(0)}$ is used (with a slight abuse of notation) for the general continuous kernel $K^{(0)}(x,x')$ and its discrete matrix counterpart $K^{(0)}_{jk} = K^{(0)}(x_j, x_k)$ over the sample points. The smoothing operator $P^{(0)}$ is constructed as the row-stochastic normalized kernel matrix

$$P^{(0)}_{ij} = \frac{K^{(0)}_{ij}}{\sum_k K^{(0)}_{ik}}. \tag{1}$$

A first coarse representation of f is then generated by the convolution $\tilde{f}^{(0)} = f * P^{(0)}$ that captures the low-frequencies of the function. For the next steps, a parameter value $\mu > 1$ is fixed. A sharper normalized Gaussian kernel matrix $P^{(\ell)}$ is constructed at level ℓ with scale σ/μ^{ℓ}. Then, the residual $d^{(\ell-1)} = f - \tilde{f}^{(\ell-1)}$ is computed. It captures the error of the approximation of f at the previous $\ell - 1$ step. A more detailed representation of f is generated. It is given by

$$\tilde{f}^{(\ell)} = \tilde{f}^{(\ell-1)} + d^{(\ell-1)} * P^{(\ell)} = \tilde{f}^{(\ell-1)} + g^{(\ell)},$$

with $g^{(\ell)} = d^{(\ell-1)} * P^{(\ell)}$. The iterative algorithm stops once the norm of the residual vector $d^{(\ell)}$ is smaller than a predefined tolerance. Stopping at iteration L, the final LP model has the form

$$\tilde{f}^{(L)} = \tilde{f}^{(0)} + \sum_{\ell=1}^{L} g^{(\ell)} = f * P^{(0)} + \sum_{\ell=1}^{L} d^{(\ell-1)} * P^{(\ell)}. \tag{2}$$

Extending this multi-scale representation to a new data point $x \in \mathbb{R}^M$ is now straightforward by setting

$$\begin{aligned}\tilde{f}^{(L)}(x) &= f * P^{(0)}(x) + \sum_{\ell=1}^{L} d^{(\ell-1)} * P^{(\ell)}(x) \\ &= \sum_j f_j P^{(0)}(x, x_j) + \sum_{\ell=1}^{L}\sum_j d_j^{(\ell-1)} P^{(\ell)}(x, x_j).\end{aligned}$$

The kernels $P^{(\ell)}$ are directly extended for a new point x as

$$P^{(\ell)}(x, x_j) = \frac{K^{(\ell)}(x, x_j)}{\sum_k K^{(\ell)}(x, x_k)} \tag{3}$$

with $K^{(\ell)}(x,x') = \kappa^{(\ell)} e^{-\frac{\|x-x'\|^2}{(\sigma/\mu^{\ell})^2}}$. Observe that when defining $P^{(\ell)}$, $\kappa^{(\ell)}$ disappears for being also present in the denominator.

Next, we show that the L_2 norms of the residuals $d^{(\ell)}$ decay extremely fast.

2.2. Error analysis for the LP scheme

For analyzing the LP error, the previously defined kernel is considered. First notice that, when working in the continuous kernel setting, the summation becomes an integral. Therefore, we have $P^{(\ell)}(x,x') = K^{(\ell)}(x,x')$ for a Gaussian function since the denominator in (3) is just $\int K^{(\ell)}(x,z)dz = 1$.

Furthermore, for all ℓ, writing now $P^{(\ell)}(x) = P^{(\ell)}(x,0)$, P is an approximation to a delta function satisfying

$$\begin{aligned}&\int P^{(\ell)}(x)\,dx = 1,\\ &\int xP^{(\ell)}(x)\,dx = 0,\\ &\int \|x\|_2^2 P^{(\ell)}(x)\,dx \le 2C, \text{ where } C \text{ is a constant.}\end{aligned} \tag{4}$$

Assume that f is in L_2, i.e., $\int_x f^2(x)\,dx < \infty$. The LP scheme is a relaxation process for which in the first step the function f is approximated by $\mathcal{G}^{(0)}(f) = f * P^{(0)}(x)$. In the second step f is approximated by $\mathcal{G}^{(0)}(f) + \mathcal{G}^{(1)}(d^{(0)})$, where $d^{(0)} = f - \mathcal{G}^{(0)}(f)$ and $\mathcal{G}^{(1)}(d^{(0)}) = d^{(0)} * P^{(1)}(x)$, and so on. Taking the Fourier transform of $P^{(\ell)}(x)$ results in (see Fishelov, 1990)

$$\left|\hat{P}^{(\ell)}(\omega) - 1\right| \le \frac{(\sigma/\mu^{\ell})^2}{2}\int \|x\|_2^2\|\omega\|_2^2 P^{(\ell)}(x)\,dx \le C\,(\sigma/\mu^{\ell})^2\,\|\omega\|_2^2. \tag{5}$$

We first analyze the error $d^{(0)}(x)$ in the first step, which is defined by $d^{(0)}(x) = f - f * P^{(0)}(x)$. Taking the Fourier transform of $d^{(0)}(x)$ together with the bound in (5) yields

$$\left|\hat{d}^{(0)}(\omega)\right| = \left|\hat{f}(w)\right|\left|\hat{P}^{(0)}(\omega) - 1\right| \le C\|\omega\|_2^2\sigma^2\left|\hat{f}(\omega)\right|. \tag{6}$$

The error in the second step is

$$d^{(1)}(x) = d^{(0)} - \mathcal{G}^{(1)}(d^{(0)}) = \left(f - f * P^{(0)}\right) - d^{(0)} * P^{(1)} = d^{(0)} - d^{(0)} * P^{(1)}. \tag{7}$$

Taking the Fourier transform of (7) yields

$$\begin{aligned}\left|\hat{d}^{(1)}(\omega)\right| &= \left|\hat{d}^{(0)}(\omega) - \hat{d}^{(0)}(\omega)\hat{P}^{(1)}(\omega)\right| \\ &= \left|\hat{d^{(0)}}(\omega)\right|\left|\hat{P}^{(1)}(\omega) - 1\right|.\end{aligned} \tag{8}$$

Using (5) and (6) we obtain

$$\left|\hat{d}^{(1)}(\omega)\right| \le C\|\omega\|_2^2\left|\hat{d}^{(0)}(\omega)\right|(\sigma/\mu)^2 \le C\sigma^2\,(\sigma/\mu)^2\left|\hat{f}(\omega)\right|\|\omega\|_2^4. \tag{9}$$

Since $\mu > 1$, then $\left|\hat{d}^{(1)}(\omega)\right| \le C\sigma^2\frac{\sigma^2}{\mu^2}\left|\hat{f}(\omega)\right|\|\omega\|_2^4$. Similarly, for the ℓth step the error is bounded by

$$\left|\hat{d}^{(\ell)}(\omega)\right| \le C\sigma^2\left(\frac{\sigma^2}{\mu^{(\ell+1)}}\right)^{\ell}\left|\hat{f}(\omega)\right|\|\omega\|_2^{2(\ell+1)}. \tag{10}$$

By Parseval's equality we obtain

$$\begin{aligned}\left\|d^{(\ell)}\right\|_{L^2} = \left\|\hat{d}^{(\ell)}\right\|_{L^2} &\le C\sigma^2\left(\frac{\sigma^2}{\mu^{(\ell+1)}}\right)^{\ell}\left\|\hat{f}(\omega)\|\omega\|_2^{2(\ell+1)}\right\|_{L^2} \\ &\le C\sigma^2\left(\frac{\sigma^2}{\mu^{(\ell+1)}}\right)^{\ell}\|f\|_{2\ell+2,2},\end{aligned} \tag{11}$$

where $\|f\|_{m,2}$ denotes the Sobolev norm of a function with up to m derivatives in L_2. Thus, the L_2 norm of the LP error decays at a very fast rate.

