Auto-adaptative Laplacian Pyramids for High-dimensional Data Analysis

Ángela Fernández*[1], Neta Rabin†[2], Dalia Fishelov‡[2] and José R. Dorronsoro§[1]

[1]Dpto. Ing. Informática, UAM, 28049, Madrid, Spain
[2]Dept. Exact Sciences, Afeka, Tel-Aviv, 69107, Israel

Abstract

Non-linear dimensionality reduction techniques such as manifold learning algorithms have become a common way for processing and analyzing high-dimensional patterns that often have attached a target that corresponds to the value of an unknown function. Their application to new points consists in two steps: first, embedding the new data point into the low dimensional space and then, estimating the function value on the test point from its neighbors in the embedded space.

However, finding the low dimension representation of a test point, while easy for simple but often not powerful enough procedures such as PCA, can be much more complicated for methods that rely on some kind of eigenanalysis, such as Spectral Clustering (SC) or Diffusion Maps (DM). Similarly, when a target function is to be evaluated, averaging methods like nearest neighbors may give unstable results if the function is noisy. Thus, the smoothing of the target function with respect to the intrinsic, low-dimensional representation that describes the geometric structure of the examined data is a challenging task.

In this paper we propose Auto-adaptive Laplacian Pyramids (ALP), an extension of the standard Laplacian Pyramids model that incorporates a modified Leave One Out cross validation (LOOCV) procedure that avoids the large cost of standard LOOCV and offers the following advantages: (i) it selects automatically the optimal function resolution (stopping time) adapted to the data and its noise, (ii) it is easy to apply as it does not require parameterization, (iii) it does not overfit the training set and (iv) it adds no extra cost compared to other classical interpolation methods. We illustrate numerically ALP's behavior on a synthetic problem and apply it to the computation of the DM projection of new patterns and to the extension to them of target function values on a radiation forecasting problem over very high dimensional patterns.

Keywords: Laplacian Pyramids, LOOCV, Manifold Learning, Diffusion Maps.

1 Motivation

An important challenge in data mining and machine learning is the proper analysis of a given dataset, especially for understanding and working with functions defined over it. In particular, manifold learning algorithms have become a common way for processing and analyzing high-dimensional data and the so called "diffusion analysis" allows us to find the most appropriate geometry to study such functions [25]. These methods are based on the construction of a diffusion operator that depends on the local geometry of the data, which is then used to embed the high-dimensional points into a lower-dimensional space maintaining their geometric properties and, hopefully, making easier the analysis of functions over it. On the other hand, extending functions in such an embedding for new data points may be challenging, either because of the noise or the presence of low-density areas that make insufficient the number of available training points. Also it is difficult to set the neighborhood size for new, unseen points as it has to be done according to the local behavior of the function.

The classical methods for function extension like Geometric Harmonics [8] have parameters that need to be carefully set, and in addition there does not exist a robust method of picking the correct neighborhood in the embedding for function smoothing and evaluation. A first attempt to simplify these approaches was Laplacian Pyramids (LP), a multiscale model that generates a smoothed version of a function in an iterative manner by using Gaussian kernels of decreasing widths [21]. It is a simple method for learning functions from a general set of coordinates and can be also applied to extend embedding coordinates, one of the big challenges in diffusion methods. Recently [1] introduced a geometric PCA based out-of-sample extension for the purpose of adding new points to a set of constructed embedding coordinates.

A naïve way to extend the target function to a new data point could be to find the point's nearest neighbors (NN) in the embedded space and average their function values. The NN method for data lifting was compared in [12] with the LP version that was proposed in [21], and this last method performed better than NN. Buchman et al. [3] also described a different, point-wise adaptive approach, which requires setting the nearest neighborhood radius parameter for every point.

Nevertheless, and as it is often the case in machine learning, when we apply the previous LP model, we can overfit the data if we try to refine too much the prediction during the training phase. In fact, it is difficult to decide when to stop training to obtain good generalization capabilities. A usual approach is to apply the Cross Validation (CV) [13, chap. 9] method to get a validation error and to stop when this error starts to increase. An extreme form of CV is the Leave One Out CV (LOOCV): a model is built using all

*a.fernandez@uam.es
†netar@afeka.ac.il
‡daliaf@post.tau.ac.il
§jose.dorronsoro@uam.es

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 errors. Although LOOCV has a theoretical backing and often yields good results, it has the drawback of a big computational cost, though not in some important cases (see Section 3).

In this paper we propose Auto-adaptive LP (ALP), a modification in the LP training algorithm that merges training and approximate LOOCV in one single phase. To do so we simply build the kernel matrix with zeros in its diagonal. As we shall see, with this change we can implement an LOOCV approximation without any additional cost during the training step. This reduces significantly training complexity and provides an automatic criterion to stop training so that we greatly avoid the risk of severe overfitting that may appear in standard LP. This effect can be observed in Figure 1 when our LP proposal is applied to the synthetic example used in Section 4. The solid and dashed black lines represent the LP training error and the LOOCV error per iteration respectively, and the dashed blue line represents the error for our proposed method. The blue line, that corresponds to the ALP training error attains its minimum at the same iteration prescribed by LOOCV for LP.

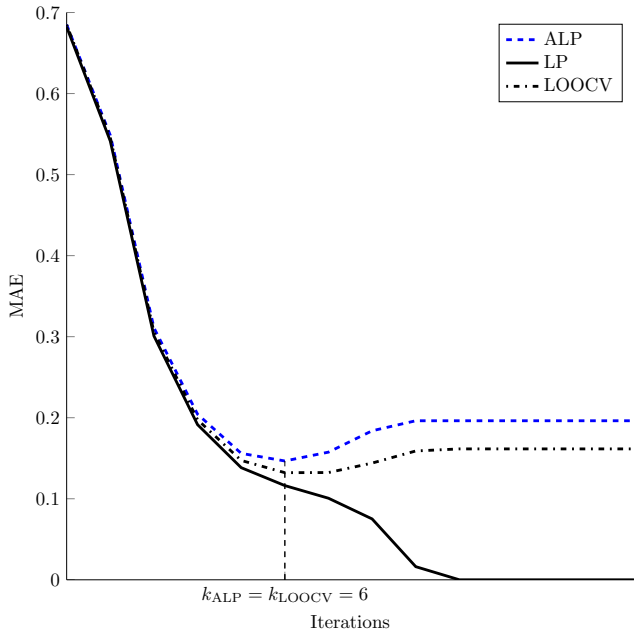


Figure 1: Training and LOOCV errors for the original and modified LP models applied to a synthetic example consisting on a perturbed sine shown in Section 4.

Therefore, ALP doesn't overfit the data and, moreover, doesn't essentially require any parametrization or expert knowledge about the problem, while still achieving a good test error. Moreover, it adds no extra cost compared to other classical neighbor-based interpolation methods.

