A framework for online prediction using kernel adaptive filtering

Eder Arley León Gómez
A framework for online prediction using kernel adaptive filtering

Eder Arley León Gómez

Thesis or degree work presented as a partial requirement for the title of:
Master in Engineering - Electrical Engineering

Advisor:
Ph.D. César Germán Castellanos Domínguez

Co-advisor:
Ph.D. Sergio Garcia Vega

Line of research:
Machine learning

Investigation Group:
Control and Digital Signal Processing Group

National University of Colombia
Faculty of Engineering and Architecture
Department of Electrical Engineering, Electronics and Computing
Manizales, Colombia
2019
Marco de predicción en línea usando filtros adaptativos kernel

Eder Arley León Gómez

Trabajo de grado presentado como requisito parcial para el título de:
Magister en Ingeniería - Ingeniería Eléctrica

Director:
Ph.D. César Germán Castellanos Domínguez
Co-director:
Mg. Sergio García Vega

Línea de investigación:
Aprendizaje de Máquina
Grupo de investigación:
Grupo de Control y Procesamiento Digital de Señales

Universidad Nacional de Colombia
Facultad de Ingeniería y Arquitectura
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Manizales, Colombia
2019
Dedication

To all my family

"The culture and education are fundamental for a free Colombia"
Eder Arley León Gómez
Abstract

Nowadays, the task of predicting in schemas online is an essential field of study for machine learning. The Filters Adaptive based on kernel methods have taken an essential role in this type of task; this is primarily due to their condition of universal approximation, their ability to solve nonlinear problems and the modest computing cost they possess. However, although they have significant advantages with similar methods, they present different challenges to be solved such as: (1) the tuning of the kernel bandwidth parameters and the learning rate; (2) the limitation in the model size, product of the number of elements that the filtered dictionary may contain; and, (3) the efficient construction and modeling of multiple filters. The improvement of these conditions will allow an improvement in the representation of time series dynamics, which translates into a decrease in prediction error.

This thesis document addresses the previous issues raised from three proposals. The first is through the interactive search for adequate kernel bandwidth and learning rate, which is achieved by minimizing the correntropy within a proposed cost function. The second contribution corresponds to a scheme of sequential construction of filters, which unlike other methods of state of the art, does not restrict the samples to a single dictionary, and that additionally updates the weights of the samples shared in several filters. The third and last one corresponds to the integration of a kernel bandwidth update method with another that sequentially builds a filter bank. These different proposed frameworks were validated in synthetic data sets as in the real world. The results, in general, show an improvement in the convergence rate, the reduction of the mean square error and the size of the dictionary with different filters of state of the art and a neural network for a specific case.

Keywords: machine learning, forecasts, kernel adaptative filtering, dictionary, learning rate, kernel bandwidth, clustering adaptive.
Resumen

La tarea de predicción en esquemas secuenciales en línea, es hoy un importante campo de estudio para el aprendizaje de máquina. Los Filtros Adaptativos basados en métodos kernel han tomado un papel importante para este tipo de tareas, esto se debe en gran medida a su condición de aproximación universal, su capacidad de solucionar problemas no lineales y al modesto costo computación que poseen. Sin embargo, aunque tienen ventajas significativas con métodos similares, presentan diferentes desafíos a solucionar como: (1) la sintonización de los parámetros del ancho de banda del kernel y la tasa de aprendizaje; (2) la limitación en el tamaño de modelo, producto del número de elementos que pueda contener el diccionario del filtro; y, (3) la eficaz construcción y modelamiento de múltiples filtros. El mejoramiento de estas condiciones permitirá una mejora en la representación de las dinámicas de series de tiempo, lo que se traduce en una disminución del error de predicción.

Este documento de tesis aborda las problemáticas anteriores planteadas a partir de tres propuestas. La primera es vía de la búsqueda interactiva de un adecuado ancho de banda del kernel y tasa de aprendizaje, lo cual se logra mediante la minimización de la correntropía dentro de una función de costos propuesta. El segundo aporte corresponde a un esquema de construcción secuencial de filtros, que a diferencia de otros métodos del estado del arte, no restringe las muestras a un único diccionario, y que adicionalmente actualiza los pesos de las muestras compartidas en varios filtros. La tercera y última, corresponde a la integración de un método de actualización del ancho de banda del kernel con otro que construye secuencialmente un banco de filtros. Estos distintos marcos propuestos, fueron validados en conjuntos de datos sintéticos como del mundo real. Los resultados en general presentan una mejora en la tasa de convergencia, la reducción del error cuadrático medio y el tamaño del diccionario con diferentes filtros del estado del arte y un red neuronal para un caso específico.

Palabras clave: aprendizaje de máquina, predicción, filtros adaptativos kernel, diccionario, tasa de aprendizaje, ancho de banda del kernel, agrupamiento adaptativo.
Content

Abstract

Resumen

Nomenclature

List of tables ................................................................. XI
List of figures ................................................................. XI
List of algorithms ............................................................ XII

1. Preliminaries 2
   1.1. Introduction .......................................................... 2
   1.2. Review of approaches ............................................... 2
   1.3. Problems statement .................................................. 4
   1.4. Objectives ................................................................ 4
       1.4.1. General objective ............................................... 4
       1.4.2. Specifies objectives ............................................. 4
   1.5. Contribution ............................................................ 5
   1.6. Publications ............................................................ 5
   1.7. Dataset .................................................................. 6
       1.7.1. Synthetics ......................................................... 6
       1.7.2. Wind speed ....................................................... 6
       1.7.3. Price kWh into Colombia energy market ................. 6

2. Mathematical background 8
   2.1. Online prediction of time series from kernel methods .......... 8
   2.2. Kernel Adaptative Filterings ......................................... 8
       2.2.1. Least Mean Square (LMS) ..................................... 9
       2.2.2. Non-linear regression in RKHS ............................. 11
       2.2.3. Kernel Leats Mean Square ................................. 12
       2.2.4. Quantized Kernel Leats Mean Square ................... 14

3. Online sequential update of parameters 16
   3.1. Background mathematical ............................................. 17
       3.1.1. Descending gradient method ............................... 17
# Nomenclature

## Acronyms

<table>
<thead>
<tr>
<th>Finished</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>ALD</td>
<td>Approximate Linear Dependency</td>
</tr>
<tr>
<td>AR</td>
<td>Autoregressive</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive Mobile Average</td>
</tr>
<tr>
<td>ARIMA</td>
<td>Autoregressive Integrated Moving Average</td>
</tr>
<tr>
<td>CC</td>
<td>Coherence Criterion</td>
</tr>
<tr>
<td>CV</td>
<td>Cross-Validation</td>
</tr>
<tr>
<td>DNA</td>
<td>Does Not Apply</td>
</tr>
<tr>
<td>Ex-KRLS</td>
<td>Extended Kernel Recursive Least Square</td>
</tr>
<tr>
<td>IDEAM</td>
<td>Instituto de Hidrología, Meteorología y Estudios Ambientales</td>
</tr>
<tr>
<td>KAF</td>
<td>Kernel Adaptive Filtering</td>
</tr>
<tr>
<td>KFDA</td>
<td>Kernel Fisher Discriminant Analysis</td>
</tr>
<tr>
<td>kWh</td>
<td>kilowatt hour</td>
</tr>
<tr>
<td>KLMS</td>
<td>Kernel Least Mean Square</td>
</tr>
<tr>
<td>KPCA</td>
<td>Kernel Principal Component Analysis</td>
</tr>
<tr>
<td>KRLS</td>
<td>Kernel Recursive Least Square</td>
</tr>
<tr>
<td>KRN</td>
<td>Kernel Regularization Network</td>
</tr>
<tr>
<td>NICE</td>
<td>Nearest Instance Centroid Estimation</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>RKHS</td>
<td>Reproducing Kernel Hilbert Space</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>QKLMS</td>
<td>Quantized Kernel Least Mean Square</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>UPME</td>
<td>Unidad de Planeación Minero Energética</td>
</tr>
<tr>
<td>VQ</td>
<td>Vector Quantification</td>
</tr>
</tbody>
</table>
Functions

<table>
<thead>
<tr>
<th>Notation</th>
<th>Denomination</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;.,.&gt;</td>
<td>Inner product</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>\nabla</td>
<td>Gradient</td>
</tr>
<tr>
<td>\top</td>
<td>Transpose of a matrix</td>
</tr>
<tr>
<td>\varphi(.)</td>
<td>A mapping induced by a reproducing kernel</td>
</tr>
<tr>
<td>\varphi(t)</td>
<td>Transformed filter input at time or iteration t</td>
</tr>
<tr>
<td>J_w</td>
<td>Cost function</td>
</tr>
<tr>
<td>J(\theta_0, \theta_1)</td>
<td>Function to minimizate</td>
</tr>
<tr>
<td>V_\lambda(X, Y)</td>
<td>Correntropy</td>
</tr>
<tr>
<td>exp</td>
<td>Exponential function</td>
</tr>
</tbody>
</table>

Variables

<table>
<thead>
<tr>
<th>Notation</th>
<th>Denomination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mathcal{N}</td>
<td>Prediction horizon</td>
<td>\mathcal{N} \in \mathbb{N}</td>
</tr>
<tr>
<td>\mathcal{L}</td>
<td>Time delay</td>
<td>\mathcal{L} \in \mathbb{R}</td>
</tr>
<tr>
<td>\eta</td>
<td>Learning rate</td>
<td>\eta \in \mathbb{R}</td>
</tr>
<tr>
<td>\sigma</td>
<td>Bandwidth kernel</td>
<td>\sigma \in \mathbb{R}</td>
</tr>
<tr>
<td>\lambda</td>
<td>Bandwidth correntropy</td>
<td>\lambda \in \mathbb{R}</td>
</tr>
<tr>
<td>x_t</td>
<td>Input vector</td>
<td>x_t \in \mathbb{R}^{1 \times L}</td>
</tr>
<tr>
<td>y</td>
<td>KAF error at time t</td>
<td>y \in \mathbb{R}</td>
</tr>
<tr>
<td>\hat{y}_t</td>
<td>Real value</td>
<td>\hat{y}_t \in \mathbb{R}</td>
</tr>
<tr>
<td>e_t</td>
<td>Output estimation error at time or iteration i</td>
<td>e_t \in \mathbb{R}</td>
</tr>
<tr>
<td>C_t</td>
<td>Dictionary or center set at iteration t</td>
<td>C_t \in \mathbb{R}^{m \times m}</td>
</tr>
<tr>
<td>w_t</td>
<td>Weight estimate at time or iteration i</td>
<td>w_t \in \mathbb{R}</td>
</tr>
<tr>
<td>\phi</td>
<td>Quantification threshold</td>
<td>\phi \in \mathbb{R}^+</td>
</tr>
<tr>
<td>\delta</td>
<td>Centroid quantification threshold</td>
<td>\delta \in \mathbb{R}^+</td>
</tr>
<tr>
<td>\zeta</td>
<td>Quantification threshold between samples;</td>
<td>\zeta \in \mathbb{R}^+</td>
</tr>
<tr>
<td>\kappa(C_t, x_t)</td>
<td>Kernel function evaluated at C_t and u_t</td>
<td>\kappa(C_t, x_t) \in \mathbb{R}^+ (0, 1)</td>
</tr>
<tr>
<td>g_i</td>
<td>i-th centroid</td>
<td>g_i \in \mathbb{N}</td>
</tr>
<tr>
<td>G_i</td>
<td>i-th center of cluster i</td>
<td>G_i \in \mathbb{R}^{1 \times L}</td>
</tr>
<tr>
<td>\mathcal{G}</td>
<td>Set of cluster</td>
<td>\mathcal{G} \in \mathbb{N}</td>
</tr>
</tbody>
</table>
List of tables