2.3. Overfitting risk

Since the error of the LP method decays fast, setting a small error threshold may easily result in $\tilde{f}^{(\ell)} \simeq f$ and hence cause overfitting of the data. In order to understand the approximation process, we express $\tilde{f}^{(\ell)} = \tilde{f}^{(\ell-1)} + g^{(\ell)}$ by

$$\begin{aligned}\tilde{f}^{(\ell)} &= \tilde{f}^{(\ell-1)} + g^{(\ell)} = \tilde{f}^{(\ell-1)} + (f - \tilde{f}^{(\ell-1)}) * P^{(\ell)} \\ &= f * P^{(\ell)} + \tilde{f}^{(\ell-1)} * (I - P^{(\ell)}),\end{aligned}$$

where I denotes the identity matrix. Now, taking the limit $\tilde{f}^{(\ell)} \to \phi$ yields

$$\phi = f * \lim P^{(\ell)} + \phi * \lim(I - P^{(\ell)}),$$

i.e., $\phi = f$, for $P^{(\ell)} \to I$. In practice, when ℓ is large enough, numerically $P^{(\ell)} \simeq I$, so that $K^{(\ell)}(x_i, x_j) \simeq 0$, $i \ne j$. Then $d_j^{(\ell)} = 0$ for all j and the LP stays almost the same for a large enough ℓ. In other words, care has to be taken when deciding when to stop the LP iterations in order to avoid overfitting.

3. Auto-adaptive Laplacian pyramids

The standard way to prevent overfitting is to use an independent validation subset and to stop the iterations as soon as the validation error on that subset starts to increase. This can be problematic for small samples as it introduces a random dependence on the choice of the particular validation subset. k-fold Cross Validation (CV) is then usually the standard choice to avoid this. Samples are randomly distributed in k subsets, and iteratively $k-1$ subsets are used for training while the remaining $N-(k-1)$ are used for validation. In the extreme case when $k = N$, i.e., when just one pattern is used for validation, CV becomes Leave-One-Out Cross Validation (LOOCV) and the train iterations are stopped when the LOOCV error starts to

increase. Besides its simplicity, LOOCV has the attractive property of being an almost unbiased estimator of the true generalization error (see for instance Cawley and Talbot, 2004; Elisseeff and Pontil, 2002), possibly with a high variance (Kohavi, 1995). In our case, LOOCV can be easily applied using for training a $N \times N$ normalized kernel matrix $P_{(p)}$. This is just the previous matrix P, where we set to 0 the pth rows and columns when x_p is held out of the training sample and used for validation. The most obvious drawback of LOOCV is its rather high cost, which in our case would be in principle $N \times O(LN^2) = O(LN^3)$, where we recall that L is the number of LP iterations. However, it is often the case for other models that there are ways to estimate the LOOCV error with a smaller cost. This can be done exactly in the case of k-Nearest Neighbors (Fukunaga and Hummels, 1989) or Ordinary Least Squares (Hastie et al., 2008, Chapter 7); or approximately for Support Vector Machines (Chapelle et al., 2002) or Gaussian Processes (Rasmussen and Williams, 2005). We show next how to perform LOOCV without essentially augment training cost.

3.1. Standard ALP

In the context of this work and to alleviate the LOOCV cost, notice first that here when x_p is removed from the training sample, the test value of the $f_{(p)}$ extension at the point x_p is given by

$$f^{(L)}_{(p)}(x_p) = \sum_{j\neq p} f_j P^{(0)}(x_p, x_j) + \sum_{\ell=1}^{L}\sum_{j\neq p} d^{(\ell-1)}_{(p);j} P^{(\ell)}(x_p, x_j)$$
$$= \sum_{j} f_j \tilde{P}^{(0)}(x_p, x_j) + \sum_{\ell=1}^{L}\sum_{j} d^{(\ell-1)}_{(p);j} \tilde{P}^{(\ell)}(x_p, x_j).$$

Here $\tilde{P}^{(\ell)}$ is a modification of $P^{(\ell)}$, where the diagonal elements in $\tilde{P}^{(\ell)}$ are set to zero, i.e., $\tilde{P}_{i,j} = P_{i,j}$ when $j \neq i$. $d^{(\ell)}_{(p)}$ are the different previously defined errors computed using the $P^{(\ell)}_{(p)}$ matrices.

This observation leads to the modification of the standard LP that was proposed in Rabin and Coifman (2012), and which simply consists on applying the LP procedure described in Section 2 but replacing the P matrix by its 0-diagonal version $\tilde{P}$. This modification requires the computation of the function approximation $\tilde{f}^{(0)} = f * \tilde{P}^{(0)}$ at the beginning, and then the vectors $\tilde{g}^{(\ell)} = \tilde{d}^{(\ell-1)} * \tilde{P}^{(\ell)}$, $\tilde{f}^{(\ell)} = \tilde{f}^{(\ell-1)} + \tilde{g}^{(\ell)}$ and $\tilde{d}^{(\ell)} = f - \tilde{f}^{(\ell)}$ are computed at each iteration. This algorithm is denoted as the *Auto-adaptive Laplacian Pyramid (ALP)* (Fernández et al., 2016).

According to the previous formula for the $f^{(L)}_{(p)}(x_p)$, we can take the ALP values $\tilde{f}^{(L)}_p = \tilde{f}^{(L)}(x_p)$ given by

$$\tilde{f}^{(L)}(x_p) = \sum_{j} f_j \tilde{P}^{(0)}(x_p, x_j) + \sum_{\ell=1}^{L}\sum_{j} \tilde{d}^{(\ell-1)}_{j} \tilde{P}^{(\ell)}(x_p, x_j),$$

as approximations to the LOOCV validation values $f^{(L)}_{(p)}(x_p)$.

The squared LOOCV error at each iteration may be approximated by

$$\sum_p (f(x_p) - f^{(L)}_{(p)}(x_p))^2 \simeq \sum_p (f(x_p) - \tilde{f}^{(L)}_p)^2 = \sum_p (\tilde{d}^{(L)}_p)^2,$$

which is just the training error of ALP at the current iteration. In other words, working with the $\tilde{P}$ matrix instead of P, the training error at step L gives in fact an approximation to the LOOCV error at this step.

For the parameter selection, choosing as customarily done $\mu = 2$, the only required parameter would be the initial σ but assuming it is wide enough, its $\sigma/2^{\ell}$ scalings will yield an adequate final kernel width. Furthermore, in this paper we propose a heuristic technique to compute the initial bandwidth σ and the maximum number of iterations $\max_{\text{its}}$ ensuring that all the interesting σ values (and only the interesting ones) will be tested. Relevant values of σ are obtained in between the ones that yield an all-ones P matrix, and the ones that produce a $P = I$ matrix. In the first case, every data point obtains the same weight, resulting in a mean function. In the second case, as mentioned in Section 2.3, the original function is reproduced. To approximate these values, the parameters are fixed in the following way:

$$\sigma = 10 \max(W_{ij})$$
$$\max_{\text{its}} = \log_2(\sigma/\sigma_{\min}), \quad \text{with } \sigma_{\min} = 1/5 \min(W_{ij}),$$

where W represents the distance matrix.

We also point out that it is straightforward to extend the model to a vectorial function $F = (F_1, \ldots, F_M)$. The squared ALP error at the Lth iteration becomes

$$\sum_p \|F(x_p) - F^{(L)}_{(p)}(x_p)\|^2 \simeq \sum_p \|F(x_p) - \tilde{F}^{(L)}_p\|^2 = \sum_p \|\tilde{D}^{(L)}_p\|^2,$$

where here $D = (D_1, \ldots, D_M)$ is the vector formed of the ALP residuals D_m of each component F_m of F. Again, the optimal value for L is set as the iteration where this estimate of the LOOCV error begins to grow.

The cost of running L steps of ALP is just $O(LN^2)$ and, thus, we gain the advantage of approximating the exhaustive LOOCV without any additional cost on the overall algorithm. The complete training and test procedures are presented in Algorithms 1 and 2 respectively. Notice that in this work, the computed error for setting the optimal stopping iterations number of the algorithm is the Root Mean Squared Error (RMSE). This type of error measure was selected because it is the classical error to compute when dealing with functions and due to the improved results obtained empirically compared with the Mean Absolute Error (MAE).