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 ALP in Section 3, which we apply to a synthetic example in Section 4 and to a real world example in Section 5. The paper ends with some conclusions in Section 6.

2 The Laplacian Pyramids

The Laplacian Pyramid (LP) is an iterative model that was introduced by Burt and Adelson [4] for its application in image processing and, in particular, image encoding. In the traditional algorithm, LP decomposes the input image into a series of images, each of them capturing a different frequency band of the original one. This process is carried out by constructing Gaussian-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 Vetterly [10]) and used for signal processing applications, for example as a reconstruction scheme in [19].

In [21], it was introduced a multi-scale algorithm 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 (recently applied in [20]).

We review next the LP procedure as described in [21] (the down-sampling step, which is part of Burt and Adelson's algorithm is skipped here). Let $S = \{x_i\}_{i=1}^N \in \mathbb{R}^N$ be the sample dataset; the algorithm approximates a function f defined over S by constructing a series of functions $\{\tilde{f}_i\}$ obtained by several refinements d_i over the error approximations. In a slight abuse of language we will use the same notation 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 result of this process gives a function approximation

$$f \simeq \tilde{f} = \tilde{f}_0 + d_1 + d_2 + d_3 + \cdots$$

In more detail, a first level kernel $K^0_\sigma(x, x') = \Phi\left(\text{dist}(x, x')/\sigma\right)$ is chosen using a wide, initial scale σ and where $\text{dist}(x, x')$ denotes some distance function between points in the original dataset. A usual choice and the one we will use here is the Gaussian kernel with Euclidean distances, i.e., to take $\text{dist}(x, x') = \|x - x'\|$ and then

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

where κ is the Gaussian kernel normalizing constant. As before, we will use the K^0 notation for both the general continuous kernel $K^0(x, x')$ and for 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 normalized row-stochastic matrix of the kernel

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

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 we fix a $\mu > 1$, construct at step i a sharper Gaussian kernel P^i with scale σ/μ^i, and the residual

$$d_{i-1} = f - \tilde{f}_{i-1},$$

which captures the error of the approximation to f at the previous $i-1$ step, is used to generate a more detailed representation of f given by

$$\tilde{f}_i = \tilde{f}_{i-1} + d_{i-1} * P^i = \tilde{f}_{i-1} + g_{i-1},$$

with $g_\ell = d_\ell * P^{\ell+1}$. The iterative algorithm stops once the norm of d_i residual vector is smaller than a predefined error. Stopping at iteration L, the final LP model has thus the form

$$\tilde{f}_L = \tilde{f}_0 + \sum_0^{L-1} g_\ell = f * P^0 + \sum_0^{L-1} d_\ell * P^{\ell+1}, \qquad (2)$$

and extending this multi-scale representation to a new data point $x \in \mathbb{R}^N$ is now straightforward because we simply set

$$\begin{aligned}\tilde{f}_L(x) &= f * P^0(x) + \sum_0^{L-1} d_\ell * P^{\ell+1}(x) \\ &= \sum_j f_j P^0(x, x_j) + \sum_1^L \sum_j d_{\ell-1;j} P^\ell(x, x_j).\end{aligned}$$

where we extend the P^ℓ kernels to a new x as

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

The overall cost is easy to analyze. Computing the convolutions $\tilde{f}_0 = f * P^0$, $g_\ell = d_\ell * P^{\ell-1}$ has a $O(N^2)$ cost for a size N sample, while that of obtaining the d_ℓ is just $O(N)$. Thus, the overall cost of L LP steps is $O(LN^2)$.

We observe that if we set a very small error threshold and run afterwards enough iterations, we will end up having $\tilde{f}_\ell = f$ over the training sample. In fact, $\tilde{f}_\ell = \tilde{f}_{\ell-1} + g_{\ell-1}$ and, therefore,

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

with I denoting the identity matrix. Now, if we have $\tilde{f}^\ell \to \phi$, it follows taking limits that

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

i.e., $\phi = f$, for $P^\ell \to I$.

In practice, we will numerically have $P^\ell = I$ as soon as ℓ is large enough so that we have $K^\ell(x_i, x_j) \simeq 0$. We then have $d_{\ell;j} = 0$ for all j and the LP model doesn't change anymore. In other words, care has to be taken when deciding to stop the LP iterations to avoid overfitting. In fact, we show next that when using Gaussian kernels, as we do, the L_2 norm of the LP errors $\tilde{d}_\ell$ decay extremely fast.

First notice that working in the continuous kernel setting, we have $P = K$ for a Gaussian kernel, for then the denominator in (1) is just $\int K(x, z)dz = 1$. Assume that f is in L_2, so $\int_x f^2(x)\, dx < \infty$. The LP scheme is a relaxation process for which in the first step the function f is approximated by $\mathcal{G}^0(f) = f * P^0(x)$. For all ℓ, $P^\ell(x)$ is an approximation to a delta function satisfying

$$\begin{aligned}\int P^\ell(x)\, dx &= 1, \\ \int x P^\ell(x)\, dx &= 0, \\ \int x^2 P^\ell(x)\, dx &\le 2C.\end{aligned} \qquad (3)$$

In the second step f is approximated by $\mathcal{G}^0(f) + \mathcal{G}^1(d_0)$, where $d_0 = \mathcal{G}^0(f) - f$ and $\mathcal{G}^1(d_0) = d_0 * P^1(x)$, and so on. Taking the Fourier transform of $P^\ell(x)$, we have (see [15])

$$\left|\hat{P}^\ell(\omega)\right| \le 1 + \frac{\sigma^2}{2} \int x^2 P^\ell(x)\, dx \le 1 + C\sigma^2, \qquad (4)$$

where we have used (3).

We first analyze the error $d_0(x)$ in the first step, which is defined by $d_0(x) = f * P^0(x) - f$. Taking the Fourier transform of $d_0(x)$ and using (4) we have

$$\left|\hat{d}_0(\omega)\right| = \left|\hat{f}(w)\right| \left|\hat{P}^0(\omega) - 1\right| \le C\sigma_0^2 \left|\hat{f}(\omega)\right|. \qquad (5)$$

The error in the second step is

$$d_1(x) = d_0 - \mathcal{G}_1(d_0) = \big(f * P^0 - f\big) - d_0 * P^1. \qquad (6)$$

Taking the Fourier transform of (6) yields

$$\begin{aligned}\left|\hat{d}_1(\omega)\right| &= \left|\hat{d}_0(\omega) - \hat{d}_0(\omega)\hat{P}^1(\omega)\right| \\ &= \left|\hat{d}_0(\omega)\right| \left|\hat{P}^1(\omega) - 1\right|.\end{aligned}$$

Using (4) and (5) we obtain

$$\left|\hat{d}_1(\omega)\right| \le C \left|\hat{d}_0(\omega)\right| \sigma_1^2 \le C\sigma_0^2\sigma_1^2 \left|\hat{f}(\omega)\right|.$$

Since $\sigma_1 = \frac{\sigma_0}{\mu}$ with $\mu > 1$, then $\left|\hat{d}_1(\omega)\right| \le C\sigma_0^2 \frac{\sigma_0^2}{\mu^2} \left|\hat{f}(\omega)\right|$. Similarly, for the ℓ^{th} step the error is bounded by

$$\left|\hat{d}_\ell(\omega)\right| \le C\sigma_0^2 \left(\frac{\sigma_0^2}{\mu^2}\right)^\ell \left|\hat{f}(\omega)\right|.$$

By Parseval's equality we obtain

$$\|d_\ell\|_{L^2} \le C\sigma_0^2 \left(\frac{\sigma_0^2}{\mu^2}\right)^\ell \|f\|_{L^2}.$$

Thus, the error's L_2 decays faster than any algebraic rate.