1-1. Wind speed time serie. .................................................. 6
1-2. Price kWh into Colombia energy market .............................. 7
3-1. Configuration and error final for Mackey-Glass .................. 21
3-2. Configuration and error final for Wind speed ..................... 23
4-1. Mean Square Error for Time serie of Lorenz ...................... 32
4-2. Mean Square Error for Time Serie of price of the Colombian energy market. 33
List of figures

2-1. Schematic diagram of a Kernel Adaptive Filtering. ........................................ 9
2-2. Outline of the LMS algorithm. ................................................................. 10
2-3. Non-linear Mapping $\varphi(.)$ to a larger dimension feature space. ............. 12

3-1. Interactive search in the descent gradient method . ................................. 17
3-2. Reconstruction for Mackey-Glass ............................................................. 20
3-3. Learning curve for Mackey-Glass ............................................................. 20
3-4. Update of $\sigma$ for Mackey-Glass ......................................................... 21
3-5. Update of $\eta$ for Mackey-Glass ............................................................. 21
3-6. Reconstruction for wind speed ............................................................... 22
3-7. Learning rate for wind speed .................................................................. 22
3-8. Update of $\sigma$ for wind speed ............................................................. 22
3-9. Update of $\eta$ for wind speed ............................................................... 22

4-1. K-means: input dataset . ........................................................................... 25
4-2. K-means: Seed point selection ............................................................... 25
4-3. K-means: Iteration 2 ............................................................................. 25
4-4. K-means: Final clustering ..................................................................... 25
4-5. Creation of a new dictionary. ................................................................. 28
4-6. Assignment of $x_t$ to a dictionary. ......................................................... 28
4-7. Union of dictionaries. ........................................................................... 28
4-8. Forecasting of the Lorenz time serie. ...................................................... 31
4-9. Forecasting of the Price of the Colombia Energy Market. ...................... 33

5-1. KLMS-NICE with udpate of $\sigma$: input dataset ................................... 36
5-2. KLMS-NICE with udpate of $\sigma$: new sample .................................... 36
5-3. KLMS-NICE with udpate of $\sigma$: update cluster ............................... 36
5-4. KLMS-NICE with udpate of $\sigma$: update sigma ................................. 36
5-5. Reconstruction test set ........................................................................... 37
5-6. Learning rate ......................................................................................... 37
5-7. algoritmh NICE with update of $\sigma$ ...................................................... 38

A-1. Learning curve. ..................................................................................... 40
List of Algorithms

1. Least Mean Square .................................................. 11
2. Kernel Least Mean Square ......................................... 14
3. Quantized Kernel Least Mean Square ....................... 15
4. Kernel Least Mean Square with update of $\eta$ and $\sigma$ ................................................. 19
5. K-means ................................................................. 25
6. Nearest Instance Centroid Estimation ...................... 26
7. Multiple dictionaries in online sequential learning. ............... 30
1. Preliminaries

1.1. Introduction

The time series forecasting plays a vital role in different sectors such as financial markets, telecommunications, data traffic on the Internet, meteorological, among others [1, 2]. That is why to achieve good predictions of time series, guide making of right decisions, that depending on the application could bring into the acquisition of benefits from economic or technical [3]. While making forecasts is essential, implementation is restricted and depends on the practical application, grouped into two groups: (1) offline, where the model is training before beginning the predictions [4]; (2) online, where the algorithms are sequentially trained under a real-time environment [5, 6].

Although most of the developed prediction operate work in an offline scheme [7, 8, 9], with the increase in flow and amount of data in real-time in multiple platforms like Google [10], or companies of sectors such as financial or energy, has led to the need to generate prediction models that work in online information flow environments [11]. Therefore, this thesis aims to address the task of generated prediction models via the sequential online circulation of information.

1.2. Review of approaches

Today, although there is a wide variety of statistical prediction methods such as the Autoregressive (AR), the Autoregressive Integrated Moving Average (ARMA), the Integrated Self Regressive Mobile Media (ARIMA), Bayesian approaches, among others [3]. These present difficulties in the forecast of non-stationary time series [12]. However, today, Machine Learning has become an alternative for forecasting this type of time series, due to the improvement of the results they have had in terms of precision [7]. Of all the diversity of methods existing in the Machine Learning, there are alternatives such as models with Neural Networks, Support Vector Machines (SVM) and Kernel Adaptive Filters (KAFs) [13, 14, 15]. Although methodologies with Neural Networks are an alternative for prediction tasks, they present difficulties in applications where an online prediction scheme is necessary, because they need a long training time and a high computational cost [16]. In the case of the SVMs, commonly used in classification problems, they are used to prediction tasks, but like the
Neural Networks present high computational requirements [17, 18].

The KAFs belong to the family of learning methods with kernels, which have become influential within the Machine Learning techniques due to the possibility of pattern recognition and the modest computational cost of [14]. Within the multiple applications in which the kernel methods have expanded, can be found the SVMs [19], Kernel Regularization Network (KRN) [20], Kernel Main Component Analysis (KPCA) and Kernel Fisher Discriminant Analysis (KFDA) [21]. While there are now methods that perform the prediction tasks, these methods are differentiated from others by their ability to learn under real-time environments, the ability to combine the universal approximation properties of neural network sand and the convex optimization of adaptive filters linear, through the Reproduction Hilbert Spaces (RKHS), which involves mapping the data to a space of infinite dimension [14]. In RKHS, the internal products between the data can be easily calculated using the "KernelTrick". Within all the variety of KAFs developed today we can mention the Kernel LeastMean Square algorithm (KLMS) [22], Kernel Recursive Least Square (KRLS) [23], and Extended Kernel Recursive Least Square [24], among others. The operating scheme of a KAF is via a dataset that enters sequentially over time, where some or all are stored in the dictionary. This dictionary has a weight vector that indicates the ponderation into the dynamics of time series, calculates between product prediction error and learning rate of the algorithm.

The KAFs have a calculate simple, then is the result of multiplicate a weights vector and a kernel between the current sample and a dictionary. [14]. Although the operation is simple, presents significant challenges and inconveniences to be solved as: (1) the reduction computational cost or dictionary size [25, 26, 27]; (2) the choice of an appropriate learning rate, which allows an optimal relation between the prediction error and convergence rate the algorithm [28, 29, 30]; y (3), the choosing the best kernel bandwidth [31], favorite due to its universal approximating capability, desirable smoothness and numeric stability [14, 32].

The choice of a good dictionary is essential in the task of prediction with KAFs because having stockpiled samples could bring a high computational cost or conditions of overfitting [25]. By contrast, a few samples could compromise obtaining good results [14]. These requirements make it necessary to have an equilibrium between the dictionary size and prediction error. Nevertheless, today, some methods limit the dictionary size, where only the essential input data are accepted. Some are the novelty criterion [33], the criterion of prediction variance [34], the criterion of Approximate Linear Dependence (ALD) [23], the surprise criterion [23] and the Vector Quantification (VQ) [25]. Although these methods limit the dictionary size, efforts must still be added to limit the dictionary size of the KAFs and improve the accuracy of predictions.

Although some propose methods reduce the dictionary size as mentioned above the struc-
tures, have been designed with the consideration of having an only dictionary. However, the algorithm of nearest instances proposed in [35], divides the increasing amount of data in the dictionaries with the creation of sub-dictionaries, ignoring those data that are outside of a domain region of some dictionaries in particular. The above is done by comparing the Euclidean distance between the data entered with the representative of the different existing dictionaries (centroids).

Another critical aspect is the tuning of parameters, usually done by heuristic methods. However, there are multiple, and more reliable ways to calculate the parameters of the KAFs for both $\eta$ and $\sigma$. In the case of the kernel bandwidth, its tuning can be via cross-validation [36, 37], complement methods [38], Silverman rule [39] or via interactive search in an online schema [31]. In the same way, the current literature presents tuning models for the learning rate via the relation between $\eta$ and the error described in [29, 40, 41].

1.3. Problems statement

Given the problems find in the state art in the kernel adaptive filtering, we three problems in this document:

- The tuning of kernel bandwidth and the learning rate in a prediction framework online with kernel adaptive filtering.
- The structuring of a selection framework of samples based on the creation and learning of multiples dictionaries.
- The adjust online of kernel bandwidth in KAFs with multiple dictionaries.

This problems are considered in the approach general objective to below.

1.4. Objectives

1.4.1. General objective

Develop an integration framework based on kernels for the analysis of high-dimensional data, which must have the capacity to work under sequential data with the adaptation of their free parameters and the selection of dictionaries.

1.4.2. Specifies objectives

- Design an adaptive strategy in KAFs that allows the tuning of the kernel bandwidth and the learning rate in online sequential prediction tasks.
1.5 Contribution

- Develop an online classification methodology that manages to generate multiple prediction models based on KAFs with the obtaining different dictionaries.

- Formulate a methodology that combines the tuning of kernel bandwidth and the creation of multiple dictionaries.

1.5. Contribution

The contributions of the present document are for each specific objective as:

- A methodology that tuning the learning rate and bandwidth of Gaussian kernel function, via the minimization of the correntropy as cost function.

- A framework of clustering of samples in KAFs, which allows the creation of multiples dictionaries allowing participation of samples in multiples dictionaries.

- A structure than combine a KAF with multiples dictionaries and other that adjust the kernel bandwidth.

1.6. Publications

With the work of this document, we have publications in:


1.7. Dataset

1.7.1. Synthetics

- **Mackey-Glass time series**: this dataset correspond the model of differential Equation (1-1), of which we consider the values of $b = 1$, $a = 0.2$ and $\tau = 30$ [14].

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ac(t-\tau)}{1 + x(t-\tau)^{10}} \tag{1-1}$$

- **Lorenz time serie**: the second dataset considered is the chaotic system of Lorenz described by the set of Equation (1-2). Where we took into account the values of $\sigma = 10$, $\beta = \frac{8}{3}$ and $\rho = 28$ [35].

$$\begin{cases} \frac{dx}{dt} = \sigma(y-x) \\ \frac{dy}{dt} = x(\rho - z) - y \\ \frac{dz}{dt} = -xy - \beta z \end{cases} \tag{1-2}$$

1.7.2. Wind speed

One of the time series real used corresponds to measurements in the wind speed in the cities of Puerto Bolívar in Colombia. Which as of great interest because recently the currents Atlas of wind energy of Colombia, published by the Instituto de Hidrología, Meteorología y Estudios Ambientales (IDEAM) and the Unidad de Planeación Minero Energética (UPME), classified as areas with favorable conditions for the use of Energy Sources Renewable [42]. These measurements were taken with a sampling frequency of one data per hour, corresponding to the average wind speed every hour in meters per second $^1$. The Table 1-1 presents the characteristics of the databases:

<table>
<thead>
<tr>
<th>Weather station</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Samples</td>
<td>Samples</td>
</tr>
<tr>
<td>Puerto Bolivar</td>
<td>900</td>
<td>100</td>
</tr>
</tbody>
</table>

Tabla 1-1.: Wind speed time serie.