Algorithm 1 The ALP Training Algorithm

Input: $\{x_i, F_i\}_{i=1}^N$.
Output: $(\{\tilde{d}^{(\ell)}\}, \sigma, \mu, L)$, the trained model.
$\mu \leftarrow 2$, $W_{ij} \leftarrow x_i - x_j \quad \forall i, j$, $\sigma \leftarrow 10 \max(W_{ij})$, $\max_{\text{its}} \leftarrow \log_2(5 \frac{\sigma}{\min(W_{ij})})$.
$\tilde{F}^{(0)} \leftarrow F$; $\tilde{F}^{(0)} \leftarrow 0$; $\ell \leftarrow 1$.
while $(\ell < \max_{\text{its}})$ **do**
  $\tilde{K}^{(\ell)} \leftarrow e^{-\frac{\|W\|^2}{\sigma^2}}$, with 0-diagonal. % LOOCV approximation.
  $\tilde{P}^{(\ell)} \leftarrow \text{normalize}(\tilde{K}^{(\ell)})$.
  $\tilde{F}^{(\ell)} \leftarrow \tilde{F}^{(\ell-1)} + \tilde{D}^{(\ell-1)} * \tilde{P}^{(\ell)}$.
  $\tilde{D}^{(\ell)} \leftarrow F - \tilde{F}^{(\ell)}$.
  $\text{err}^{(\ell)} \leftarrow \|\tilde{D}^{(\ell)}\|^2$. % For each point a mean error value is computed.
  $\sigma \leftarrow \sigma/\mu$; $\ell \leftarrow \ell + 1$.
end while
$L \leftarrow \arg\min_{\ell}\{\text{err}^{\ell}\}$. % Optimal iteration.

Algorithm 2 The ALP Testing Algorithm

Input: $\{x_i\}_{i=1}^N$, x_{te}, $(\{\tilde{D}^{(\ell)}\}, \sigma, \mu, L)$.
Output: $\hat{F}_{\text{te}}$.
$\hat{F}_{\text{te}} \leftarrow 0$.
for ℓ=1 **to** L **do**
  $K^{(\ell)}_{i;\text{te}} \leftarrow e^{-\frac{\|x_i - x_{\text{te}}\|^2}{\sigma^2}} \quad \forall i$.
  $P^{(\ell)} \leftarrow \text{normalize}(K^{(\ell)})$.
  $P^{(\ell)} \leftarrow \text{normalize}(K^{(\ell)})$.
  $\hat{F}_{\text{te}} \leftarrow \hat{F}_{\text{te}} + \tilde{D}^{(\ell-1)} * P^{(\ell)}$.
  $\sigma \leftarrow \sigma/\mu$.
end for

As it has just being argued, the obvious advantage of ALP is that when the training error is evaluated, in practice the LOOCV error after each LP iteration is estimated. Therefore, the evolution of these LOOCV values automatically defines the optimal iteration at which the algorithm is stopped, i.e., just when the training error starts to increase. Thus, the risk of overfitting is removed and in addition the training errors can be used as an approximation to the generalization error. This effect can be seen in Fig. 1 that illustrates the application of ALP to the synthetic problem described in the next subsection. In this example, the optimum stopping time for ALP is exactly the same as the LOOCV error would generate, where the training error stabilizes afterwards at

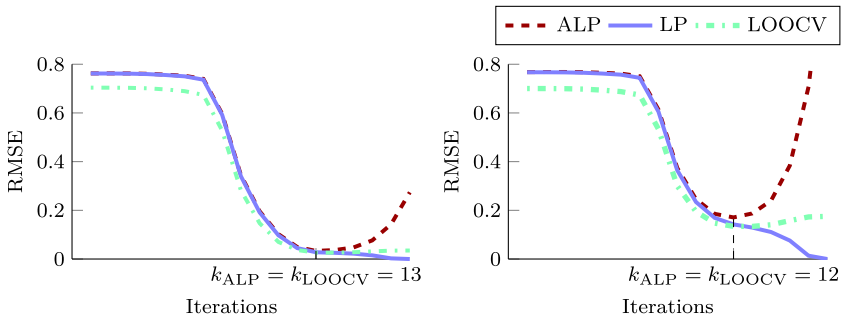


Fig. 1. Training errors for the original LP models (with and without LOOCV) and its modified version, the ALP model, applied over a perturbed sine. Left: $\delta = 0.1$ over 4000 patterns; right: $\delta = 0.5$ over 2000 patterns.

a slightly larger value. Moreover, ALP achieves an automatic selection of the width of the Gaussian kernel which makes this version of LP to be auto-adaptive as it does not require costly parameter selection procedures.

3.2. ALP with local resolution

When working with scattered datasets, it is often the case that the available data is not equally distributed, and there exist regions with different sample statistics and density characteristics. In these cases, it makes sense to define different models, with different stopping times, adapted to each region.

Focusing on the ALP method, the original algorithm may be modified to take into account this particular issue. We propose to use a point-wise σ value, computed by using the neighborhood of each sample point. To accomplish this goal, the train and test phases are modified as follows.

- **Training step:** An error per point is computed in terms of the ν nearest neighbors mean error. The optimal number of iterations will be given then by $L_i^* \leftarrow \arg\min_\ell \{\text{err}_i^{(\ell)}\}$.
- **Test step:** The optimal number of iterations L_{te}^* of each test point will be given by the optimal number of iterations of the nearest training point x_n, i.e. $L_{\text{te}}^* = L_n^*$.

The proposed modification introduces a new parameter ν, that represents the number of neighbors taken into account to define the best local σ per point. This parameter will have the same value for every point in the training set, and it will be determined by CV.

To sum up and remark the changes with respect to the original ALP algorithm, the corresponding training and test procedures are presented in Algorithms 3 and 4.

Algorithm 3 The Local ALP Training Algorithm

Input: $\{x_i, F_i\}_{i=1}^N$.
Output: $(\{\tilde{D}^{(\ell)}\}, \sigma, \mu, \{L_i\}_{i=1}^N)$.
$\mu \leftarrow 2$, $W_{ij} \leftarrow x_i - x_j \quad \forall i, j$, $\sigma \leftarrow 10 \max(W_{ij})$, $\max_{\text{its}} \leftarrow \log_2(5\frac{\sigma}{\min(W_{ij})})$.
$\tilde{D}^{(0)} \leftarrow f$; $\tilde{F}^{(0)} \leftarrow 0$; $\ell \leftarrow 1$.
while $(\ell < \max_{\text{its}})$ **do**
  $\tilde{K}^{(\ell)} \leftarrow e^{-\frac{\|W\|^2}{\sigma^2}}$, with 0-diagonal.
  $\tilde{P}^{(\ell)} \leftarrow \text{normalize}(\tilde{K}^{(\ell)})$.
  $\tilde{F}^{(\ell)} \leftarrow \tilde{F}^{(\ell-1)} + \tilde{D}^{(\ell-1)} * \tilde{P}^{(\ell)}$.
  $\tilde{D}^{(\ell)} \leftarrow F - \tilde{F}^{(\ell)}$.
  $\text{err}_i^{(\ell)} \leftarrow \|\tilde{D}_i^{(\ell)}\|^2 \quad \forall i$. % For each point the mean error of the nearest points is computed.
  $\sigma \leftarrow \sigma/\mu$; $\ell \leftarrow \ell + 1$.
end while
$L_i \leftarrow \arg\min_\ell \{\text{err}_i^{(\ell)}\}$. % Optimal iteration ($\sigma$ value) per point.

Algorithm 4 The Local ALP Testing Algorithm

Input: $\{x_i\}_{i=1}^N$, x_{te}, $(\{\tilde{D}^{(\ell)}\}, \sigma, \mu, \{L_i\}_{i=1}^N)$.
Output: $\hat{F}_{\text{te}}$, L_{te}.
$\hat{F}_{\text{te}} \leftarrow 0$.
for $\ell = 1$ **to** $\max\{L_i\}$ **do**
  $K_{i;\text{te}}^{(\ell)} \leftarrow e^{-\frac{\|x_i - x_{\text{te}}\|^2}{\sigma^2}} \quad \forall i$.
  $P^{(\ell)} \leftarrow \text{normalize}(K^{(\ell)})$.
  $\hat{F}_{\text{te}} \leftarrow \hat{F}_{\text{te}} + \tilde{D}^{(\ell-1)} * P^{(\ell)}$.
  $\sigma \leftarrow \sigma/\mu$.
  if ℓ==1 **then**
    $L_{\text{te}} \leftarrow L\left(\arg\min_i \{x_i - x_{\text{te}}\}\right)$. % Return also the optimal iteration per point.
  end if
end for

When applying this new ALP version, the risk of overfitting is still removed. In this case, in which the width of the kernel is automatically adapted to each point, the overfitting risk is prevented in a point-wise manner. We will see an example of this behavior in the following subsection.

3.3. A synthetic example

For a better understanding of the proposed method and its advantages, we illustrate in this section the classic ALP algorithm and its local resolution version on a synthetic example of a composition of sines with different frequencies plus additive noise.