We see next how we can estimate a final iteration value L that prevents overfitting without incurring on additional costs.

3 Auto-adaptative Laplacian Pyramids

The standard way to prevent overfitting is to use an independent validation subset and to stop the above ℓ iterations as

soon as the validation error on that subset starts to increase. This can be problematic for small samples and introduces a random dependence on the choice of the particular validation subset; k-fold cross validation is then usually the standard choice, in which we randomly distribute the sample in k subsets, and iteratively use $k-1$ subsets for training and the remaining one for validation. In the extreme case when $k = N$, i.e., we use just one pattern for validation, we arrive at Leave One Out Cross Validation (LOOCV) and stop the training iterations when the LOOCV error starts to increase. Besides its simplicity, LOOCV has the attractive of being an almost unbiased estimator of the true generalization error (see for instance [5, 14]), although with possibly a high variance [18]. In our case LOOCV can be easily applied using for training a $N \times N$ normalized kernel matrix $P_{(p)}$ which is just the previous matrix K where we set to 0 the p-th 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 is $N \times O(LN^2) = O(LN^3)$ cost. 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 [16] or of Ordinary Least Squares ([17], Chapter 7), or approximately for Support Vector Machines [6] or Gaussian Processes [22].

In order to alleviate it, notice first that when we removed x_p from the training sample, the test value at x_p of the $f^{(p)}$ extension built is

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

where $d_\ell^{(p)}$ are the different errors computed using the $P_{(p)}^\ell$ matrices and where $\tilde{P}$ is now just the matrix P with its diagonal elements set to 0, i.e. $\tilde{P}_{i,i} = 0$, $\tilde{P}_{i,j} = P_{i,j}$ when $j \neq i$.

This observation leads to the modification we propose on the standard LP algorithm given in [21], and which simply consist on applying the LP procedure described in Section 2 but replacing the P matrix by its 0-diagonal version $\tilde{P}$, computing then $\tilde{f}_0 = f * \tilde{P}^0$ at the beginning, and $\tilde{d}_\ell = f - \tilde{f}_\ell$, $\tilde{g}_\ell = \tilde{d}_\ell * \tilde{P}^{\ell+1}$ and $\tilde{f}_\ell$ vectors at each iteration. We call this algorithm the Auto-adaptative Laplacian Pyramid (ALP).

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 $f_L^{(p)}(x_p)$. But then we can approximate the square LOOCV error at iteration L as

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

which is just the training error of ALP. 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. The cost of running L steps of ALP is just $O(LN^2)$ and, thus, we gain the advantage of the exhaustive LOOCV without any additional cost on the overall algorithm. The complete training procedure is presented in Algorithm 1 and the test algorithm is shown in Algorithm 2.

Algorithm 1 The ALP Training Algorithm

Input: x_{tr}, y_{tr}, σ_0, μ.
Output: $(\{d_i\}, \sigma_0, \mu, k)$ % The trained model.
$\sigma \leftarrow \sigma_0$; $d_0 \leftarrow y_{\text{tr}}$.
$\tilde{f}_0 \leftarrow 0$; $i \leftarrow 1$.
while $(\text{err}_i < \text{err}_{i-1})$ **do**
  $K \leftarrow e^{-\|x_{\text{tr}} - x_{\text{tr}}\|^2/\sigma^2}$.
  $P_i \leftarrow \text{normalize}(K)$.
  $\tilde{P} \leftarrow P$ with 0-diagonal. % LOOCV.
  $\tilde{f}_i \leftarrow \tilde{f}_{i-1} + d_{i-1} * \tilde{P}_i$.
  $d_i \leftarrow f - \tilde{f}_i$.
  $\text{err}_i \leftarrow d_i/s_{\text{tr}}$, with s_{tr} the number of patterns in x_{tr}.
  $\sigma \leftarrow \sigma/\mu$; $i \leftarrow i + 1$.
end while
$k \leftarrow \min_i d_i$. % Optimal iteration.

Algorithm 2 The ALP Testing Algorithm

Input: x_{tr}, x_{te}, $(\{d_i\}, \sigma_0, \mu, k)$.
Output: $\hat{y}_{\text{te}}$.
$\hat{y}_{\text{te}} \leftarrow 0$; $\sigma \leftarrow \sigma_0$.
for i=0 **to** k-1 **do**
  $K \leftarrow e^{-\|x_{\text{tr}} - x_{\text{te}}\|^2/\sigma^2}$.
  $P_i \leftarrow \text{normalize}(K)$.
  $\hat{y}_{\text{te}} \leftarrow \hat{y}_{\text{te}} + d_i * P_i$.
  $\sigma \leftarrow \sigma/\mu$.
end for

The obvious advantage of ALP is that when we evaluate the training error, we are actually estimating the LOOCV error after each LP iteration. Therefore, the evolution of these LOOCV values tells us which is the optimal iteration to stop the algorithm, i.e., just when the training error approximation to the LOOCV error starts growing. Thus, we do not only remove the danger of overfitting but can also use the training error as an approximation to the generalization error. This effect can be seen in Figure 1 which illustrates the application of ALP to the synthetic problem described in the next section and where the optimum stopping time for ALP is exactly the same that the one that would give the LOOCV error and where training error stabilizes afterwards at 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-adaptative as it does not require costly parameter selection procedures. In fact, choosing as customarily done $\mu = 2$, the only required parameter would be the initial σ but provided it is wide enough, its $\sigma/2^\ell$ scalings will yield an adequate final kernel width.