1.7.3. Price kWh into Colombia energy market

The last dataset used for the analysis of proposed algorithms corresponds to the average price of kWh into Colombian energy market $^2$. This database, like the previous set, has a sampling frequency of one data per hour. The Table 1-2 presents in detail the selected data set

1 The data set is publicly available at http://www.ideam.gov.co/solicitud-de-informacion
### Dataset

<table>
<thead>
<tr>
<th>Samples</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>25000</td>
<td>0am 1/1/14 - 3pm 8/11/16</td>
<td>9000 2am 8/11/16 - 3am 16/11/17</td>
</tr>
</tbody>
</table>

**Tabla 1-2.** Price kWh into Colombia energy market
2. Mathematical background

2.1. Online prediction of time series from kernel methods

Today amount of information obtained daily increases at high rates every year [43], which causes significant challenges for the management and interpretation of all that information in real-time [44]. One application for the constant flow of information could be the forecast the behavior of the data in different applications of sectors as the commercial, energy industry, data-traffic on the Internet, among others. Nowadays, multiple tools have been developed focused on real-time prediction, one of them is the KAFs, models derived from the RKHS [44, 45], which use a linear structure and the internal product in the space of infinite dimension to implement linear adaptive filtering via the use of "kernel trick" [19].

In this chapter, we summarize the most common kernel-based learning methods for online sequential prediction tasks, the central theme of this document.

2.2. Kernel Adaptative Filterings

Adaptive filtering is a machine learning approach based on the automatic adjustment of free parameters and model topology [46]. This supervised adaptation is carried via online sequential learning of errors made with the passage of the iterations, from the diagnosis of the input-output relationship of the filter. Within the adaptive filtering exists, the one carried out via learning with kernels, which combines the benefits of linear adaptive filtering and kernel methods.

The adaptation of filter is realize via out by means of its error, which allows the adjustment of its parameters $w_{i-1}$, by means of a change indicated by $\nabla w_i$ with at updates each of its parameters $w_i$ (Equation (2-1)).

$$w_i = w_{i-1} + \nabla w_i$$  \hspace{1cm} (2-1)

The Figure 2-1 describes the operation of an adaptive filter, where for an input vector $(x_i)$ its output $(\hat{y}_i)$, is compared with the real value $(\tilde{y}_i)$, in order to adjust the internal parameters of model. This procedure occurs sequentially with the entry of more data.
2.2 Kernel Adaptative Filterings

2.2.1. Least Mean Square (LMS)

Least Mean Square algorithm (LMS) is a filter that has an optimal functioning in stationary time series, and although it is not part of the study of this document, it is the preliminary for the understanding of KAFs. This algorithm is based on the minimization of cost function error; for this, considered a given input number $x_1, x_2, ..., x_n$, each with a weight $w_1, w_2, ..., w_n$. The resulting output $y$ is the result of the linear combination of inputs and weights, as indicated in Equation (2-3). The objective of LMS algorithm is to find for a training dataset, the optimal adjustment of weights in order to obtain an adequate relation between the input-output.

\[ y = \sum_{k=1}^{n} w_k x_k \quad (2-2) \]

Since the algorithm is based on the cost function minimization, this function will be given by the mean square error, between the real value $y_i$ and the filter response $\hat{y}_i$, the function is given by:

\[ J_i = \frac{1}{2} e_i^2 = \frac{1}{2} (y_i - \hat{y}_i)^2 \quad (2-3) \]
When we start the algorithm, the first weight is initialized in zero and up-to-date with the number samples entered, where are compared the filter response with the prediction error (see Equation (2-4)):

\[
\begin{cases}
  w_0 = 0 \\
  e_i = y_i - w_{i-1}^T x_i
\end{cases}
\]  

(2-4)

The generalized form of the cost function for the i-th entry will be given by:

\[
J_w = \sum_{k=1}^{n} [y_i - w_i; \top x_i]^2
\]

(2-5)

If we implement the descending gradient method to calculate the minimum of the function with the instantaneous gradient of the task in time \(i\), we will obtain:

\[
\begin{cases}
  \nabla J_i = - \sum_{k=1}^{n} x_i (y_i - w_{i-1}^T x_i) \\
  \hat{\nabla} J_i = -x_i (y_i - w_{i-1}^T x_i)
\end{cases}
\]

(2-6)

The estimator \(w_i\) will be being updated from the previous estimator \(w_{i-1}\) and an adjustment conditioned by a step size \(\eta\):

\[
w_i = w_{i-1} + \eta x_i (y_i - w_{i-1}^T x_i)
\]

(2-7)

The algorithm 1 outlines the simplicity of LMS, where in the first instance the vector of weights starts in \(w_0\) and a learning rate \(\eta\).
2.2 Kernel Adaptive Filterings

Algorithm 1: Least Mean Square

- **Input**: \( \{x_i \in X, y_i\}, i = 1, 2, \ldots \)
- **Initialization**: 
  1. \( w_0 = 0 \), initial weight 
  2. \( \eta > 0 \), learning rate 
- **while** \( \{x_i, y_i\} \) **do**
  1. Output of filter 
  2. \( y = \sum_{k=1}^{p} w_k x_k \)
  3. Error of filter 
  4. \( e_i = y_i - w_{i-1}^T x_i \)
  5. Update of the weights 
  6. \( w_i = w_{i-1} + \eta e_i x_i \)
- **end**

2.2.2 Non-linear regression in RKHS

Although the LMS algorithm as filter adaptive has optimal results in linear time series [14], it present difficulties in real-world applications where the data has non-stationary behavior [47]. One way to extend its implementation is to map the input data \( x_t \), via a non-linear function \( \varphi(\cdot) \). This idea is based on the idea that the linear algorithm can be formulated in the space of high-dimensional characteristics RKHS in terms of its internal products. However, this methodology could have difficulties when dimensional space is infinite. Nevertheless, this is achieved with the exploitation of the so-called kernel trick. To understand this idea, the Figure 2-3 schematizes a dataset located in the space of dimension 2, which when mapped to a space of higher dimension (dimension 3 in this case) can identify and interpret better the characteristics of the dataset. This fact allows classifying better all the dataset. This similarity measure is done in a higher dimension space, where considered an infinite dimension space. This operation is realized with a kernel function [48].

A kernel is a positive, continuous and symmetric defined function \( \kappa: \mathcal{U}x\mathcal{U} \rightarrow \mathbb{R}^L \), where \( \mathcal{U} \) is the input domain and subset of \( \mathbb{R}^L \). The kernel trick indicated in Equation (2-8), defines that for a pair of vectors \( c_t \) and \( x_t \), the internal product between them in a space of infinite dimension, can be calculated via the operation in a kernel function. Today the kernels commonly used correspond to the Gaussian kernel [49], kernel polynomial [50], linear kernel, expository kernel, among others. The Equations (2-9) and (2-10) present the Gaussian and polynomial kernel forms respectively, where \( \sigma \) is the kernel bandwidth and \( P \) is the order for the polynomial kernel. While there are multiple kernels, the Gaussian kernel is the default option in KAFs due to its universal approach capability, desirable smoothness, and numerical stability [31].
Figure 2-3.: Non-linear Mapping $\varphi(.)$ to a larger dimension feature space.

$$\varphi(c_t)^{\top} \varphi(x_t) = \kappa(c_t, x_t)$$  \hspace{1cm} (2-8)

$$\kappa(c_t, x_t) = \exp \left( -\frac{||c_t - x_t||^2}{2\sigma^2} \right)$$  \hspace{1cm} (2-9)

$$\kappa(c_t, x_t) = (c_t^\top x_t + 1)^P$$  \hspace{1cm} (2-10)

2.2.3. Kernel Leasts Mean Square

The KLMS and LMS algorithms have the same development in their mathematical formulation. However, KLMS considers the \textit{kernel trick}, due to the considerations mentioned in the previous section. The initialization of KLMS is seen in the set of Equation (2-11).

$$\begin{cases}
  w_0 = 0 \\
  e_i = d_i - w_{i-1} \varphi(i) \\
  w_i = w_{i-1} + \eta e_i \varphi(i)
\end{cases}$$  \hspace{1cm} (2-11)

In Equation (2-11) the weights will be updated as they are entering data, which is expressed as a linear combination of the sample vectors history named as \textbf{dictionary} $C_t$, and the current data:

$$w_i = w_{i-2} + \eta e_{i-1} \varphi(i-1) + \eta e_i \varphi(i)$$
$w_i = w_0 + \eta \sum_{i=1}^{N} e_i \varphi(i)$

$w_i = \eta \sum_{i=1}^{N} e_i \varphi(i)$

When a new data $x_t$ arrives, the filter response will be expressed as the internal product between the weights and the latest data:

$w_i \varphi(x_t) = \left[ \eta \sum_{j=1}^{i} e_j \varphi(C_j)^\top \right] \varphi(x_t)$

(2-12)

Replacing 2-8 in 2-12 and simplifying $w_j = \eta e_j$, we get:

$w_i \varphi(x_t) = \sum_{j=1}^{i} w_j \kappa(C_j, x_t)$

(2-13)

Where in general we will have the filter output for each iteration will be given by the Equation (2-14).

$\hat{y}_i = \sum_{j=1}^{i} w_j \kappa(C_l, x_t)$

(2-14)

The filter summary is presented in detail in the algorithm 2, where:

- We start the algorithm with the parameters initialization of $N$, $\eta$, $\sigma$, $C$ and $w_1$ (line 2 to 6).
- For sample $x_t$ is computed the forecast into the predition horizon $N$ (line 9).
- It is estimated the current sample (line 11) and forecast error (line 13).
- Add a new sample to dictionary and a weight (line 15 to 16).
Algorithm 2: Kernel Least Mean Square

Input: \( \{x_t \in X, y_t\}, t = 1, 2, ... \)

1 Initialization:
2 \( \mathcal{N} \), prediction horizon
3 \( \eta > 0 \), learning rate
4 \( \sigma > 0 \) kernel bandwidth
5 \( C = \{x_1\} \), initial dictionary
6 \( w_1 = [\eta y_1] \), initial weight

7 while \( \{x_t, y_t\} \) do
8     Prediction of the next exit
9     \( y_{t+N} = \sum w_{t-1} \kappa (C_{t-1}, x_t) \)
10    Entry-exit training
11    \( \hat{y}_t = \sum w_{t-1} \kappa (C_{t-1}, x_{t-N}) \)
12    Prediction error
13    \( e_t = y_t - \hat{y}_t \)
14    Network update
15    \( C_y = \{C_{t-1}, x_t\} \)
16    \( w_t = [w_{t-1}, \eta e_t] \)
17 end

2.2.4. Quantized Kernel Least Mean Square

While the KLMS is an algorithm that represents the dynamics a time series, this is based on the establishment of a weights vector \( w_t \) and a dictionary \( C_t \) and due to then is sequential learning method, its size increases with each new data. This increase in the dictionary size is one of the biggest obstacles. However, the QKLMS algorithm implements the online vector quantization method in order to limit the dictionary size [25].