Consider a sample x with N points equally spaced over the range $[0, 10\pi]$. The target function f is given by

$$f = \sin(x) + 0.5\sin(3x) \cdot I_2(x) + 0.25\sin(9x) \cdot I_3(x) + \varepsilon,$$

where I_2 is the indicator function of the interval $(10\pi/3, 10\pi]$, I_3 that of $(2 \cdot 10\pi/3, 10\pi]$ and $\varepsilon \sim \mathcal{U}([-\delta, \delta])$ is uniformly distributed noise. In other words, there is a single frequency in the interval $[0, 10\pi/3]$, two frequencies in $(10\pi/3, 2 \cdot 10\pi/3]$ and three in $(2 \cdot 10\pi/3, 10\pi]$. For the classic ALP algorithm, two different simulations are executed: the first one with 4000 points with small $\delta = 0.1$ noise and the second one with 2000 points and a larger $\delta = 0.5$ (observe that $|f| \leq 1.75$). In both cases, $1/3$ of the original set is randomly chosen for test purposes.

Recall that the main advantage of ALP is the approximation of the LOOCV error obtained while we evaluate the training error. Due to this fact, if the algorithm iterations stop as the error starts to grow, the risk of overfitting is removed. This effect can be observed in Fig. 1 where ALP is applied to this synthetic example. The solid blue and dashed green lines represent the LP training error and the true LOOCV error per iteration respectively, and the dashed red line represents the error for the ALP method. Notice that the ALP training error attains its minimum at the same iteration prescribed by exact LOOCV for LP.

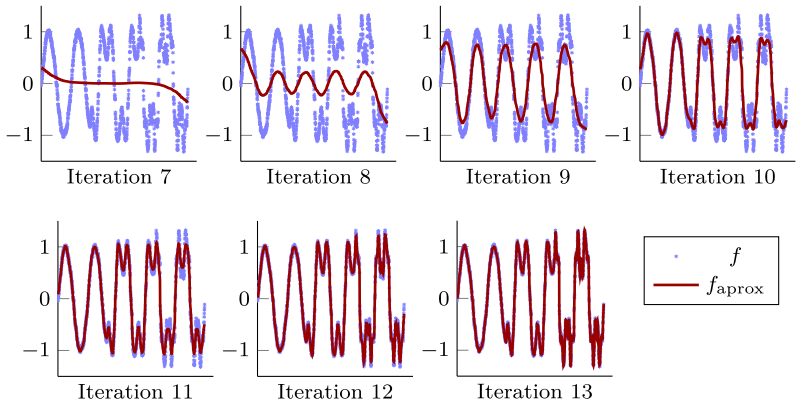


Fig. 2. Interpolations given by the ALP model for the last seven steps (out of 13) on the small noise example.

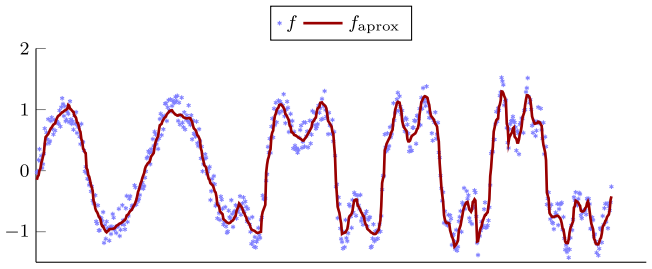


Fig. 3. Final interpolations given by the ALP model for the large noise example.

Recall also that the ALP model automatically adapts its multiscale behavior to the data, trying to refine the prediction in each iteration using a more localized kernel, given by a smaller σ. This behavior can be observed in Fig. 2, which shows the evolution of the ALP predictions for the 0.1-noise experiment. At the beginning, the model approximates the function just by a coarse mean of the target function values, and in the subsequent iterations when the model starts using sharper kernels and refined residuals, the approximating function captures the different frequencies and amplitudes of the composite sines. In this particular case, the minimum LOOCV value is reached after 13 iterations. Note that we start plotting from the 7th iteration, because before the result resembles a mean function due to a large initial σ value.

Next, the same synthetic experiment is carried out, but now the amplitude of the uniform noise distribution is increased to $\delta = 0.5$. The predicted function is represented in Fig. 3 and it is obtained after 12 iterations (as shown in the right hand image of Fig. 1). As expected, the number of LP iterations is now slightly smaller than in the previous example because the algorithm selects a more conservative, smoother prediction in the presence of noisier and, thus, more difficult data.

In any case, we can conclude that the ALP model captures very well the essential underlying behavior of both samples, as it identifies the three different frequencies of the sine and their amplitudes even when the noise level increases.

Next, we illustrate in this subsection the advantages of the modified ALP version with local resolution (ALP$_l$). For this purpose, the same sine example is used, but the focus is on the third interval, where the three frequencies of the sine are present. The function takes the form $f = \sin(x) + 0.5\sin(3x) + 0.25\sin(9x)$, with 4000 points and no noise ($\delta = 0$, and thus $\varepsilon = 0$). In this case, also three different regions are defined, but in terms of the subsampling density. In particular, we divide the interval $[0, 10\pi]$ into three parts. The first region holds 400 samples, the second 1400 and the third one 2200. In this scenario it seems logical to adapt the final bandwidth of the Gaussians to each region, expecting larger bandwidth values (few iterations) for sparse regions, obtaining then a smoother, coarser approximation; for the dense regions we expect a smaller σ value (more iterations), that will capture finer details, as there are more points to estimate the original function).

The parameters of the method have been fixed automatically, as done in the ALP case, except the parameter ν for the local approximation that has been cross-validated using 10 folds and a grid $\{10, 20, \ldots, 200\}$, selecting finally a value of 50. As seen in Fig. 4, the resolution selection works as expected, presenting three main different values corresponding to the three different subsampling regions.

The original ALP and ALP$_l$ algorithms are compared in Fig. 5 for the three different regions, where we can acknowledge that in general the blue ALP$_l$ points are very close to the red ALP values. Both methods present very similar results, as expected because they used very similar σ values, and they give a good result for prediction. In particular, this can be observed in the second and third region, where the sample set is dense and big enough. The first region, which only has a few number of available sampled point, is the most difficult region for approximation and therefore the results are less accurate.

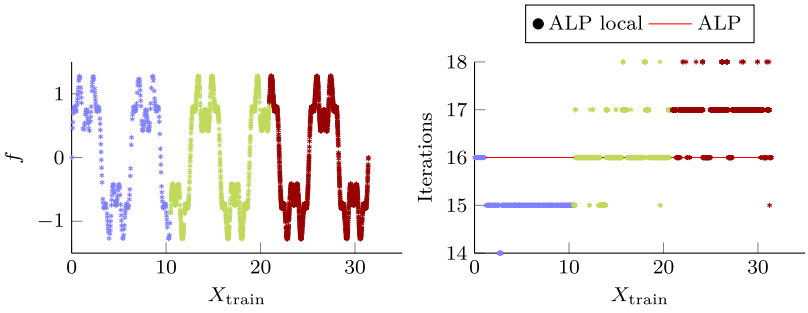


Fig. 4. Left: Representation of the synthetic example over the training set, colored by subsampling region. Right: Stopping number of iterations for the synthetic example over the training set, represented as a constant red line for ALP and as asterisks of different colors depending on region for ALP$_l$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

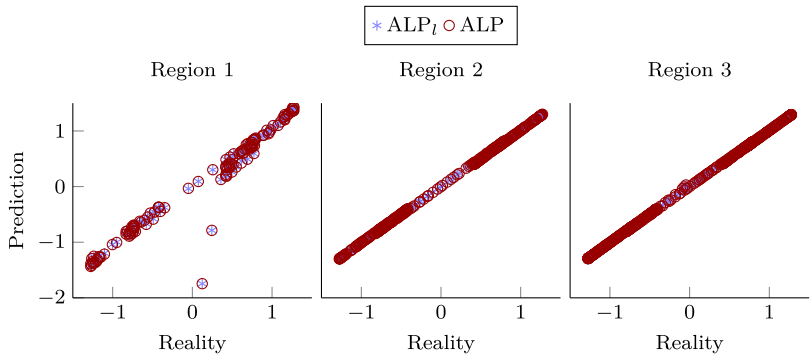


Fig. 5. Sine example test results. Comparison between ALP (red points) and ALP$_l$ (blue points) predictions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

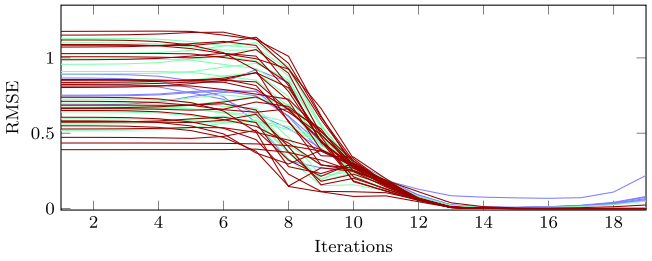


Fig. 6. Training errors of a subset of the sample points. The different colors represent the three subsampling regions as in previous plots, where blue curves correspond to points in the first region, green curves correspond with the second region and red ones with the third region. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Finally, we would like to emphasize that the new ALP$_l$ algorithm prevents overfitting as well as its original version, as explained before. This fact can be appreciated in Fig. 6 where each curve presents the same shape than the ALP curve (like the one shown in Fig. 1). It should be noticed that the red lines, corresponding with points in the last region, represents an error over an easy synthetic problem with lots of subsampled points. Because of this, the error is 0, as the method is able to recover the original function perfectly.