4 A Synthetic Example

For a better understanding of this theory, we first illustrate the proposed ALP algorithm on a synthetic example of a

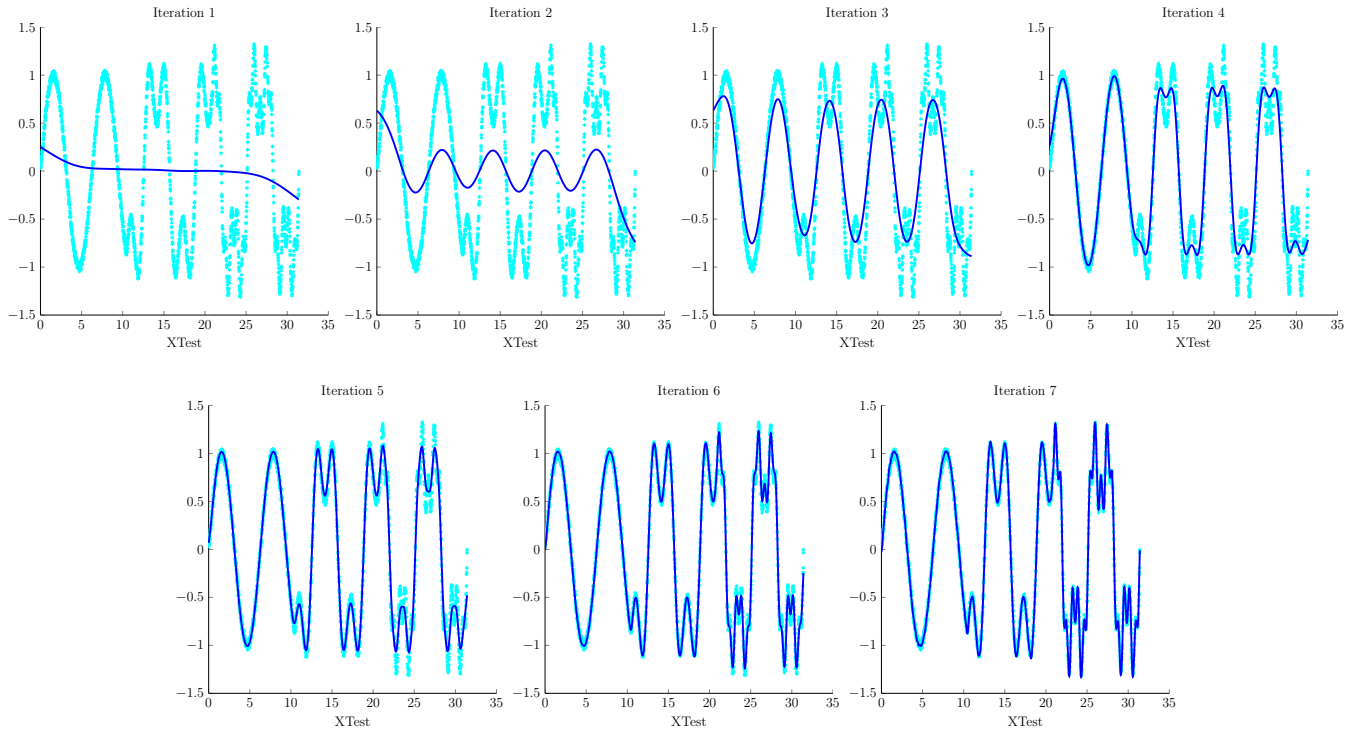


Figure 2: Evolution of the Auto-adaptative Laplacian Pyramids model for the first example.

composition of sines with different frequencies plus additive noise.

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

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

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, we have a single frequency in the interval $[0, 10\pi/3]$, two frequencies in $(10\pi/3, 2 \cdot 10\pi/3]$ and three in $(2 \cdot 10\pi/3, 10\pi]$. We run two different simulations, the first one with $4,000$ points with small $\delta = 0.05$ noise and the second one with $2,000$ points and a larger $\delta = 0.25$ (observe that $|f| \leq 1.75$). In both cases, odd indexed points form the training set and even points form the test set.

Recall 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 Figure 2, which shows the evolution of the prediction of the ALP for small noise experiment. As we can see, at the beginning, the model approximates the function just by a coarse mean of the target function values; however in the subsequent iterations that start using sharper kernels and refined residuals, the approximating function starts capturing the different frequencies and amplitudes of the composite sines. In this particular case the minimum LOOCV value is reached after 7 iterations, a relatively small number which makes sense as we have a simple model with small noise.

When we repeat the same synthetic experiment but now enlarging the amplitude of the uniform distribution to $\delta =$ 0.25, the predicted function is represented in Figure 3 and it is obtained after 6 iterations. As it was to be 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.

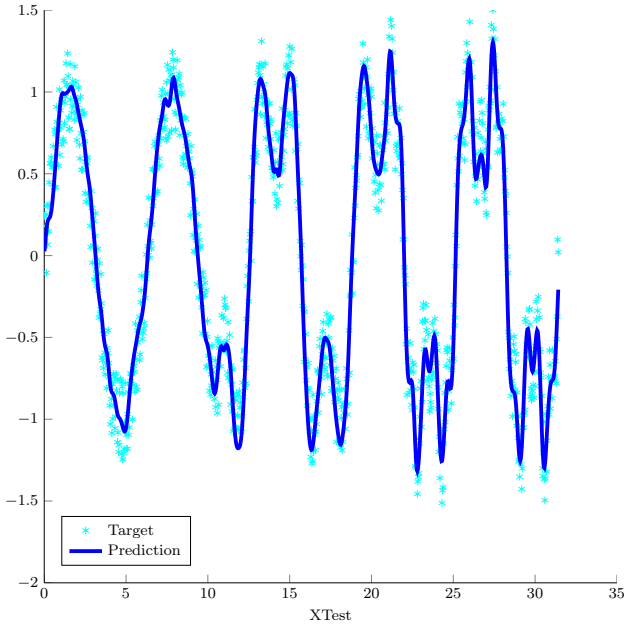


Figure 3: Prediction of the Auto-adaptative Laplacian Pyramids model for the second example.

In any case, we can conclude that the ALP model captures

very well the essential behavior underlying both samples, catching the three different frequencies of the sine and their amplitudes even when the noise level increases.

5 ALP for Eigenvector Estimation in Spectral Clustering and Diffusion Maps

5.1 Spectral Manifold Learning

A common assumption in many big data problems is that although original data appear to have a very large dimension, they lie in a low dimensional manifold $\mathcal{M}$ of which a suitable representation has to be given for adequate modeling. How to identify $\mathcal{M}$ is the key problem in manifold learning, where the preceding assumption has given rise to a number of methods among which we can mention Multidimensional Scaling [9], Local Linear Embedding [23], Isomap [26], Spectral Clustering (SC) [24] and its several variants such as Laplacian Eigenmaps [2], or Hessian eigenmaps [11]. Diffusion methods (DM) [7] also follow this set up and we center our discussion on them.