The quantification is developed by quantifying the weights vector \( w_t \) of the Equation (2-11), for this, when a new input data \( x_t \) enters the algorithm, calculates the euclidean distance between the incoming sample and each element of the dictionary, and quantifies \( x_t \) to the nearest vector according to the preselected quantization threshold. Otherwise, a sample will be assigned to the dictionary with \( C_t = \{C_t, x_t\} \). This procedure is schematized in greater detail in algorithm 3, Where additionally the steps of the algorithm 2 we have:

- We will calculate and select the sample nearest in the dictionary (line 16 to 17)
- If the distance is minor that the quantification threshold we update the dictionary and the weights (line 19 to 20).
If the distance is higher that the quantification threshold we add the sample into dictionary and a new weight (line 22 to 23).

\textbf{Algorithm 3:} Quantized Kernel Least Mean Square

\begin{algorithm*}
\textbf{Input} : \{x_t \in X, y_t\}, t = 1, 2, ...

\textbf{Initialization:}
\begin{itemize}
\item \(\mathcal{N}\), prediction horizon
\item \(\eta > 0\), learning rate
\item \(\sigma > 0\), kernel bandwidth
\item \(\varepsilon\), quantification threshold
\item \(C_1 = \{x_1\}\), initial dictionary
\item \(w_1 = [\eta y_1]\), initial weight
\end{itemize}

\textbf{while} \{x_t, y_t\} \textbf{do}
\begin{itemize}
\item Prediction of the next exit
\item \(y_{t+N} = \sum w_{t-1} \kappa(C_{t-1}, x_t)\)
\item Output-input training
\item \(\hat{y}_t = \sum w_{t-1} \kappa(C_{t-1}, x_t)\)
\item forecast error
\item \(e_t = y_t - \hat{y}_t\)
\item \textbf{Network update}
\begin{itemize}
\item \(\text{dist}(x_t, C_{t-1}) = \arg\min_{1 \leq j \leq \text{size}(C_{t-1})} \|x_t - C_{t-1}\|\)
\item \(j^* = \arg\min_{1 \leq j \leq \text{size}(C_j)} \|x_t - C_{t-1}\|\)
\item if \(\text{dist} \leq \phi\) then
\begin{itemize}
\item \(C_t = C_{t-1}\)
\item \(w^*_i = w^*_i + \eta e_t\)
\end{itemize}
\item else
\begin{itemize}
\item \(C_t = \{C_t, x_t\}\)
\item \(w_t = [w_t, \eta e_t]\)
\end{itemize}
\end{itemize}
\end{itemize}
\end{algorithm*}
3. Online sequential update of parameters

As indicated by Equation (2-9), the Gaussian kernel must be tuned to the selection of the optimal bandwidth or smoothing parameter ($\sigma$). This parameter is crucial in the task of sequential learning in KAF due to the implications it has on the precision and speed of convergence. The negative repercussions go with a too large $\sigma$, where the internal products in RKHS will have a value close to 1, and the data will have similar characteristics. On the other hand, if $\sigma$ is too small, the internal products in RKHS will have a value close to zero where is difficulty inference in the similarities between the dataset.

Today, there are mostly $\sigma$ tuning techniques such as cross-validation [51, 52], penalty functions [53], add-on methods [53, 38], Silverman rule [39], among others. However, most of these are not adequate in determining an optimal $\sigma$ in adaptive online filtering. The reason for the above is because, in this type of application where the data flow is large and the amount of data is not specified, it prevents the algorithm from adjusting to abrupt changes in the system. However, online adaptation methodologies have been proposed [31], showing the convergence to an appropriate value and consequently improving the accuracy of the KAF. However, the cost function used in this method does not consider the non-linear properties of the data. Another key parameter in learning is the learning rate ($\sigma$) because it determines the convergence time and the generalization of the model [14]. A too-large $\eta$ increases the convergence speed of the algorithm but increases the mismatch, in contrast, to a small value that decreases the time of convergence but increasing the generalization of the models.

In the present chapter, a mechanism for updating bandwidth kernel and the learning rate via the minimization correntropy as cost function. The distribution of the chapter is organized as follows: In Sections 3.1.1 and 3.1.2, we briefly and conceptually review the gradient descending method and the definition of correntropy. In Section 3.2, the process of the mathematical formulation of the proposed method is schematized. Finally, in Sections 3.3 and 3.4, we present the results and conclusions of the simulations carried out.
3.1. Background mathematical

3.1.1. Descending gradient method

The search for an optimal value that minimizes or maximizes a cost function is a priority for tuning of free parameters. However, in many cases finding such a value is not an easy task to do. This task is illustrated in Figure 3-1, where the minimum value of an objective function is sought iteratively and sequentially.

Where for a defined and differentiable cost function $f$, we have the hypothesis that it is maximum or minimum value decreases or increases as the case may be, from $f_a$, $\pm \nabla f_a$.

The previous is described in Equation (3-1) as:

$$z_{i+1} = z_n \pm \beta \nabla_z (z_n)$$  \hspace{1cm} (3-1)

Where $z_{n+1}$ is the minimum update, $\pm$ is the search address, $z_n$ is the minimum value present, $\beta$ is the rate of learning the gradient descending method and $\nabla_z (z_n)$ the direction of the change in the maximum or minimum value to find.
3.1.2. Correntropy

The correntropy defined in the Equation (3-2), is a measure of local similarity between two random variables \( X \) and \( Y \), for a window of a space determined by a bandwidth \( \lambda \).

\[
V_{\lambda}(X, Y) = E[\kappa_{\lambda}(X - Y)] = \int \kappa(x, y) dF_{xy}(x, y) \tag{3-2}
\]

where \( E \) denotes the expectation operator, \( \kappa(\cdot, \cdot) \) is a shift-invariant Mercer kernel, and \( dF_{xy}(x, y) \) denotes the joint distribution function of \( X \) and \( Y \). Without mentioned otherwise, the kernel function of correntropy is the Gaussian kernel:

\[
\kappa(x, y) = G_{\sigma} = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{3-3}
\]

Where \( \sigma > 0 \) is the kernel bandwidth, and \( \lambda = \frac{1}{2\sigma^2} \) is the kernel parameter.

3.2. Formulation

As mentioned before, the improvement of outputs of a KAF is related to an optimal determination of parameters \( \sigma \) and \( \eta \). It is for this reason that this document proposes the update of its parameters under an online sequential scheme, based on the minimization of cumulative corrections, defined for the Equation (3-4). Where \( e_t = y_t - \hat{y}_t \) is the prediction error and \( \lambda \) the bandwidth correntropy.

\[
V = \exp\left(-\frac{e_t^2}{\lambda^2}\right) \tag{3-4}
\]

The task is to update the step size \( (\eta) \) and kernel bandwidth \( (\sigma) \), knowing that the output of the KAF is given by \( y_t = \sum \eta^* e_t \kappa_{\sigma^*}(C_t, x_t) \), where \( \sigma^* \) and \( \eta^* \) the optimal values in the KAF output. The search for these values is done with a gradient descending method, as outlined in Equations (3-5) and (3-6).

\[
\eta_t = \eta_{t-1} + \beta \frac{\partial V_t}{\partial \eta_{t-1}} \tag{3-5}
\]

\[
\sigma_t = \sigma_{t-1} + \beta \frac{\partial V_t}{\partial \sigma_{t-1}} \tag{3-6}
\]

The reason for the search is that with the passage of the iterations in training, for when \( \lim_{t \rightarrow \infty} \eta_t = \eta^* \) and \( \lim_{t \rightarrow \infty} \sigma_t = \sigma^* \), get the best for the KAF output. The solution of Equation (3-5) and 3-6 are shown in the Equation (3-7) and 3-8 respectively. The Appendix B
3.2 Formulation and Appendix C outline in detail the mathematical procedure for the calculation of the solution.

\[ \sigma_t = \sigma_{t-1} + \frac{\beta}{\lambda^2 \sigma_{t-1}^3} \exp \left( -\frac{\epsilon_t^2}{2\lambda^2} \right) \eta_{t-1} e_t e_{t-1} ||x_t - x_{t-1}||^2 \kappa_{\sigma_{t-1}} (x_t, x_{t-1}) \]  

(3-7)

\[ \eta_t = \eta_{t-1} + \beta \exp \left( -\frac{\epsilon_t^2}{2\lambda^2} \right) e_t e_{t-1} \kappa_{\sigma_{t-1}} (x_t, x_{t-1}) \]  

(3-8)

Where \( \sigma_t \) and \( \eta_t \) are the values in each iteration, \( \beta \) is the learning rate of the search, \( || \cdot || \) represents the norm \( l_2 \). The algorithm 4 schematizes the update proposal methodology of \( \eta \) and \( \sigma \) merged with the KLMS algorithm described in Section 2.2.3. The update is done in lines 16 and 18.

---

**Algorithm 4:** Kernel Least Mean Square with update of \( \eta \) and \( \sigma \)

**Input:** \( \{(x_t \in X, x_n)\), \( t = 1, 2, ... \) 

1. **Initialization:**
   2. \( N \), Prediction horizon
   3. \( \lambda \), correntropy bandwidth
   4. \( \eta_1 = 0 \), initial learning rate
   5. \( \sigma_1 \) > bandwidth of the Gaussian kernel function of the KAF
   6. \( C_1 = \{ x_1 \} \) initial dictionary
   7. \( w_1 = [y_1 y_1] \), initial weight

8. **while** \( \{x_t, y_t\} \) **do**

   9. Prediction next exit
   10. \( y_{t+N} = \sum w_{t-1} \kappa (C_{t-1}, x_t) \)
   11. Input-output training
   12. \( \hat{y}_t = \sum w_{t-1} \kappa (C_{t-1}, x_{t-N}) \)
   13. Prediction error
   14. \( e_t = y_t - \hat{y}_t \)

15. Optimization of \( \sigma \)

   16. \( \sigma_t = \sigma_{t-1} + \frac{\beta}{\lambda^2 \sigma_{t-1}^3} \exp \left( -\frac{\epsilon_t^2}{2\lambda^2} \right) \eta_{t-1} e_t e_{t-1} ||x_t - x_{t-1}||^2 \kappa_{\sigma_{t-1}} (x_t, x_{t-1}) \)

17. Optimization of \( \eta \)

   18. \( \eta_t = \eta_{t-1} + \beta \exp \left( -\frac{\epsilon_t^2}{2\lambda^2} \right) e_t e_{t-1} \kappa_{\sigma_{t-1}} (x_t, x_{t-1}) \)

19. Network update

   20. \( C_t = \{ C_t, x_{t-N} \} \)

   21. \( w_t = [w_t, \eta e_t] \)

22. **end**
3.3. Results

The framework for updating the $\sigma$ and $\eta$ parameters proposed in prediction tasks was validated using a qualitative interpretation of the reconstruction of the time series, the learning curve described in Appendix A.1 and the update values of the $\sigma$ and $\eta$ parameters in two chosen datasets. The first one corresponds to the Mackey-Glass synthetic data set (see Section 1.7.1). The second set was real-world data, corresponding to wind speed data from the city of Puerto Bolivar (see Section 1.7.2). The results obtained were compared with the algorithm KLMS, QKLMS, and the KLMS with a variable adaptation of the bandwidth (KLMS-VKS).

3.3.1. Synthetic data

For this first time series, we had a training set of 500 samples and 100 of validation. The tuning parameters of each KAF were of a delay time of 10 samples and a prediction horizon of one sample made with parameters of $\eta = 0.2$ and $\sigma = \frac{1}{\sqrt{2}}$ only for KLMS and QKLMS. On the other hand, the initialization of the parameters of $\sigma$ and $\eta$ for the adaptive algorithms KLMS-VKS and the proposed one was $\frac{1}{\sqrt{2}}$ and 0.01 respectively. In the case of the learning rate, we take a value of 0.01 with a bandwidth of 0.7 for the correlation function.