4. Experiments over real data

This section presents numerous interpolation experiments over real datasets, for which we analyze the behavior and properties of ALP and its modified version, ALP$_l$. For this analysis, the results are compared against a k-Nearest Neighbor (k-NN) model, a standard interpolation method. All the real problems presented in the paper will deal with the missing values problem, which is one of the most interesting and more frequent problems in the interpolation context, for being crucial to tackle other problems like classification or regression. The importance of missing values problems is shown in Liu et al. (2016), where missing values are imputed using k-NN and Self-Organizing Map techniques or in Xia et al. (2017), where the classification problem with missing values is solved with a random-forest based algorithm. In this type of problem, when the data is regressed many times against each of the columns, methods like k-NN need to be tuned manually to fit the behavior of each target function while ALP and ALP$_l$ automatically choose a scale that fits the function and data.

For all the experiments we will follow the subsequent methodology:

1. Select the less correlated features among a complete dataset to simulate the missing value problem, if needed.
2. Define 10 independent train-test folds for each case study. We consider 3 possible scenarios: a 90%-10% train-test split, a 80%-20% and a 70%-30%. In total, 30 experiments per problem will be executed.
3. Parameter selection in one of the training folds. As previously explained, the parameters needed for ALP and ALP$_l$ are automatically selected by the algorithm, except in the case of the parameter ν in the local version. Recall that this parameter is used for determining the number of neighbors involved in the point-wise final bandwidth selection for each training point. The value of this parameter is estimated using a 10-fold CV. For the k-NN algorithm, the k neighbors are also obtained via a 10-CV.
4. Run the models in each of the 30 defined experiments.
5. Evaluate the results. The Root Mean Squared Error (RMSE) is used for measuring errors, as previously discussed. Notice that, for making all the datasets errors comparable among them, the errors presented are divided by the standard deviation of the target function in the test set. We will present the median and standard deviation of each set of 10 experiments, together with a statistical significance test between models, in this case a Mann–Whitney U test (Mann and Whitney, 1947) applied over the errors obtained. The null hypothesis of this test states that both models come from continuous distributions with equal medians, rejecting the null hypothesis at the 5% significance level.
6. Derive plots to illustrate the results. When the dimension of the data is large, Principal Component Analysis (PCA Jolliffe, 2002) will be applied for making the visualization possible.

Next, the described procedure is applied to a number of real datasets.

4.1. Wisconsin breast cancer (diagnostic) dataset

The first example selected is a classification UCI dataset (Lichman, 2013): the Wisconsin breast cancer. The features in this dataset are characteristics of each cell nuclei presented on an image of a breast mass. In this work, we changed the original target function and instead we take one of the cell characteristics to be the target function for interpolation, simulating that it has several missing values. The selected feature for being the target function is the less correlated with the other ones, which in this example is the number 12.

As explained in the general case, three different scenarios will be considered: a 90%-10% train-test split, a 80%-20% and a 70%-30%. In all cases, the data has been standardized before building the models. For the ν parameter selection, the grid $\{10, 20, \ldots, 200\}$ was used in the ALP$_l$ case selecting values of 70, 110 and 170 for the 10%, 20% and 30% test splits respectively. The grid $\{1, 2, \ldots, 10\}$ was used for the k-NN method, obtaining $k = 2$ for all the partitions.

The interpolation RMSE errors for this example are shown in Table 1, together with the result of applying the Mann–Whitney U significance test, marking in bold font the winners and ties in first position. Notice that we consider the three different scenarios depending on the number of test samples: 10% of random test samples, 20% or 30%. Observing the results in the table we can conclude that the proposed

Table 1
Normalized RMSE errors for the Breast Cancer example.

	ALP	ALP_l	k-NN
10%	**0.4181 ± 0.1025**	**0.4007 ± 0.0881**	0.4797 ± 0.0666
20%	**0.4194 ± 0.0941**	**0.4265 ± 0.0914**	**0.4580 ± 0.0488**
30%	**0.5431 ± 0.1214**	**0.4517 ± 0.1246**	**0.4673 ± 0.0383**

method, in its original form and in its local version, yields the best results for the 10% case, outperforming k-NN as an interpolation method and ties it when less information is available during training. Even though the error values between ALP and ALP_l are not significant in this example, the ALP_l errors are smaller when there is more training data available. This implies that finer structure in the data is captured with ALP_l when data is at hand (see the 10% case).

For reinforcing these results, we present some plots to visually appreciate the effects of these methods for modeling the considered dataset. In Table 2 we present a comparison between the expected result (the reality, in the x axis) and the interpolated one (the prediction, in the y axis) for ALP_l and ALP. Both methods present similar results, but if we focus our attention on the left lower side of the image, ALP_l points are a bit nearer to the diagonal than ALP ones.

In Fig. 7, the final number of iterations are depicted over the PCA training coordinates (recall that the number of iterations is related to the analysis scale of the data in each region). It can be appreciated that points with similar final number of iterations are located in the same region, so it seems that the number of iterations it is also related with the data structure and the ALP_l method is able to capture this information.

4.2. Wine quality dataset

For another real, but small dataset example, we consider the Wine Quality dataset from the UCI repository. The wine dataset holds several features based on physicochemical tests such as acidity, chlorides or sulfur dioxide from each wine type. There is a separate dataset for red and white variants of wine. We have discarded the discrete features number 2 and 8 from both datasets.

As in the previous example, we change the problem for being a missing values one. The feature with missing values, selected for being the less correlated with the other features, is in this case the feature number 4 for the wine red dataset, that corresponds to the residual sugar, and feature number 10 for the white wine dataset, that corresponds to the sulphates information.

Table 3
Normalized RMSE errors for the Wine examples.

		ALP	ALP_l	k-NN
10%	Red	**0.9190 ± 0.0950**	**0.9072 ± 0.0991**	**0.9354 ± 0.1071**
	White	0.8527 ± 0.0212	**0.8191 ± 0.0405**	0.8704 ± 0.0243
20%	Red	**0.8845 ± 0.0676**	**0.8898 ± 0.0690**	**0.9246 ± 0.0826**
	White	0.8627 ± 0.0197	**0.8293 ± 0.0279**	0.8767 ± 0.0191
30%	Red	**0.8489 ± 0.0501**	**0.8725 ± 0.1057**	0.9043 ± 0.0654
	White	0.8712 ± 0.0110	**0.8540 ± 0.0139**	0.8963 ± 0.0104

We consider again three different scenarios: a 90%-10% train-test split, a 80%-20% and a 70%-30% and, in all cases, the data has been standardized before building the models.

For the parameter selection of the ALP_l algorithm and for the k of k-NN, the used grids where the same than for the first example, obtaining the values of 50, 40 and 50 for the 10%, 20% and 30% test splits respectively in the case of the red wine dataset and $\nu = \{110, 60, 90\}$ for the white wine dataset. for the 10% test case values of $\nu = 70$, for the 20% case $\nu = 110$ and for the 30% test split, $\nu = 170$. For the k parameter, the best values were $k = \{4, 3, 3\}$ for the 10%, 20% and 30% cases respectively for the red wine dataset and $k = 7$ for all the white wine cases.