The key assumption in Diffusion methods is that the metric of the low dimensional Riemannian manifold where data lie can be approximated by a suitably defined diffusion metric. The starting point in DM (and in SC) is a weighted graph representation of the sample with a similarity matrix w_{ij} given by the kernel $e^{-\|x_i - x_j\|^2/2\sigma^2}$. In order to control the effect of the sample distribution, a parameter $\alpha \in [0,1]$ is introduced in DM and we work instead with the α-dependent similarity $w_{ij}^{(\alpha)} = {}^{w_{ij}}/_{g_i^\alpha g_j^\alpha}$, where $g_i = \sum_j w_{ij}$ is the degree of vertex i. The new vertex degrees are now $g_i^\alpha = \sum_j w_{ij}^\alpha$ and we define a Markov transition matrix $\tilde{W}^{(\alpha)} = \{\tilde{w}_{ij}^{(\alpha)} = {}^{w_{ij}^{(\alpha)}}/_{g_i^\alpha}\}$. Fixing the number t of steps considered in the Markov process, we can take into account t neighbors of any point in terms of the t-step diffusion distance given by

$$D_{ij}^t = \|\tilde{w}_{i,\cdot}^{(\alpha;t)} - \tilde{w}_{j,\cdot}^{(\alpha;t)}\|_{L^2({}^1/_{\phi_0})},$$

where ϕ_0 is the stationary distribution of the Markov process and $\tilde{w}^{(\alpha;t)}$ the transition probability in t steps (i.e., the t-th power of $\tilde{w}^\alpha$). We will work with $t = 1$ and use $\alpha = 1$. As it is shown in [7], when $\alpha = 1$, the infinitesimal generator L_1 of the diffusion process coincides with the manifold's Laplace–Beltrami operator Δ and, thus, we can expect the diffusion projection to capture the underlying geometry.

Now, the eigenanalysis of the Markov matrix $\tilde{W}^{(\alpha)}$ gives an alternative representation of the diffusion distance as

$$D_{ij}^t = \sum_k \lambda_k^{2t} (\psi_i^{(k)} - \psi_j^{(k)})^2,$$

with λ_k the eigenvalues and $\psi^{(k)}$ the left eigenvectors of the Markov matrix $\tilde{W}^{(\alpha)}$ and $\psi_i^{(k)} = \psi^k(x_i)$ are the eigenvectors' components [7]. The eigenvalues λ_k decay rather fast, a fact we can use to perform dimensionality reduction by setting a fraction δ of the second eigenvalue λ_1 (the first one λ_0 is always 1 and carries no information) and retaining thus a number $d = d_t$ of those λ_k for which $\lambda_k^t > \delta \lambda_1^t$. This δ is a parameter we have to choose besides the previous α and t. Usual values are either $\delta = 0.1$ or the more strict (and larger dimension inducing) $\delta = 0.01$. Once fixed, we would thus arrive to the diffusion coordinates

$$\Psi = \begin{pmatrix} \lambda_1^t \psi_1(x) \\ \vdots \\ \lambda_d^t \psi_d(x) \end{pmatrix}$$

and we can approximate the diffusion distance as

$$D_{ij}^t \sim \sum_1^d \lambda_k^t (\psi_i^k - \psi_j^k)^2 = \|\Psi(x_i) - \Psi(x_j)\|^2.$$

In other words, the diffusion distance in $\mathcal{M}$ can be approximated by Euclidean distance in the DM projected space. This makes DM a very useful tool to apply procedures in the projected space such as K-means clustering that usually rely on an implicit Euclidean distance assumption. All the steps to compute DM are summarized in Algorithm 3.

While very elegant and powerful, DMs have the drawback of relying on the eigenvectors of the Markov matrix. This makes it difficult to compute the DM coordinates of a new, unseen pattern x. Moreover, the eigenanalysis of the $\tilde{W}^{(\alpha)}$ matrix would have in principle a potentially very high $O(N^3)$ cost, with N sample's size. However, both issues can be addressed in terms of function approximation. The standard approach is to apply the Nyström extension formula [27] but LPs [21] and, hence, ALPs, can also be used in this setting, as we discuss next.

To do so we consider the eigenvalue components $\psi_j^{(k)}$ as the values $\psi^{(k)}(x_j)$ at the points x_j of an unknown function $\psi^{(k)}(x)$ which we try to approximate by an ALP scheme. The general LP formula (2) for the eigenvector $\hat{\psi}^{(k)}$ extended to an out-of-sample point x becomes now

$$\begin{aligned} \hat{\psi}^{(k)}(x) &= \psi^{(k)} * P^0(x) + \sum_0^{L-1} d_\ell * P^{\ell+1}(x) \\ &= \sum_{i=1}^N \mathcal{P}_0(x, x_i)\psi^{(k)}(x_i) + \sum_{h=1}^H \sum_{i=1}^N \mathcal{P}_h(x, x_i) d_h^{(k)}(x_i), \end{aligned} \tag{7}$$

with the differences $d_h^{(k)}$ given now by $d_h^{(k)}(x_i) = \psi^{(k)}(x_i) - \sum_{\ell=0}^{h-1} \hat{(}\psi^{(k)})^{(\ell)}(x_i)$.

We illustrate next the application of these techniques to the analysis of solar radiation data where we relate actual aggregated radiation values with Numerical Weather Prediction (NWP) values.

5.2 Diffusion Maps of Radiation Data

A current important problem that is getting a growing attention in the Machine Learning community is the prediction of renewable energies, particularly solar energy and, therefore, of solar radiation. We will consider here the prediction of the total daily incoming solar energy in a series

Algorithm 3 Diffusion Maps Algorithm.

Input: $\mathcal{S} = \{x_1, \ldots, x_N\}$, the original dataset.
Parameters: t, α, σ, δ.

Output: $\{\Psi(x_1), \ldots, \Psi(x_N)\}$, the embedded dataset.

1: Construct a graph $G = (\mathcal{S}, W)$ where

$$W_{ij} = w(x_i, x_j) = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}.$$

2: Define the initial density function as

$$g(x_i) = \sum_{j=1}^{N} w(x_i, x_j).$$

3: Normalize the weights by the density:

$$w^{(\alpha)}(x_i, x_j) = \frac{w(x_i, x_j)}{g_i^\alpha g_j^\alpha}.$$

4: Define the transition probability

$$\tilde{W}_{ij}^{(\alpha)} = \tilde{w}^{(\alpha)}(x_i, x_j) = \frac{\tilde{w}^{(\alpha)}(x_i, x_j)}{g_i^{(\alpha)}},$$

where $g_i^{(\alpha)} = \sum_{j=1}^{n} w^{(\alpha)}(x_i, x_j)$ is the graph degree.

5: Obtain eigenvalues $\{\lambda_r\}_{r \geqslant 0}$ and eigenfunctions $\{\psi_r\}_{r \geqslant 0}$ of $\tilde{W}^{(\alpha)}$ such that

$$\begin{cases} 1 &= \lambda_0 > |\lambda_1| \geqslant \cdots \\ \tilde{W}^{(\alpha)}\psi_r &= \lambda_r \psi_r. \end{cases}$$

6: Compute the embedding dimension using a threshold $d = \max\{\ell : |\lambda_\ell| > \delta |\lambda_1|\}$.