The Figure 3-2 presents the reconstruction of the Mackey-Glass time series, where qualitatively an optimal reconstruction of the curve is observed for each of the algorithms used; However, it is not possible to infer in which of all the methods the best results are presented. However, the learning curve of the Figure 3-3 presents a faster convergence of the algorithm compared to the other KAFs, with a further improvement around the mean square error.

![Figure 3-2: Reconstruction for Mackey-Glass](image)

![Figure 3-3: Learning curve for Mackey-Glass](image)
3.3 Results

On the other hand, this improvement in the convergence and precision results of the algorithm can be associated with the update of $\eta$ and $\sigma$ expressed in the Figures 3-4 and 3-5, where these values are updated throughout the iterations. In the case of $\sigma$ the proposed algorithm converges more quickly to a value close to 1 with respect to the KLMS-VKS algorithm. This rapid convergence occurs in a similar way with $\eta$, where we approach 0.22.

![Figure 3-4: Update of $\sigma$ for Mackey-Glass](image)

![Figure 3-5: Update of $\eta$ for Mackey-Glass](image)

The previous results were summarized in Table 3-1, where we can see the parameters of kernel bandwidth and learning, as the MSE obtained for the test set.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>KLMS</th>
<th>QKLMS</th>
<th>KLMS-VKS</th>
<th>Proposal</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>0.114</td>
<td>0.121</td>
<td>0.110</td>
<td>0.107</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>$\frac{1}{\sqrt{2}}$</td>
<td>$\frac{1}{\sqrt{2}}$</td>
<td>0.73470</td>
<td>0.709</td>
</tr>
<tr>
<td>$\eta$</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.123</td>
</tr>
</tbody>
</table>

Tabla 3-1.: Configuration and error final for Mackey-Glass

3.3.2. Wind speed

In this proof, we have utilized the same configuration as the preceding section. However, the time series size is more extensive, with 900 samples for the training set and 100 for the test set (see Table 1-1). The time series reconstruction can be observed in Figure 3-6; for this, we use the KAF obtained at the end training. This chart shows that reconstruction is not the most appropriate in terms of magnitude. Nevertheless, the algorithms can reconstruct the trajectory in terms of growth and decrease of the time series.
The graphs in the Figure 3-7 figure show the evolution of MSE throughout the iterations. Where it is possible to see, the decrease in the error for all methods, presenting two abrupt changes approximately in iterations 450 and 600. Although the algorithms converge to similar speed with a slight delay in the proposed method, at the end training, we obtain a decrease of error, which is possible in greater detail and clarity in Table 3-2.

In Figures 3-8 and 3-9 are possible to see the optimization of $\sigma$ and $\eta$ throughout time, where we can note a similar growth behavior with a difference of approximately 0.2 in the magnitude between KLMS-VKS and the proposal. In the case of the learning rate, the update is speedy for the first 100 interactions keeping an updating incremental in all training (see Figure 3-9). This is due to KAF are encoding the dynamic of the time serie. As a summary, Table 3-2 shows the values finals for kernel bandwidth and the learning rate.
3.4 Conclusions

In this chapter, an update scheme of the kernel bandwidth, and the learning rate for adaptive kernel filters was presented. This model updates the mentioned parameters throughout the iterations through the optimization of the cost function, using a stochastic gradient algorithm that maximizes the correntropy cost function. Therefore, the prediction error decreases throughout the training, which improves the convergence time while maintaining the robustness of the Adaptive Filter. The above is due to the inherent capacity of the function of correntropy at outliers.

The validation of the scheme is validated in two sets of data, synthetic and the real world, showing an improved convergence rate accompanying a decrease in the mean square error, which allows solutions for real-world applications.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>KLMS</th>
<th>QKLMS</th>
<th>KLMS-VKS</th>
<th>Proposal</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>0.114</td>
<td>0.121</td>
<td>0.110</td>
<td>0.107</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>$\frac{1}{\sqrt{2}}$</td>
<td>$\frac{1}{\sqrt{2}}$</td>
<td>0.734</td>
<td>0.709</td>
</tr>
<tr>
<td>$\eta$</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.123</td>
</tr>
</tbody>
</table>

Tabla 3-2.: Configuration and error final for Wind speed
4. Online grouping in Kernel Adaptative Filterings

Grouping of data via cluster is a methodology of subsetting a particular set of particular data, where samples from the same cluster have similar characteristics to each other, but differences from samples from other clusters [54]. This partitioning of the samples is widely used in tasks of pattern recognition, image analysis, information retrieval, data compression, among others [55]. Among the clustering algorithms can be mentioned k-means [56] and K-Nearest Neighbor [57]. Both techniques are elementary to understand and implement, which makes them very popular and applicable to many problems [58]. Although, algorithms majority work in supervised learning schemes, offline and with labeled data [59]; in practical applications, it is necessary a model online where the information not labeled and did not consider a number groups. A scheme clustering online and implemented in KAF have proposed in [35], where were built clusterings (dictionaries for KAFs), and that restrict the samples participation in the only dictionary. Which implies limiting samples to evaluate and update the KAF. However, this algorithm considers the participation of a sample in an only KAF.

In this chapter, we present an online grouping mechanism applied in KAF, which allows for improvement in terms of precision for time series with abrupt changes. The chapter distribution is organized as follows: A review of the grouping state models is given in Section 4.1. In Section 4.2, the proposed grouping model applied in KAF is detailed. Finally, Sections 4.3 and 4.4 presents the results and conclusions obtained for the databases used.

4.1. Background mathematical

4.1.1. K-means algorithm

Suppose we have a data set \( \{x_1, x_2, ..., x_n\} \) n-dimensional and clustered into a set of clusters \( C = \{c_k : k = 1, 2, ..., n\} \). Our goal is to partition the data set into some number \( k \) of clusters, where we shall suppose for the moment that the value of \( k \) is given. The goal of K-means is to minimize the sum of the squared error over all \( k \) clusters as indicate Equation (4-1):

\[
J(C) = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - u_k||^2
\]  

(4-1)
The Figures 4-1 to 4-4 shows an illustration of the K-means algorithm for dataset of 2-dimensional with three clusters. The procedure is the next: (Figure 4-1) input dataset; (Figure 4-2) four points selected as cluster centers and initial assignment of the data points to clusters; (Figure 4-3) iterations that updating cluster labels and their centers; (Figure 4-4) final clustering obtained by K-means algorithm at convergence.

The algorithm 5 schematizes the procedure developed by K-means.

<table>
<thead>
<tr>
<th>Algorithm 5: K-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Initialize $m_k$</td>
</tr>
<tr>
<td>2 while until converge do</td>
</tr>
<tr>
<td>3 Assing each data point to its closed cluster center:</td>
</tr>
<tr>
<td>4 $z_k = \text{argmin}_k</td>
</tr>
<tr>
<td>5 Update each cluster center by computing</td>
</tr>
<tr>
<td>6 $u_k = \frac{1}{N_k} \sum_{i=1}^k x_i$</td>
</tr>
<tr>
<td>7 end</td>
</tr>
</tbody>
</table>

Figure 4-1.: K-means: input dataset
Figure 4-2.: K-means: Seed point selection
Figure 4-3.: K-means: Iteration 2
Figure 4-4.: K-means: Final clustering
4.1.2. Nearest Instance Centroid Estimation

Nearest Instance Centroid Estimation (NICE) is an in-line sequential grouping algorithm, which assigns elements and creates groups based on the minimum distance with the nearest centroid and threshold previously outlined $\delta$ \cite{35}. Its operation for a current sample is calculated with the distance to the minimum centroid, resulting in two operations: (1) if the minimum distance with the centroid is less than a threshold, the sample will be assigned to said nearest cluster; (2) If the distance is higher, then a new cluster will be created.

**Algorithm 6: Nearest Instance Centroid Estimation**

**Input**: \( \{x_t \in X, \}, t = 1, 2, \ldots \)

1. **Initialization**:
   2. $\delta$: centroid distance threshold
   3. $g_1 = x_1$: centroid for cluster 1
   4. $s_1 = 1$: effective cluster of cluster 1 for centroid update
   5. $G_1 = \{x_1\}$: center of cluster 1
   6. $G = \{x_1\}$: set of cluster

7. While \( \{x_t\} \) do
   8. Compute minimum centroid distance
   9. $d_{\text{min}}^c = \text{argmin}_{1 \leq j \leq |G|} ||x_i - g_j||^2$
   10. Select nearest-neighbor cluster
   11. $j^* = \text{argmin}_{1 \leq j \leq |G|} ||x_i - g_j||^2$

   **Nearest-Instance-Centroid-Estimation**

12. If $d_{\text{min}}^{c,j^*} \leq \delta$ then
   13. Update cluster $j^*$
   14. $G_{j^*} = \{G_{j^*}, x_i\}$
   15. Update cluster $j^*$ centroid, the effective size
   16. $g_{j^*} = \frac{s_{j^*} \cdot g_{j^*} + x_i}{s_{j^*} + 1}$
   17. $s_{j^*} = s_{j^*} + 1$

18. Else
   19. From new cluster
   20. $C_{|C|+1} = \{x_t\}$: new cluster
   21. $C_{|C|+1} = x_t$: centroid
   22. $s_{|C|+1} = 1$: effective size
   23. $C = \{C, C_{|C|+1}\}$

24. End

25. End
4.2. Formulation

We propose to enhance online prediction tasks by partitioning the set of centers (also known as dictionaries) into distinct regions that encode different patterns of the input space. Thus, dictionaries are directly learned from data using a kernel bandwidth metric and their centroids (a single representative of each dictionary) are updated through iterations. More formally, when a new input vector $x_t$ arrives, the euclidean distance between $x_t$ and each centroid $\gamma_i$ is computed. This reduces the number of hyperparameters, which makes the proposed method easier to implement in practice. Thus, metrics based on coherence [60], approximate linear dependence [23], and surprise criterion [27] were not considered in this work. Then, if the euclidean distance between $x_t$ and the closest dictionary $i^*$ remains below a given threshold $\delta R$, the input vector $x_t$ will be stored as a new center (see Figure 4-5). Otherwise, the input vector $x_t$ will be used to form a new dictionary. This can be seen in Figure 4-6, where the input vector $x_t$ is stored in the dictionary $C_3$ and its centroid $\gamma_3$ is updated as follows:

$$
\gamma_j = \frac{S_{\gamma_j} - 1}{S_{\gamma_j}} \gamma_j + \frac{1}{S_{\gamma_j}} x_t
$$

(4-2)

where $\gamma_i$ is the $i$-th centroid and $S_{\gamma_i}$ denotes the dictionary sizes. This equation is used because, in contrast with batch methods, the calculation is simple in scheme online [61]. The reason is then only used the new sample, the old centroid and the dictionary size.

This strategy learns the clusters directly from data, and updates their centroid locations through the iterations. Note, the above strategy has similarities to some previously proposed methods [35]. However, the following are the key differences between both proposals: i) our proposal may store an input vector in several dictionaries at the same time; ii) it presents an additional update scheme for updating weights such as the presented in [25], implemented in the case where input vectors belong to more than two dictionaries. This allows the algorithm to adjust to abrupt changes in the system, which is useful in nonstationary conditions.