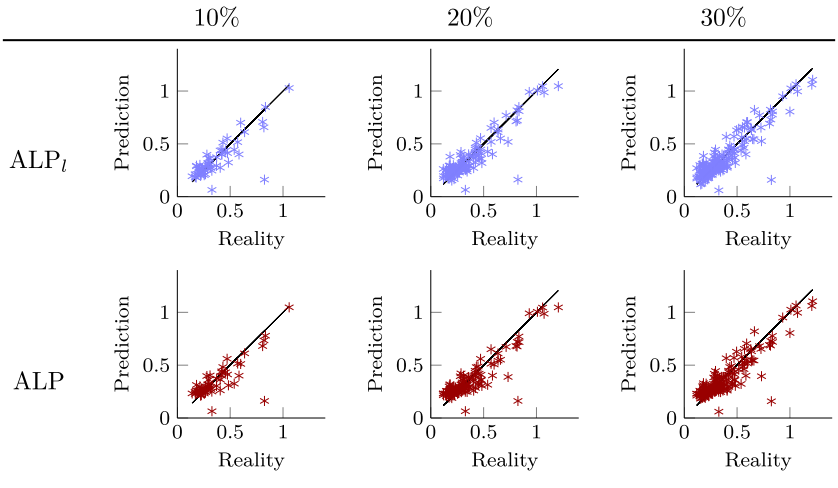
The RMSE errors for both examples are shown in Table 3, where it is also shown the result of the significance test over the errors obtained at a 5% significance level. It can be appreciated that for the red wine variant, there is no significance difference between models except for the 30% case, where the ALP models outperform k-NN. For the 10% and 20% cases, even though there is no significance difference, the errors is smaller also in the ALP models. For the white variant, the ALP_l model is clearly better than the other two, independently of the test subset size.

In Tables 4 and 5 a comparison between the expected result (the target, in the x axis) and the interpolated one (the prediction, in the y axis) for ALP and ALP_l is shown for both examples. The previous conclusions can be also observed in the images: for the red wine variant, both models look almost indistinguishable, while for the white wine the results are clearly better (they follow better the diagonal line).

4.3. Mice protein dataset

The Mice Protein Expression (Higuera et al., 2015) dataset from the UCI repository consists of the expression levels of 77 protein modifications that produce detectable signals in the nuclear fraction of cortex

Table 2
Breast Cancer example. Comparison between ALP (red points) and ALP_l (blue points) predictions over the test set for different test subsets.



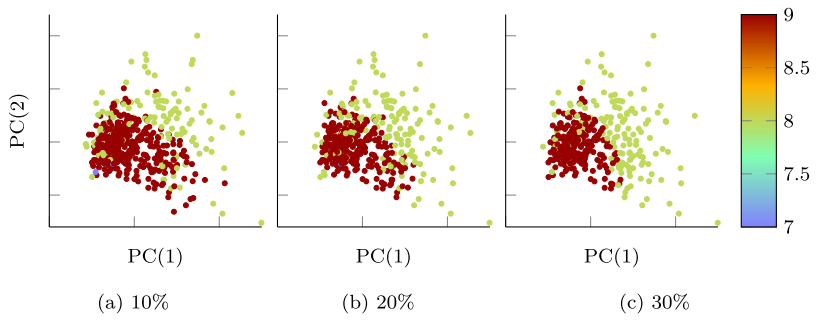


Fig. 7. Breast Cancer Wisconsin (Diagnostic) number of iterations. The image plots the final number of iterations, represented in different colors according to the colorbar, over the PCA components of the training set. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Red wine example. Comparison between ALP (red points) and ALP_l (blue points) predictions over different test subset sizes.

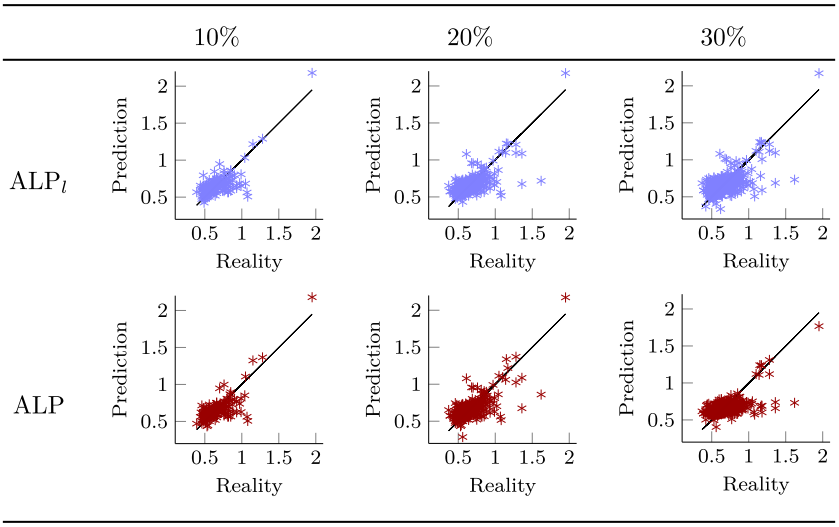
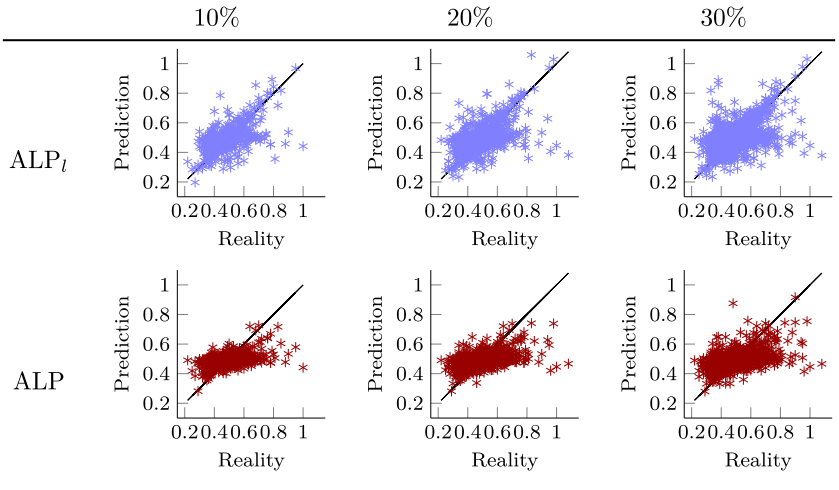


Table 5
White wine example. Comparison between ALP (red points) and ALP_l (blue points) predictions over different test subset sizes.



of mice. The dataset contains a total of 1080 measurements per protein and each measurement can be considered as an independent sample. The original task for this problem is the classification of different types of mice but, as explained before, here the dataset will be used to demonstrate the interpolation capabilities of the proposed methods. Nevertheless, this real-life dataset presents a high rate of missing values, and interpolation methods can be a good solution for filling in these gaps (Rabin and Fishelov, 2017). For measuring the quality of the results, the experimental data will be set as the subset of the 66 features from the entire dataset that have no missing values. From

Table 6
Normalized RMSE errors for the Mice Protein example.

	ALP	ALP_l	k-NN
10%	**0.1883 ± 0.0243**	**0.1874 ± 0.0241**	**0.1962 ± 0.0302**
20%	**0.2020 ± 0.0222**	**0.1998 ± 0.0223**	**0.2122 ± 0.0356**
30%	**0.2133 ± 0.0235**	**0.2133 ± 0.0235**	0.2529 ± 0.0426

this dataset, we will simulate that one feature has some gaps on it. In particular, feature number 52 is chosen as the missing data feature as it is less correlated with the rest of the feature columns in the data.

We consider again three different scenarios: a 90%-10% train-test split, a 80%-20% and a 70%-30% and, in all cases, the data has been standardized before building the models.

For the ν parameter selection of the ALP_l algorithm and for the k of k-NN, the used grids where the same as for the previous examples, obtaining for the 10% test case values of $\nu = 40$, for the 20% case $\nu = 130$ and for the 30% test split, $\nu = 140$. For the k parameter the best value was 2 for all the partitions.

The RMSE errors are shown in Table 6, where the significance test results (at the 5% significance level) are also shown, following the same notation as in the previous examples. In this case there is a tie between the three compared methods in almost all the tested cases, but as in the first example, and even though the difference is not significant, it can be seen that when more data is available (the 10% case) the ALP_l obtains a lower error.

In Table 7 we present a comparison between the expected result (the target, in the x axis) and the interpolated one (the prediction, in the y axis) for ALP and ALP_l. The results obtained with both models look very similar, as expected.