7: Formulate Diffusion Map:

$$\Psi = \begin{pmatrix} \lambda_1^t \psi_1(x) \\ \vdots \\ \lambda_d^t \psi_d(x) \end{pmatrix}.$$

of meteorological stations located in Oklahoma in the context of the AMS 2013-2014 Solar Energy Prediction Contest hosted by the Kaggle company.[1] While the ultimate goal would be here to obtain best predictions, we will use the problem to illustrate the application of ALP in the previously described DM setting.

The input data are an ensemble of 11 numerical weather predictions (NWP) from the NOAA/ESRL Global Ensemble Forecast System (GEFS). We will just use the main NWP ensemble as being the one with the highest probability of being correct. Input patterns contain five time-steps (from 12 to 24 UCT-hours in 3 hour increments) with 15 variables per time-step for all points in a 16×9; each pattern has thus a very large $10,800$ dimension. The NWP forecasts from 1994–2004 yield $4,018$ training patters and the years 2005, 2006 and 2007, with $1,095$ patterns, are used for testing.

Our first goal is to illustrate how applying ALP results in good approximations to the DM coordinates of new points that were not available for the first eigenanalysis of the initial Markov matrix. To do so, we normalize training data to 0 mean and a standard deviation 1 and use the DM approach explained above, working with a Gaussian kernel, whose σ parameter has been established as the 50% percentile of the Euclidean distances between all sample points. We recall that we fix the diffusion step t to 1 and also the α parameter, so that data density does not influence diffusion distance computations. To decide the best dimensionality for the embedding, the precision parameter δ has been fixed at a relatively high value of 0.1, i.e., we only keep the eigenvalues that are bigger than the 10% of the first non trivial eigenvalue. This choice yields an embedding dimension of 3, which enables to visualize the results. We apply DM with these parameters over the training set and obtain the corresponding eigenvectors, i.e., the sample values $\psi^{(k)}(x_i)$ over the training patterns x_i, and the DM coordinates of the training points. We then apply Algorithm 1 to decide on the ALP stopping iteration and to compute the differences $d_h^{(k)}(x_i)$ in (7) and finally apply Algorithm 2 to obtain the approximations to the values of $\psi^{(k)}$ over the test points.

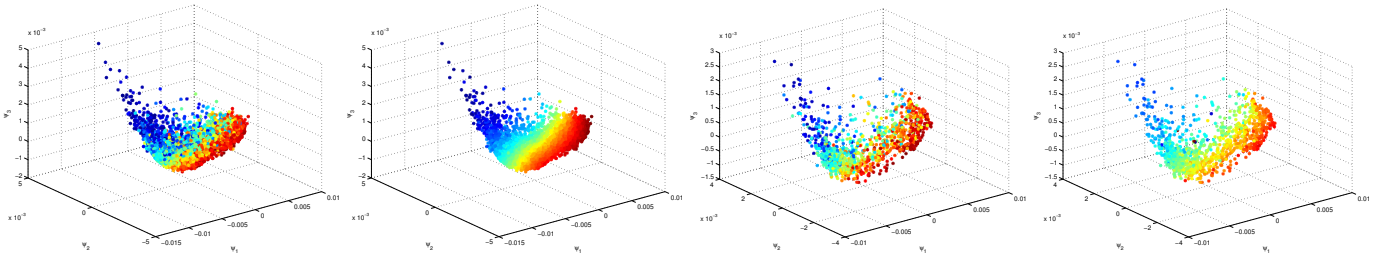
In order to measure the goodness of the new coordinates, we have also performed the DM eigenanalysis to the entire NWP data, i.e., to the training and test inputs together. In this way we can compare the extended embedding obtained using ALP with the one that would have been obtained if we had known the correct diffusion coordinates.

5.3 Results

For this experiments the results have been obtained computing a DM embedding only over the training sample, and using ALP first to extend DM to the test sample and then for radiation prediction. In Figure 5 training and test results are shown. The three diffusion coordinates for this example are colored by the target, i.e. by the solar radiation (first and third), or by the prediction given by ALP (second and fourth). In the first and third image we can see that DM captures the structure of the target radiation in the sample, with the low radiation data (blue) appearing far apart from the points with high radiation (red). If we compare this with the prediction-colored embeddings (second and fourth) we can observe that the radiation values have been smoothed across color bands and that the general radiation trend is captured approximately along the second DM feature even if not every detail is modeled (recall that measured radiation is the target value and, thus, is not included in the DM transformation). This behavior can be observed for the training points in first and second plots but also for the test ones in the third and fourth ones, where it can be seen that the ALP method makes a good extension of the target function for new points.

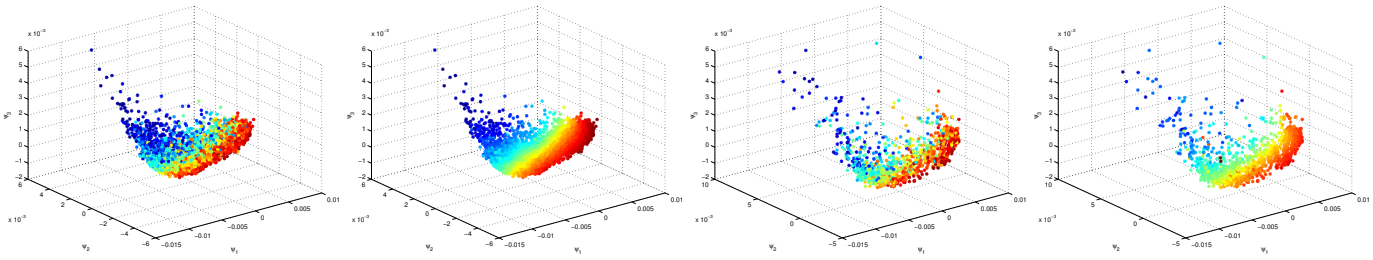
For comparison purposes training and test ALP results

[1] American Meteorological Society 2013-2014 Solar Energy Prediction Contest (`https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest`).



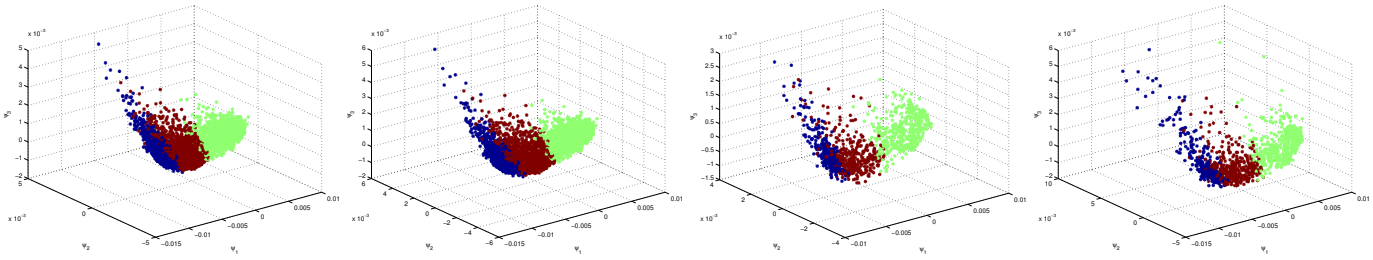
(a) Training Radiation. (b) Training ALP Prediction. (c) Test Radiation. (d) Test ALP Prediction.