In addition, with the aim of reducing computational complexity, the dictionaries are merged during the adaptation process. That is, as seen in Figure 4-7, the dictionaries $C_n$ and $C_m$ will be merged as long as one of the following two conditions is met:

$$
S_{C_n} \cap S_{C_m} > S_{C_n} - S_{C_n} \cap S_{C_m}
$$

(4-3)

$$
S_{C_n} \cap S_{C_m} > S_{C_m} - S_{C_n} \cap S_{C_m}
$$

(4-4)
Then, the centroids and dictionary size will be updated as follows:

\[
\gamma_{new} = \frac{S_{\gamma_1} + S_{\gamma_2}}{S_{\gamma_1}}
\]

(4-5)

\[
S_{new} = S_{\gamma_1} + S_{\gamma_2} - S_{\gamma_1} \cap S_{\gamma_2}
\]

(4-6)

**Figure 4-5.** Creation of a new dictionary. **Figure 4-6.** Assignment of \( x_t \) to a dictionary. **Figure 4-7.** Union of dictionaries.
4.2 Formulation

The Equations (4-5) and (4-6) proposed in [62], presents an update scheme of centroids in multiple clusters and oriented in sequential online applications. This mechanism updates the centroids in the function of dictionary size and current centroids, without considering distribution in the data. On the other hand, as mentioned earlier, as dictionaries merge, we update weights the samples shared. This condition can see in Figure 4-7, where the weights shared \( w_n^1 \) and \( w_n^2 \) of each group it is averaged \(^1\). This operation is done no matter if a sample belongs to more of two dictionaries.

The basic idea of the framework proposed is allowing that samples with similar characteristics into one or more dictionaries not this restricted to belonging to a only dictionary. However, this can increase the dictionary size. For this reason, than framework proposed implemente the Vector Quantization as is decribed in [25], which updates the weigths of a pair similar samples. This is done through the calculate euclidean distance between two samples. If distance is less than \( \zeta \) (quantification threshold), the weight is update with sum of product between learning rate and predition error.

For understand the framework poposed, we provide a step-by-step description of how we create the dictionaries sequentially online (see Algorithm 7):

- We start with initialization of \( \eta, \sigma, \delta, \zeta \), firsts dictionary and centroid of the first group (Line 1 to 4).
- The first sample \( x_1 \) is stored in the only dictionary existing, with the centroid update, and weight vector (Line 9 to 13).
- For sample two and onwards is calculate the euclidean distance between the sample and all centroids of the dictionaries (line 6), where:
  - If euclidean distance is less to \( \delta \) in a dictionaries we add \( x_t \) to these and update the dictionaries and weihts vector in accordance with Vector Quantization described in [25] (line 20).
  - If euclidean distance is greate to \( \delta \), add \( x_t \) as new a sample in the dictionary (line 22 to 25).
- For all pairs of dictionaries calculates according to Equation (4-3) or Equation (4-4) (line 29), if one is fulfilled, we combined the dictionaries according to ilustrated in Figure 4-7

\(^1\)For this case the operarions were: 
\[
\begin{align*}
    w_{update}^{C_{new}} &= \left( w_1^{C_{new}} + w_1^{C_1} \right) / 2 \\
    w_{update} &= \left( w_2^{C_2} + w_2^{C_2} \right) / 2
\end{align*}
\]
Algorithm 7: Multiple dictionaries in online sequential learning.

**input**: \( \tau \)-training data

1. \( \eta \)-learning rate; \( \sigma \)-kernel bandwidth;
2. \( \delta \)-centroid quantification threshold; \( \zeta \)-quantification threshold between samples;
3. \( C_1^1 = \{ x_1 \} \): initial dictionary;
4. \( \Omega_1^1 = \{ x_1 \} \): centroid of the 1\textsuperscript{st} group;
5. \( w_1^1 = \eta y_1 \): initial weight

**while** \( \{ x_t, y_t \} \) **do**

6. Selecting the group with the minimum distance: \( j^* = \text{argmin}_{1 \leq j \leq \text{size}(\Omega^j)} \| x_t - \Omega^j \| \);
7. Forecasting: \( y_{t+N} = \sum w_{t-1}^{j^*} \kappa(C_{t-1}^{j^*}, x_t) \);
8. Forecasting error: \( e_t = y_t - \hat{y}_t \);

**if** \( t = 1 \) **then**

9. Adding \( x_2 \) to the existing group \( C_1^1 = \{ C_1^1, x_2 \} \);
10. Number of elements in group 1: \( S_1^1 = 2 \);
11. \( \Omega_1^1 = \frac{S_1^1 - 1}{S_2^1} \ast \Omega_1^1 + \frac{1}{S_1^1} \ast x_2 \);
12. Assigning the sample weight: \( w_2^1 = \{ w_1^1, \eta \} \)

**else**

13. **for** all existing dictionaries **do**
14. \( d_z = \| x_t - \Omega^z \| \);
15. **if** \( d_z \leq \delta \) **then**
16. \( k^* = \text{argmin}_{1 \leq j \leq \text{size}(\Omega^z)} \| x_t - C_j^z \| \);
17. **if** \( \| x_t - C_{k^*}^z \| \leq \zeta \) **then**
18. \( w_{k^*} = w_{k^*} + \eta e_t \): Update weight;
19. **else**
20. Adding \( x_t \) in the z group: \( C_{t+k^*}^z = \{ C_{t+k^*}^z, x_t \} \);
21. Number of elements in the z group: \( S_{k^*}^z \);
22. Centroid update: \( \Omega^z = \frac{S_{k^*}^z - 1}{S_t^z} \ast \Omega^z + \frac{1}{S_t^z} \ast x_t \);
23. Update weight: \( w_{t+k^*}^z = \{ w_{t+k^*}^z, \eta e_t \} \);
24. **else**
25. New dictionary: \( C_1^q = \{ x_i \} \); \( \Omega^q = \{ x_i \} \); \( w_1^q = \eta y_t \)

**for** all pairs of dictionaries \( i \) and \( j \) **do**
26. **if** \( S_{C_i} \cap S_{C_j} > S_{C_i} - S_{C_j} \) or \( S_{C_i} \cap S_{C_j} > S_{C_j} - S_{C_i} \) **then**
27. Update dictionary;
28. \( \gamma_{\text{new}} = \frac{S_{C_i} + S_{C_j}}{S_{C_i} + S_{C_j}} \);
29. \( S_{\text{new}} = S_{C_i} + S_{C_j} - S_{C_i} \cap S_{C_j} \)

**output**: Salida
4.3. Results

4.3.1. Synthetic data

In the case of the Lorenz chaotic system, we use a set of 3000 samples training and 2000 validation, normalized with zero mean and variance 1 [35]. The tuning parameters of KAF was a delay time of 8 samples, a prediction horizon of samples, $\sigma = 1$ and $\eta = 0,1$ [35]. However, quantification thresholds $\zeta$ and $\delta$ were previously select a search mesh, were obtained the best result in MSE. These results indicated in Table 4-1. This search mesh was for values of $\zeta$ between 0.1 and 9.9, and of $\delta$ between 0.01 and 0.9. Furthermore, the parameters chosen for LSTM prediction model were 4 layers with nodes of 200, 100, 90, and 50 respectively, 500 epochs and a batch size of 100 [63].

The Figure 4-8 presents the reconstruction of the testing dataset for the proposed algorithm and other prediction online methods. In this graph can be seen as all KAFs and proposal, rebuilding almost the entire time series. However, approximately in samples between 1200 y 1250 where the time serie shows a change values negative to positive, the KAFs KLMS and QKLMS do not have an adequate response. In the case of the KAFs NICE and the proposed algorithm, the reconstruction is better in this segment. The reason for the previous is the selection of a better dictionary that represents this change in the time series. On the other hand, the reconstruction with LSTM model, although encoding the dynamics did not synchronize with the validation dataset.

![Figure 4-8: Forecasting of the Lorenz time serie.](image-url)
Although Figure 4-8 is indicative of inferring prediction results, Table 4-1 quantitatively indicates the performance of the algorithms in terms of the MSE and the number of samples contained in each model. The results show a slight improvement in the MSE for the algorithm proposed in its base version as quantified. This improvement of results may be due to dictionary size and the adjustment of the weights where fusion dictionaries.

<table>
<thead>
<tr>
<th>Model</th>
<th>$\zeta$</th>
<th>$\delta$</th>
<th>MSE</th>
<th>Dictionary size</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLMS</td>
<td>DNA</td>
<td>DNA</td>
<td>0.0310</td>
<td>3000</td>
</tr>
<tr>
<td>QKLMS</td>
<td>0.1 DNA</td>
<td>0.0316</td>
<td>953</td>
<td></td>
</tr>
<tr>
<td>KLMS-NICE</td>
<td>DNA</td>
<td>5.2</td>
<td>0.0344</td>
<td>1064, 1056 y 340</td>
</tr>
<tr>
<td>QKLMS-NICE</td>
<td>0.1</td>
<td>6</td>
<td>0.0343</td>
<td>484, 393 y 159</td>
</tr>
<tr>
<td>Proposal-BV</td>
<td>DNA</td>
<td>6.6</td>
<td>0.0309</td>
<td>3000</td>
</tr>
<tr>
<td>Proposal-QBV</td>
<td>0.03</td>
<td>6.6</td>
<td><strong>0.0301</strong></td>
<td>2375</td>
</tr>
<tr>
<td>LSTM</td>
<td>DNA</td>
<td>DNA</td>
<td>1.2365</td>
<td>3000</td>
</tr>
</tbody>
</table>

**Tabla 4-1.** Mean Square Error for Time serie of Lorenz.

### 4.3.2. Price of the Colombian energy market

The second time series corresponds to the average price of KWh in the Colombian energy market. The parameters chosen for training in each KAF were a delay time of 10 samples, a prediction horizon of a sample, $\sigma = \frac{1}{\sqrt{2}}$ and $\eta = 0.1$. In the same way, like the Lorenz chaotic system, the quantification thresholds were chosen via a search mesh between 0.1 and 0.9 for $\delta$ and $\zeta$. In the case of kernel bandwidth, the search mesh was between 0.1 and 1. On the other hand, the configuration of the LSTM was keeping as in [63].