4.4. Seismic data

Seismic data analysis is another real example where interpolation is necessary. In these datasets, temporal data is collected by using a grid of sensors, in this case seismometers, and missing data values is a common problem that occurs when one of the sensors stops to function. The task at hand is a multidimensional interpolation problem, where a complete temporal series that was not recorded needs to be recovered.

The presented examples are from a marine and a land seismic networks, and the data was taken from the New Zealand open database (http://wiki.seg.org/wiki/Open_data). The number of patterns, i.e. the information of each point of the grid, available for these examples (around 70,000) was too big, so a reduced dataset of the first 10,000 coordinates available was prepared for the simulations. The objective function to interpolate here is the entire temporal series of dimension 1252 in the marine case and 2500 in the land one.

Table 8
Normalized RMSE errors for the Seismic examples.

		ALP	ALP_l	k-NN
10%	Marine	**0.2937 ± 0.0221**	**0.2783 ± 0.0099**	0.3299 ± 0.0068
	Land	**0.0989 ± 0.0011**	**0.0989 ± 0.0011**	0.1450 ± 0.0010
20%	Marine	**0.3202 ± 0.0096**	**0.3224 ± 0.0095**	0.3557 ± 0.0078
	Land	**0.1075 ± 0.0008**	**0.1075 ± 0.0008**	0.1567 ± 0.0015
30%	Marine	**0.3342 ± 0.0114**	0.3479 ± 0.0086	0.3776 ± 0.0088
	Land	**0.1187 ± 0.0012**	**0.1187 ± 0.0012**	0.1707 ± 0.0014

In this case, for CV of the number ν of neighbors for the ALP_l algorithm, we have used the knowledge of the grid structure of the data, trying as grid sizes $\{8, 24, 48, 80, 120, 168, 224\}$ (with the idea of taking in each case bigger squares around the point). In the case of the marine seismic dataset we obtain values of $\nu = 168$ for the 10% test partition, $\nu = 24$ for the 20% test partition and $\nu = 8$ for the 30% one, and in the case of the land seismic dataset we obtain $\nu = 120, 168, 168$ respectively. To set the k-NN parameter, the grid $\{1, 2, \ldots, 10\}$ was used, like in the previous examples, obtaining $k = 3$ for all the partitions in both datasets.

Table 8 presents the RMSE errors for the different models, where ALP models again outperform k-NN as interpolation methods, and again we can observe that in general when more train data is available, the ALP_l model has an advantage. This table also presents the results of the significance test applied at the 5% significance level over the errors obtained, remarking the most significant results in bold-faced.

In Tables 9 and 10 the two ALP versions are depicted and also the real objective function for an interval of the temporal series interpolation of one of the patterns. The results in this case are essentially indistinguishable, as both models are basically equal.

In this case, the difference between the ALP versions is neither significant at a 5% significance level but we can conclude that the ALP with local resolution is also able to detect this global structures where just a global σ is needed, offering the same results as the ALP original model. It should be remarked the difference with respect to the k-NN model, being again that the ALP models are a better option for interpolation in this real problem.

Table 7
Mice protein example. Comparison between ALP (red points) and ALP_l (blue points) predictions over the test set for different test subsets.

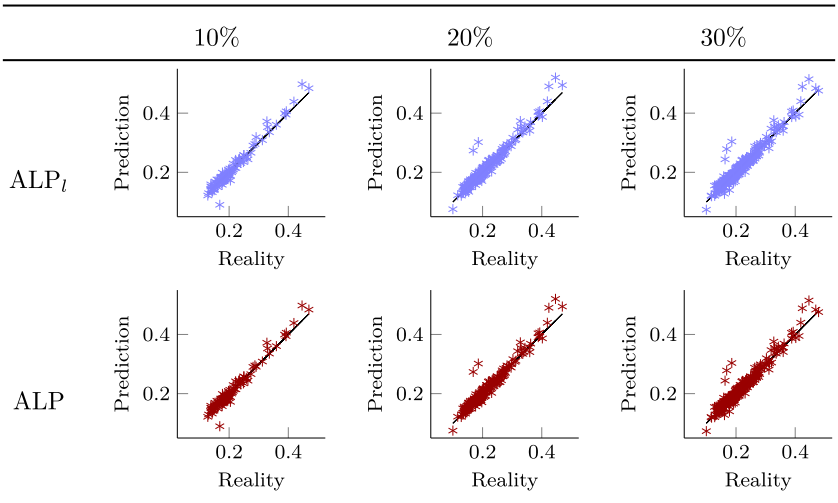


Table 9
Marine seismic example. Comparison between ALP (red points) and ALP$_l$ (blue points) predictions over one test pattern.

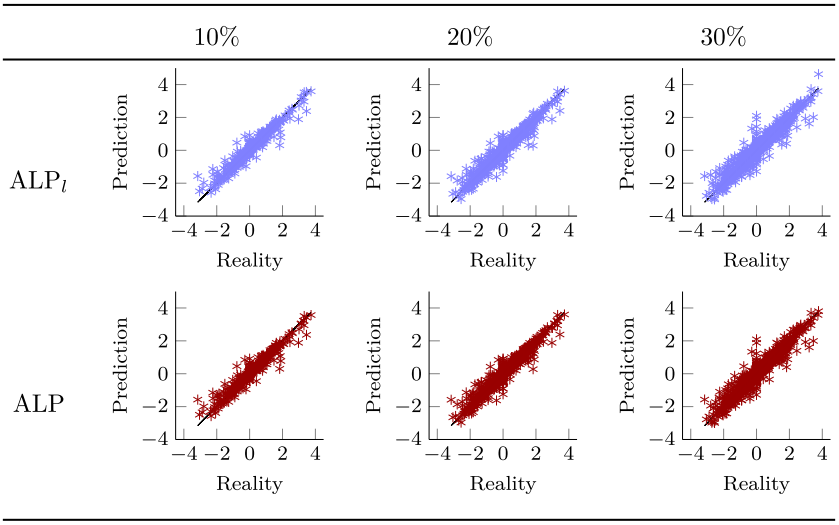
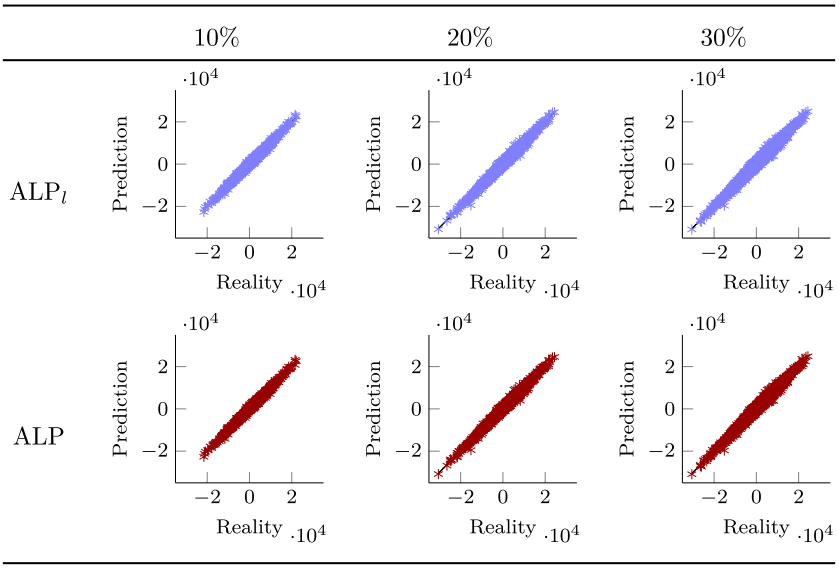


Table 10
Land seismic example. Comparison between ALP (red points) and ALP$_l$ (blue points) predictions over one test pattern.



5. Conclusions

The proposed framework provides novel, flexible and multi-scale analysis tools, which are easy to implement and are suitable for modeling different types of datasets including those with non-uniform data distribution. In particular, we introduce an extended version of the Auto-adaptive Laplacian Pyramids (ALP) where the local structure of the data is taken into account with success, defining a different width of the kernel per point. In addition, this adaptive version of LP training yields, at no extra cost, an estimate of the LOOCV value at each iteration, allowing thus to automatically decide when to stop in order to avoid overfitting.

All together, the proposed methodology provides a robust and flexible framework supported by theoretical error analysis for modeling complex datasets. This work overcomes the limitations of previous LP methods and, thus, applications that utilizes LP will benefit from this research. Indeed, these advantages are already expressed in the experimental results where ALP outperforms k-NN in the different examples presented.