Figure 4: Training and test results when the DM embedding is computed over the entire sample and ALP used only for radiation prediction.



(a) Training Radiation. (b) Training ALP Prediction. (c) Test Radiation. (d) Test ALP Prediction.

Figure 5: Training and test results when the DM embedding is computed only on the training sample and ALP is used first to extend DM to the test sample and then for radiation prediction.



(a) Training DM Full. (b) DM Training Sample. (c) Test DM Full. (d) Test Expanded by ALP.

Figure 6: Clusters over the embedded data.

when the DM embedding is computed over the entire sample are shown in Figure 4 (we could consider this as the "true" DM embedding). Comparing with Figure 5 we can see that the two DM embeddings are very similar and that the target and prediction colors seem to be more or less the same. This shows that when we apply ALP to compute the DM coordinates of new test sample points we get an embedding quite close to the ideal one obtained jointly over the train and test patterns.

In order to give a quantitative measure of the quality of the ALP projections, we perform Euclidean K-means with $K = 3$ over the three-dimensional embeddings of the test sample. We want to compare the clusters obtained over the ideal DM embedding computed with the entire sample and the ones obtained over the train DM embedding and then extended for the test coordinates. The resulting clusterings can be seen in figure 6. Notice that these clusters do not reflect radiation structure; instead more weight is apparently given to the first DM feature (that should have the largest feature values and, thus, a bigger influence when Euclidean distances are computed). Anyway, recall that this embedding was made with 15 different variable types, from which just 5 are radiation variables. Because of this, the cluster structure doesn't have to reflect just radiation's effect on the embedding, but the overall variable behavior.

To obtained a concrete metric we will compare the cluster assignments of the test points over the extended embedding (the "predicted" assignments) with those made over the full embedding (the "true" assignments). Looking at this as a classification problem, the accuracy, i.e., the percentage of test points assigned to their "real" clusters, measures how well the ALP extensions match the "true" embedding structure. As the confusion matrix in Table 1 shows, ALP does this quite well, with a total accuracy of 97.53%. Thus, if clustering of the embedded features is desired, ALP makes it possible for new patterns with a very high accuracy.

Table 1: Confusion Matrix of the K-means classification for the extended test coordinates.

	P. 1	P. 2	P. 3	$\sum$
R. 1	294	4	0	298
R. 2	9	342	2	353
R. 3	0	12	432	444
$\sum$	303	358	434	1095

Getting back to radiation prediction over the test sample, once the embedding is obtained, we try to predict over it the total daily incoming solar energy using now a two step ALP procedure. To do so, DM are applied over the training sample to obtain the DM embedded features and then a first ALP model $\mathcal{ALP}_F$ is built to extend the DM features to the test sample and a second ALP $\mathcal{ALP}_R$ to build the ALP function approximation to the target radiation. Given a new NWP test pattern, we apply $\mathcal{ALP}_F$ to obtain its extended DM features and, next, $\mathcal{ALP}_R$ over them to obtain the final radiation prediction.

In Figure 8, left, we have depicted for the second test year the real radiation in light blue and the ALP prediction in dark blue. Although the winning models in the Kaggle competition followed different approaches, it can be seen that ALP captures radiation's seasonality. In the right plot we zoom in and it can be appreciated how ALP tracks the radiation variations. Even if not every peak is caught, ALP yields a reasonably good approximation to actual radiation without requiring any particular parameter choices nor any expert knowledge about the problem we wanted to address. Figure 7 shows the evolution of standard LP training error, its associated LOOCV error and the LOOCV estimation given by ALP when they are applied for radiation forecasting. It illustrates the robustness of the ALP model against overfitting. Again, the ALP model requires here the same number of 15 iterations suggested by applying full LOOCV to standard LP.

Finally, Figure 9 shows for a test point x, plotted as a black dot, the evolution as ALP advances of the influence of sample points on x, larger for red points and smaller for blue ones. As it can be seen, as σ gets smaller, the number of high influence red points also decreases sharply and so does the possibility of overfitting.

6 Conclusions

The classical Laplacian Pyramid scheme of Burt and Adelson have been widely studied and applied to many problems, particularly in image processing. However, it has the risk of overfitting and, thus, requires the use of rather costly techniques such as cross validation to prevent it.

In this paper we have presented Auto-adaptative Laplacian Pyramids (ALP), a modified, adaptive version of LP training that 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. We have illustrated the robustness of the ALP method over a synthetic example and shown on a real radiation problem how to use it to extend Diffusion Maps embeddings to new patterns and to provide simple, yet reasonably good radiation predictions.

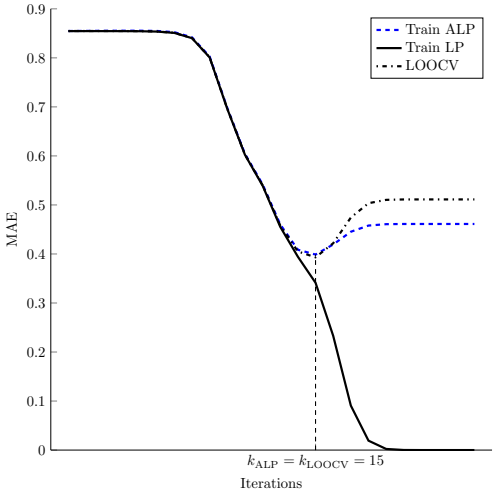


Figure 7: Training error for the solar example of ALP, standard LP and its associated LOOCV.

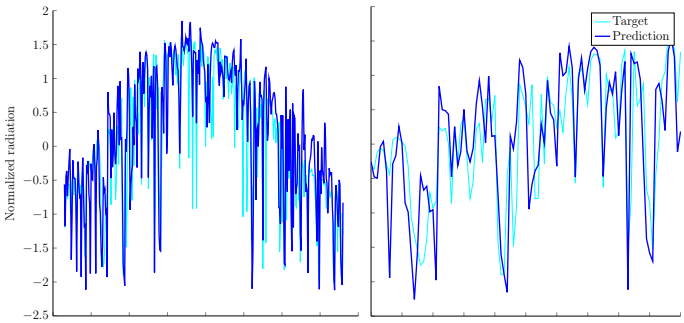


Figure 8: Prediction of the daily incoming solar energy over the second test year, and in a zoom over 100 days.