The Figure 4-9 shows the forecasting of the average price of the KWh in the validation dataset for the KAFs, the proposed algorithm, and the LSTM network. In this graph, the LSTM model (green color) does not have an adequate prediction, which is mainly due to the combination of all training set dynamics in an only prediction model. This occurs to a lesser extent in the KAFs, due to better encoding of the dynamic of the time serie. This improvement can be illustrated in Table 4-2, where you can observe as the MSE in the LSTM network is higher in comparison to all methods. However, the error in KLMS and QKLMS methods are relatively high compared to the NICE and the proposed; the reason for this is the use of training samples that do not code the dynamic of the time serie. This does not occur in prediction schemes with multiple dictionaries, because the coding of the training set via multiples KAFs, allows only the model containing the dynamics of interest. This condition allows reducing the size model, as can be seen in Table 4-2.
4.4 Conclusions

In this framework, an online sequential classification framework is presented for prediction tasks via adaptive kernel filtering. The task is focused on the grouping of the most relevant samples in each dictionary, this is done based on the shortest distance between a sample with

### Figure 4-9.: Forecasting of the Price of the Colombia Energy Market.

<table>
<thead>
<tr>
<th>Model</th>
<th>ζ</th>
<th>δ</th>
<th>MSE</th>
<th>Dictionary size</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLMS</td>
<td>DNA</td>
<td>DNA</td>
<td>0.00038</td>
<td>25000</td>
</tr>
<tr>
<td>QKLMS</td>
<td>DNA</td>
<td>0.01</td>
<td>0.00038</td>
<td>11063</td>
</tr>
<tr>
<td>KLMS-NICE</td>
<td>DNA</td>
<td>0.9</td>
<td>0.00017</td>
<td>22155, 2845</td>
</tr>
<tr>
<td>QKLMS-NICE</td>
<td>0.02</td>
<td>0.9</td>
<td>0.00011</td>
<td>5406, 337</td>
</tr>
<tr>
<td>Proposal-BV</td>
<td>DNA</td>
<td>0.4</td>
<td>0.00018</td>
<td>21362, 4558, 2, 18-1, 181, 7</td>
</tr>
<tr>
<td>Proposal-QBV</td>
<td>0.01</td>
<td>0.4</td>
<td><strong>0.00009</strong></td>
<td>9195, 7, 14-1, 5, 137, 3-1, 2, 2731</td>
</tr>
<tr>
<td>LSTM</td>
<td>DNA</td>
<td>DNA</td>
<td>1.23702</td>
<td>25000</td>
</tr>
</tbody>
</table>

### Table 4-2.: Mean Square Error for Time Serie of price of the Colombian energy market.

4.4. Conclusions

In this framework, an online sequential classification framework is presented for prediction tasks via adaptive kernel filtering. The task is focused on the grouping of the most relevant samples in each dictionary, this is done based on the shortest distance between a sample with
the centroid of each. Although this methodology is implemented in other works, the proposal developed differs by the fact that a sample can belong to multiple dictionaries, the creation of the dictionaries is limited to a possible union of these, and there is a simple scheme for the calculation of the weights of the samples that are shared in more than one of them. These conditions allow a better relationship between the number of samples and the most representative between each dictionary, which in practice translates into a better representation of the dynamics of a time series, and the possibility of choosing the best prediction scheme before possible abrupt changes that occur in the time series. These qualities of the proposed framework can be noted with the decrease in prediction error and dictionary size for both synthetic and real-world data.

While the framework is an online sequential classification approach developed in adaptive kernel filtering, it is necessary to expand research to find a more appropriate methodology for calculating weights for elements that are shared in more than one dictionary.
5. KAF-NICE with update of \( \sigma \)

As mentioned in past chapters, the tuning of kernel bandwidth is essential for the improvement of forecast results in KAF. And although in Chapter 3 was approached this topic, in schemes with multiples dictionaries today day few methodologies integrate the creation of multiple filters and updating the kernel bandwidth in each filter [35, 64, 65, 66, 67, 68]. To solve this problem in this chapter, we propose to integrate into the same model of creation of multiple dictionaries proposed in [35] and the framework of updating the kernel bandwidth proposed in [31].

Unlike the previous chapters, Chapter 5 does not contain a mathematical review; The reason for this is because the methods we integrate is detailed in Chapter 2 and extensively in Sections 3.1.1 and 4.1.2. For the above, we only present the proposal, and the results obtained in a synthetic database. We will present the development of the proposal in Section 5.1, the results and the conclusions obtained in Sections 5.2 and 5.3.

5.1. Proposal

The KLMS-NICE algorithm and in its Vector Quantification version QKLMS-NICE builds a set of filters with the passage of iterations as indicated by algorithm 6. However, bandwidth tuning is subject to prior knowledge of the time series. It is for this reason, that to solve this inconvenience we update the kernel bandwidth as indicated in Equation (5-1), equation proposed in [31].

\[
\sigma_i = \sigma_{i-1} + \rho e_{i-1} e_i \| u_{i-1} - u_i \| \kappa_{\sigma_{i-1}} (u_{i-1}, u_i) 
\]  

Equation (5-1)

This simple but effective methodology is shown in Figures 5-1 to 5-4, where:

1. Initially, we have a cluster set, and all with an assigned kernel bandwidth (see Figure 5-1).

2. With a new sample (see Figure 5-2), we select the most appropriate set of samples or dictionary to make the prediction (see Figure 5-3). The dictionary selection is performed, as indicated by algorithm 6.
3. Although so far the previous items are the methodology of a KAF-NICE, what we propose is to update the bandwidth of the cluster where the new sample is stored (see Figure 5-4). For this, we update the bandwidth as indicated by Equation (5-1).

![Figure 5-1: KLMS-NICE with update of σ](image1)
![Figure 5-2: KLMS-NICE with update of σ](image2)
![Figure 5-3: KLMS-NICE with update of σ](image3)
![Figure 5-4: KLMS-NICE with update of σ](image4)

5.2. Results

The results of the methodology were tested for a synthetic database (Mackey-Glass), which is explained in detail in Section 1.7.1. For the simulations we had a set of 800 training samples and 200 validation, the chosen parameters were a delay time of 8 samples, a horizon of prediction of a sample, a learning rate of 0.1, a threshold of quantification of 0.1 and a
5.2 Results

Comparison distance for the NICE algorithms of 3.5 and a bandwidth calculated according to the Silverman rule. The tuning of these values was given for the best results for a window of 8 samples, and obtaining a final number of 4 clusters for the algorithm KLMS-NICE and QKLMS-NICE.

The Figure 5-5 shows the results around the reconstruction of the data set; in it, we find that all the methods successfully reconstruct the signal maintaining the changes and the magnitude of the time series, which indicates that our method has a satisfactory rank with respect to others. On the other hand, the Figure 5-6 presents the learning curve of the different algorithms evaluated, where we find that the convergence rate of the proposal resembles the best of the models in comparison.

In the case of bandwidth actualization, the differences are noticeable; this can be seen in Figure 5-7, where the KLMS-VKS and QKLMS-VKS algorithms by having a single prediction model updates the kernel bandwidth to That set of samples. However, in the case we create multiple models, the sigma update is given for one of the cluster or prediction models. This can be seen clearly in this graph, where the 4 clusters created have a much lower value than for a single model, this is mainly due to the dynamics contained in each cluster differing from one another, thereby updating the Sigma has different values and threshold values.
5.3. Conclusions

The general idea of the KAF is to manage from an online learning forecast in future samples for a given prediction horizon. This proposal to adapt the bandwidth in a real-time filter schemes, facilitates tuning of this parameter, improving the results in terms of precision. For the simulations carried out, we find that the proposal has a competitive convergence rate with similar methods, with the advantage that the sigma values should not be known in advance.
6. Future work

During the development of this document, we carried out multiple tests of methodologies for the improvement of KAF. However, although substantial improvements were obtained, today it is still necessary to improve in real-time series. For this reason, to better the results, we leave to the reader’s consideration, some works for realizing to future:

1. The implementation for a filter set through a Multiples Kernel Learning unsupervised.

2. A sample selection scheme that limit dictionary size, where can improving the forecast results.

3. An online learning framework that selects the most appropriate bandwidth in schemes like NICE, without forgetting past bandwidths. This could allow adapting to time series that contain abrupt changes.
A. Appendix A: Performance metrics

In the different simulations presented in this document, the performance of the algorithms was evaluated based on the following metrics:

- **Mean Square Error (MSE):** is a performance metric that weights the quadratic difference between forecasts ($\hat{y}_t$) and the real values ($y_t$), averaged over the amount of data (n) as indicated in Equation (A-1)

  \[
  \text{MSE} = \frac{1}{n} \sum_{t=1}^{n} (\hat{y}_t - y_t)^2
  \]  

  \[(A-1)\]

- **Dictionary size:** this measure indicates the number of samples stored in the KAF dictionary or a neural network.

A.1. Simulations procedure.

- **Learning curve:** This curve is used to measure the variation of the MSE over the training of the KAF. The construction is the product of the filter response and update network throughout the training. In each iteration, the KAF update was used in the test dataset for calculated the MSE. Then the MSE is plotted against the number of iterations. Figure A-1 outlines the calculation procedure.

![Figure A-1: Learning curve.](image-url)
B. Appendix B: update of $\eta_*$

$$\eta_t = \eta_{t-1} + \beta \frac{\partial V_t}{\partial \eta_{t-1}}$$

$$\eta_t = \eta_{t-1} + \beta \frac{\partial}{\partial \eta_{t-1}} \left[ \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \right]$$

$$\eta_t = \eta_{t-1} + \beta \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \frac{\partial}{\partial \eta_{t-1}} \left[ -\frac{1}{2\lambda^2} e_t^2 \right]$$

$$\eta_t = \eta_{t-1} + \beta \left( -\frac{1}{2\lambda^2} \right) \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \frac{\partial}{\partial \eta_{t-1}} \left[ e_t^2 \right]$$

$$\eta_t = \eta_{t-1} - \left( \frac{\beta}{2\lambda^2} \right) \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \frac{\partial}{\partial \eta_{t-1}} \left[ (y_t - \hat{y}_t)^2 \right]$$

$$\eta_t = \eta_{t-1} - \left( \frac{\beta}{2\lambda^2} \right) \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \frac{\partial}{\partial \eta_{t-1}} \left[ y_t^2 - 2y_t\hat{y}_t + \hat{y}_t^2 \right]$$

$$\eta_t = \eta_{t-1} - \left( \frac{\beta}{2\lambda^2} \right) \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \left( \frac{\partial}{\partial \eta_{t-1}} \left[ y_t^2 \right] - 2 \frac{\partial}{\partial \eta_{t-1}} \left[ y_t\hat{y}_t \right] + \frac{\partial}{\partial \eta_{t-1}} \left[ \hat{y}_t^2 \right] \right)$$

For $\frac{\partial}{\partial \eta_{t-1}} \left[ y_t^2 \right] = 0$

$$\eta_t = \eta_{t-1} - \left( \frac{\beta}{2\lambda^2} \right) \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \left( -2y_t \frac{\partial}{\partial \eta_{t-1}} \left[ \hat{y}_t \right] + \frac{\partial}{\partial \eta_{t-1}} \left[ \hat{y}_t^2 \right] \right) \quad \text{(B-1)}$$

To facilitate the mathematical development, we will solve fractionally two derivatives:
1. $\frac{\partial}{\partial \eta_{t-1}} [\hat{y}_t]$, in which we replace the value of the output, obtaining:

$$\frac{\partial}{\partial \eta_{t-1}} [\hat{y}_t] = \frac{\partial}{\partial \eta_{t-1}} \left[ \sum_{i=1}^{n-1} \eta_i e_i \kappa \sigma \left( x_n, C_i^t \right) \right]$$

$$\frac{\partial}{\partial \eta_{n-1}} [\hat{y}_t] = \frac{\partial}{\partial \eta_{t-1}} \left[ \eta_1 e_1 \kappa \sigma \left( x_t, C_1^t \right) + \eta_2 e_2 \kappa \sigma \left( u_n, C_1^t \right) + \ldots + \eta_{t-1} e_{t-2} \kappa \sigma \left( x_t, C_1^t \right) \right]$$