Regarding future work, one challenge we plan to tackle is the global nature of these algorithms, which requires high computational cost as data size grows. This may be addressed by modifying the training step stages of the current algorithm by considering smaller suitable subsamples. Finally, since our method offers a general setting, it can be applied in different domains for out-of-sample extension and forecasting problems.

CRediT authorship contribution statement

Ángela Fernández: Conceptualization, Methodology, Software, Investigation, Visualization, Writing - original draft. **Neta Rabin:** Conceptualization, Methodology, Validation, Writing - original draft. **Dalia Fishelov:** Formal analysis, Supervision, Writing - review & editing. **José R. Dorronsoro:** Conceptualization, Methodology, Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

They wish to thank Prof. Ronald R. Coifman for helpful remarks. They also gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) at Universidad Autónoma de Madrid.

Funding

This work was supported by Spanish grants of the Ministerio de Ciencia, Innovación y Universidades [grant numbers: TIN2013-42351-P, TIN2015-70308-REDT, TIN2016-76406-P]; project CASI-CAM-CM supported by Madri+d [grant number: S2013/ICE-2845]; project FACIL supported by Fundación BBVA (2016); and the UAM–ADIC Chair for Data Science and Machine Learning.

References

Adams, R., Fournier, J., 2003. Sobolev Spaces, Vol. 140. Academic press.
Alexander, R., Zhao, Z., Székely, E., Giannakis, D., 2017. Kernel analog forecasting of tropical intraseasonal oscillations. J. Atmos. Sci. 74 (4), 1321–1342.
Beatson, R., Light, W., 1997. Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines. IMA J. Numer. Anal. 17 (3), 343–372.
Bengio, Y., Paiement, J.-f., Vincent, P., Delalleau, O., Roux, N.L., Ouimet, M., 2004. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In: Advances in Neural Information Processing Systems. pp. 177–184.
Bühlmann, P., Yu, B., 2003. Boosting with the L_2 loss: regression and classification. J. Amer. Statist. Assoc. 98 (462), 324–339.
Buhmann, M., 2003. Radial Basis Functions: Theory and Implementations. In: Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press.
Burt, P., Adelson, E., 1983. The Laplacian Pyramid as a compact image code. IEEE Trans. Commun. 31, 532–540.
Carozza, M., Rampone, S., 2001. An incremental multivariate regression method for function approximation from noisy data. Pattern Recognit. 34 (3), 695–702.
Cawley, G., Talbot, N., 2004. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. Neural Netw. 17 (10), 1467–1475.
Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S., 2002. Choosing multiple parameters for support vector machines. Mach. Learn. 46 (1), 131–159.
Chiavazzo, E., Gear, C.W., Dsilva, C.J., Rabin, N., Kevrekidis, I.G., 2014. Reduced models in chemical kinetics via nonlinear data-mining. Processes 2 (1), 112–140.
Coifman, R., Lafon, S., 2006a. Diffusion maps. Appl. Comput. Harmon. Anal. 21 (1), 5–30.
Coifman, R., Lafon, S., 2006b. Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. Appl. Comput. Harmon. Anal. 21 (1), 31–52.
Comeau, D., Giannakis, D., Zhao, Z., Majda, A.J., 2019. Predicting regional and pan-arctic sea ice anomalies with kernel analog forecasting. Clim. Dynam. 52 (9–10), 5507–5525.
Comeau, D., Zhao, Z., Giannakis, D., Majda, A.J., 2017. Data-driven prediction strategies for low-frequency patterns of North Pacific climate variability. Clim. Dynam. 48 (5–6), 1855–1872.
Do, M., Vetterli, M., 2003. Framing Pyramids. IEEE Trans. Signal Process. 51, 2329–2342.
Dsilva, C.J., Talmon, R., Rabin, N., Coifman, R.R., Kevrekidis, I.G., 2013. Nonlinear intrinsic variables and state reconstruction in multiscale simulations. J. Chem. Phys. 139 (18), 11B608_1.
Duchateau, N., De Craene, M., Sitges, M., Caselles, V., 2013. Adaptation of multiscale function extension to inexact matching: application to the mapping of individuals to a learnt manifold. In: International Conference on Geometric Science of Information. Springer, pp. 578–586.
Duda, R., Hart, P., Stork, D., 2001. Pattern Classification. Wiley, New York.
Elisseeff, A., Pontil, M., 2002. Leave-one-out error and stability of learning algorithms with applications. In: Suykens, J., Horvath, G., Basu, S., Micchelli, C., Vandewalle, J. (Eds.), Advances in Learning Theory: Methods, Models and Applications. In: NATO ASI, IOS Press, Amsterdam, Washington, DC, pp. 152–162.
Fernández, A., Rabin, N., Fishelov, D., Dorronsoro, J., 2016. Auto-adaptive Laplacian Pyramids. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2016. i6doc.com, pp. 59–64.
Fishelov, D., 1990. A new vortex scheme for viscous flows. J. Comput. Phys. 86 (1), 211–224.
Fukunaga, K., Hummels, D., 1989. Leave-one-out procedures for nonparametric error estimates. IEEE Trans. Pattern Anal. Mach. Intell. 11 (4), 421–423.
Hastie, T., Tibshirani, R., Friedman, J., 2008. The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer.
Higuera, C., Gardiner, K., Cios, K., 2015. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. PLOS ONE 10 (6), 1–28.
Jolliffe, I., 2002. Principal Component Analysis. Springer Series in Statistics.
Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. In: IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1137–1143.
Lichman, M., 2013. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, [online].
Liu, L., Gan, L., Tran, T., 2008. Lifting-based Laplacian Pyramid reconstruction schemes. In: ICIP. IEEE, pp. 2812–2815.
Liu, Z., Pan, Q., Dezert, J., Martin, A., 2016. Adaptive imputation of missing values for incomplete pattern classification. Pattern Recognit. 52, 85–95.
Long, A.W., Ferguson, A.L., 2019. Landmark diffusion maps (L-dMaps): Accelerated manifold learning out-of-sample extension. Appl. Comput. Harmon. Anal. 47 (1), 190–211.
M. Li, I.C., Mousazadeh, S., 2014. Multisensory speech enhancement in noisy environments using bone-conducted and air-conducted microphones. In: 2014 IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP). pp. 1–5.
Mann, H., Whitney, D., 1947. On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat. 18 (1), 50–60.
Mishne, G., Cohen, I., 2012. Multiscale anomaly detection using diffusion maps. IEEE J. Sel. Top. Signal Process. 7 (1), 111–123.
Mishne, G., Cohen, I., 2013. Multiscale anomaly detection using diffusion maps. J. Sel. Top. Signal Process. 7 (1), 111–123.
Mishne, G., Cohen, I., 2014a. Multi-channel wafer defect detection using diffusion maps. In: 2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI). IEEE, pp. 1–5.
Mishne, G., Cohen, I., 2014b. Multiscale anomaly detection using diffusion maps and saliency score. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 2823–2827.
Mishne, G., Talmon, R., Cohen, I., 2014. Graph-based supervised automatic target detection. IEEE Trans. Geosci. Remote Sens. 53 (5), 2738–2754.
N. Spingarn, S.M., Cohen, I., 2014. Voice activity detection in transient noise environment using Laplacian pyramid algorithm. In: 2014 14th International Workshop on Acoustic Signal Enhancement (IWAENC). pp. 238–242.
Nadaraya, E., 1964. On estimating regression. Theory Probab. Appl. 9 (1), 141–142.
Rabin, N., Coifman, R., 2012. Heterogeneous datasets representation and learning using Diffusion Maps and Laplacian Pyramids. In: SDM. SIAM / Omnipress, pp. 189–199.
Rabin, N., Fishelov, D., 2017. Missing data completion using diffusion maps and Laplacian Pyramids. In: International Conference on Computational Science and Its Applications. Springer, pp. 284–297.
Rasmussen, C., Williams, C., 2005. Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press.
Wang, J., Liu, G.R., 2002. A point interpolation meshless method based on radial basis functions. Internat. J. Numer. Methods Engrg. 54 (11), 1623–1648.
Watson, G., 1964. Smooth regression analysis. Sankhyā 359–372.
Xia, J., Zhang, S., Cai, G., Li, L., Pan, Q., Yan, J., Ning, G., 2017. Adjusted weight voting algorithm for random forests in handling missing values. Pattern Recognit. 69, 52–60.