Acknowledgments

The authors wish to thank Prof. Yoel Shkolnisky and Prof. Ronald R. Coifman for helpful remarks. The first author acknowledge partial support from grant TIN2010-21575-C02-01 of Spain's Ministerio de Economía y Competitividad and the UAM–ADIC Chair for Machine Learning in Modeling and Prediction.

References

[1] Y. Aizenbud, A. Bermanis, and A. Averbuch. PCA-based out-of-sample extension for dimensionality reduction. Submitted. `http://www.cs.tau.ac.il/~amir1/PS/pbe.pdf`.

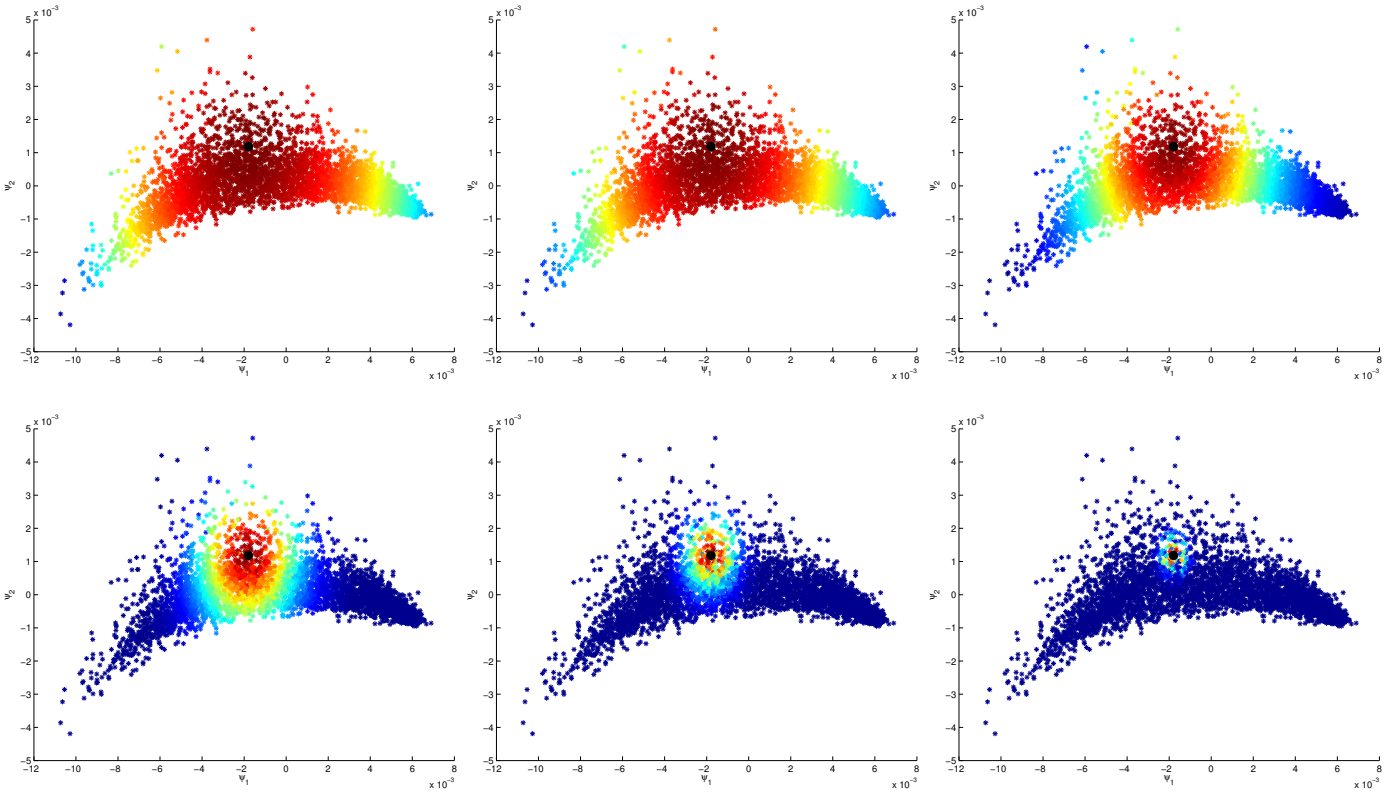


Figure 9: Evolution of the neighborhood of a test point over the embedded training points.

[2] M. Belkin and P. Nyogi. Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[3] S.M. Buchman, A.B. Lee, and C.M. Schafer. High-dimensional density estimation via SCA: An example in the modelling of hurricane tracks. *Statistical Methodology*, 8(1):18–30, 2011.

[4] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

[5] G.C. Cawley and N.L.C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17(10):1467–1475, 2004.

[6] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

[7] R.R. Coifman and S. Lafon. Diffusion Maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[8] R.R. Coifman and S. Lafon. Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21(1):31–52, 2006.

[9] T.F. Cox and M.A.A. Cox. *Multidimensional Scaling*. Chapman and Hall/CRC, 2000.

[10] M.N. Do and M. Vetterli. Framing pyramids. *IEEE Transactions on Signal Processing*, 51:2329–2342, 2003.

[11] D.L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences of the United States of America*, 100(10):5591–5596, 2003.

[12] C.J Dsilva, R. Talmon, N. Rabin, R.R. Coifman, and I.G. Kevrekidis. Nonlinear intrinsic variables and state reconstruction in multiscale simulations. *J. Chem. Phys.*, 139(18), 2013.

[13] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, New York, 2001.

[14] A. Elisseeff and M. Pontil. Leave-one-out error and stability of learning algorithms with applications. In J. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in learning theory: methods, models and applications*, NATO ASI. IOS Press, Amsterdam; Washington, DC, 2002.

[15] D. Fishelov. A new vortex scheme for viscous flows. *Journal of computational physics*, 86(1):211–224, 1990.

[16] K. Fukunaga and D.M. Hummels. Leave-one-out procedures for nonparametric error estimates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(4):421–423, 1989.

[17] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2008.

[18] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[19] L. Liu, L. Gan, and T.D. Tran. Lifting-based laplacian pyramid reconstruction schemes. In *ICIP*, pages 2812–2815. IEEE, 2008.

[20] G. Mishne and I. Cohen. Multiscale anomaly detection using diffusion maps. *J. Sel. Topics Signal Processing*, 7(1):111–123, 2013.

[21] N. Rabin and R.R. Coifman. Heterogeneous datasets representation and learning using Diffusion Maps and Laplacian Pyramids. In *SDM*, pages 189–199. SIAM / Omnipress, 2012.

[22] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[23] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[24] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):888–905, 2000.

[25] A.D. Szlam, M. Maggioni, and R.R. Coifman. Regularization on graphs with function-adapted diffusion processes. *J. Mach. Learn. Res.*, 9:1711–1739, 2008.

[26] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensional ity reduction. *Science*, 290(5500):2319–2323, 2000.

[27] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, volume 14, pages 682–688. MIT Press, 2001.