For the above equation we will have all the derivatives will be 0, to disappointment of the last one. This is due to being partially derived with respect to $\eta_{t-1}$, resulting in:

$$\frac{\partial}{\partial \eta_{t-1}} [\hat{y}_t] = e_{t-1} \kappa \sigma \left( x_t, C_{t-1}^t \right)$$  \hspace{1cm} (B-2)

2. $\frac{\partial}{\partial \eta_{t-1}} [\hat{y}_t^2]$, in which we replace the value of the output, obtaining:

$$\frac{\partial}{\partial \eta_{t-1}} [\hat{y}_t^2] = \frac{\partial}{\partial \eta_{t-1}} \left[ \left( \sum_{i=1}^{t-1} \eta_i e_i \kappa \sigma \left( x_t, C_i^t \right) \right)^2 \right]$$

$$\frac{\partial}{\partial \eta_{t-1}} [\hat{y}_t^2] = 2 \left( \sum_{i=1}^{t-1} \eta_i e_i \kappa \sigma \left( x_t, C_i^t \right) \right) \frac{\partial}{\partial \eta_{t-1}} \left[ \left( \sum_{i=1}^{t-1} \eta_i e_i \kappa \sigma \left( x_n, C_i^t \right) \right) \right]$$

Where do we have to $\hat{y}_t = \left( \sum_{t=1}^{t-1} \eta_t e_t \kappa \sigma \left( x_t, C_t^t \right) \right)$

$$\frac{\partial}{\partial \eta_{t-1}} [\hat{y}_t^2] = 2 \hat{y}_t \frac{\partial}{\partial \eta_{t-1}} \left[ \eta_1 e_1 \kappa \sigma \left( x_n, C_1^t \right) + \ldots + \eta_{t-1} e_{t-2} \kappa \sigma \left( x_n, x_1^t \right) \right]$$

As in the case of the previous item, we will have all the derivatives will be 0, to the disappointment of the last one. This is due to being partially derived with respect to $\eta_{t-1}$, resulting in:

$$\frac{\partial}{\partial \eta_{t-1}} [\hat{y}_t] = 2 \hat{y}_t e_{t-1} \kappa \left( x_n, C_{t-1}^t \right)$$  \hspace{1cm} (B-3)
Replacing Equations (B-2) and (B-3) in Equation (B-1) we will have to:

\[ \eta_t = \eta_{t-1} - \beta \frac{e_t^2}{2\lambda^2} e_t \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \left[ -2e_t e_{t-1} \kappa_{\sigma} (x_t, C_{t-1}^t) + 2\hat{y}_t e_{t-1} \kappa_{\sigma} (x_n, C_{t-1}^t) \right] \]

Factoring we will obtain:

\[ \eta_t = \eta_{t-1} - \frac{\beta}{\lambda^2} e_t \exp \left( -\frac{e_t^2}{2\lambda^2} \right) e_{t-1} \kappa (x_n, C_{t-1}^t) \left( t_n - \hat{y}_t \right) \]

Where do we have to \( e_t = y_t - \hat{y}_t \)

\[ \eta_t = \eta_{t-1} + \beta \exp \left( -\frac{e_t^2}{2\lambda^2} \right) e_t e_{t-1} \kappa_{\sigma t-1} (x_{t-1}, C_{t-1}^{t-1}) \quad \text{(B-4)} \]

Where \( \eta_t \) is learning rate update, \( \eta_{t-1} \) is the pre-upgrade learning rate, \( \beta \) is the learning rate of the search for the optimal \( \eta^* \), \( e_t \) is the error of KAF, \( e_{t-1} \) it is the previous error of the KAF and \( \kappa_{\sigma t-1} (x_n, C_{t-1}^t) \) is the kernel among the incoming sample \( x_n \) and the last sample stored in the dictionary \( C_{t-1}^t \).
C. Appendix Bc: update of $\sigma_*$

$$
\sigma_t = \sigma_{t-1} + \beta \frac{\partial V_t}{\partial \sigma_{t-1}}
$$

$$
\sigma_t = \sigma_{t-1} + \beta \frac{\partial}{\partial \sigma_{t-1}} \left[ \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \right]
$$

$$
\sigma_t = \sigma_{t-1} + \beta \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \frac{\partial}{\partial \sigma_{t-1}} \left[ -\frac{e_t^2}{2\lambda^2} \right]
$$

$$
\sigma_t = \sigma_{t-1} + \left(-\frac{\beta}{2\lambda^2}\right) \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \frac{\partial}{\partial \sigma_{t-1}} \left[ e_t^2 \right]
$$

$$
\sigma_t = \sigma_{n-1} - \frac{\beta}{2\lambda^2} \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \frac{\partial}{\partial \sigma_{t-1}} \left[ (y_n - \hat{y}_t)^2 \right]
$$

$$
\sigma_t = \sigma_{n-1} - \frac{\beta}{2\lambda^2} \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \frac{\partial}{\partial \sigma_{t-1}} \left[ y_n^2 - 2y_n\hat{y}_t + \hat{y}_t^2 \right]
$$

$$
\sigma_t = \sigma_{n-1} - \frac{\beta}{2\lambda^2} \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \left( \frac{\partial}{\partial \sigma_{t-1}} \left[ y_n^2 \right] - 2 \frac{\partial}{\partial \sigma_{n-1}} \left[ y_n\hat{y}_t \right] + \frac{\partial}{\partial \sigma_{t-1}} \left[ \hat{y}_t^2 \right] \right)
$$

For $\frac{\partial}{\partial \sigma_{n-1}} [\hat{y}_t^2] = 0$

$$
\sigma_t = \sigma_{t-1} - \frac{\beta}{2\lambda^2} \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \left( -2 \frac{\partial}{\partial \sigma_{t-1}} \left[ y_n\hat{y}_t \right] + \frac{\partial}{\partial \sigma_{t-1}} \left[ \hat{y}_t^2 \right] \right)
$$

$$
\sigma_t = \sigma_{n-1} - \frac{\beta}{2\lambda^2} \exp \left( -\frac{e_t^2}{2\lambda^2} \right) \left( -2y_t \frac{\partial}{\partial \sigma_{t-1}} \left[ \hat{y}_t \right] + \frac{\partial}{\partial \sigma_{t-1}} \left[ \hat{y}_t^2 \right] \right)
$$

(C-1)

To facilitate the mathematical development, we will solve fractionally two derivatives:
1. $\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t]$, in which we replace the value of the output obtaining:

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \frac{\partial}{\partial \sigma_{t-1}} \left[ \sum_{i=1}^{t-1} \eta e_i \kappa \sigma_i \left( x_t, C^i_t \right) \right]$$

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \frac{\partial}{\partial \sigma_{t-1}} \left[ \eta e_1 \kappa \sigma_1 \left( x_t, C^1_t \right) \right] + \frac{\partial}{\partial \sigma_{n-1}} \left[ \eta e_2 \kappa \sigma_2 \left( x_t, C^2_t \right) \right] + \ldots + \frac{\partial}{\partial \sigma_{t-1}} \left[ \eta e_{n-1} \kappa \sigma_{n-1} \left( x_t, C^{t-1}_t \right) \right]$$

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \frac{\partial}{\partial \sigma_{t-1}} \left[ \eta e_{t-1} \kappa \sigma_{t-1} \left( x_t, C^{t-1}_t \right) \right]$$

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \frac{\partial}{\partial \sigma_{t-1}} \left[ \eta e_{n-1} \exp \left( -\frac{||x_t - C^{t-1}_t||^2}{2\sigma^2_{n-1}} \right) \right]$$

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \eta e_{t-1} \frac{\partial}{\partial \sigma_{t-1}} \left[ \exp \left( -\frac{||x_t - C^{t-1}_t||^2}{2\sigma^2_{n-1}} \right) \right]$$

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \eta e_{t-1} \exp \left( -\frac{||x_t - C^{t-1}_t||^2}{2\sigma^2_{t-1}} \right) \frac{\partial}{\partial \sigma_{t-1}} \left[ -\frac{||x_t - C^{t-1}_t||^2}{2\sigma^2_{t-1}} \right]$$

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \eta e_{t-1} \kappa \left( x_t, C^{t-1}_t \right) \left( \frac{\frac{\partial}{\partial \sigma_{t-1}} \left[ ||x_t - C^{t-1}_t||^2 \right]}{4 \sigma^4_{t-1}} \cdot 2 \sigma^2_{t-1} - \frac{\partial}{\partial \sigma_{t-1}} \left[ \frac{2 \sigma^2_{t-1}}{(2\sigma^2_{t-1})^2} \right] \right)$$

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \eta e_{t-1} \kappa \left( x_t, C^{t-1}_t \right) \frac{1}{4 \sigma^4_{t-1}} \left[ 2 \sigma^2_{t-1} \right] ||x_t - C^{t-1}_t||^2$$

$$\frac{\partial}{\partial \sigma_{n-1}} [\hat{y}_t] = \eta e_{t-1} \kappa \left( x_t, C^{t-1}_t \right) \frac{1}{4 \sigma^4_{t-1}} \left[ 4 \sigma_{t-1} \right] ||x_t - C^{t-1}_t||^2$$

$$\frac{\partial}{\partial \sigma_{t-1}} [\hat{y}_t] = \eta e_{t-1} \kappa \left( x_t, C^{t-1}_t \right) \frac{1}{\sigma^3_{t-1}} ||x_t - C^{t-1}_t||^2 \quad (C-2)$$
2. \( \frac{\partial}{\partial \sigma_{n-1}} [\tilde{y}_t^2] \), in which we replace the value of the output obtaining:

\[
\frac{\partial}{\partial \sigma_{t-1}} [\tilde{y}_t^2] = \frac{\partial}{\partial \sigma_{t-1}} \left[ \left( \sum_{i=1}^{t-1} \eta e_i \kappa_{\sigma_i} (u_t, C_t) \right)^2 \right]
\]

\[
\frac{\partial}{\partial \sigma_{t-1}} [\tilde{y}_t^2] = 2 \left( \sum_{i=1}^{t-1} \eta e_i \kappa_{\sigma_i} (x_t, C_t) \right) \frac{\partial}{\partial \sigma_{t-1}} \left[ \left( \sum_{i=1}^{t-1} \eta e_i \kappa_{\sigma_i} (u_t, C_t) \right) \right]
\]

The derive \( \frac{\partial}{\partial \sigma_{t-1}} \left[ \left( \sum_{i=1}^{t-1} \eta e_i \kappa_{\sigma_i} (x_t, C_t) \right) \right] \) is calculated in the previous item, with which we will obtain the following result:

\[
\frac{\partial}{\partial \sigma_{t-1}} [\tilde{y}_t^2] = 2 \tilde{y}_t \eta e_{t-1} \kappa_{\sigma_{t-1}} (x_t, C_t) \frac{1}{\sigma_{t-1}^3} \| u_t - C_{t-1}^{\sigma_t} \|^2 \tag{C-3}
\]

Replacing Equations (C-2) and (C-3) in Equation (C-1) we will have to:

\[
\sigma_t = \sigma_{t-1} - \frac{\beta}{2 \lambda^2} \exp \left( - \frac{e_t^2}{2 \lambda^2} \right) \left( -2 \tilde{y}_t \eta e_{t-1} \kappa_{\sigma_{t-1}} (x_t, C_{t-1}^{\sigma_{t-1}}) \frac{1}{\sigma_{t-1}^3} \| u_t - C_{t-1}^{\sigma_{t-1}} \|^2 + 2 \tilde{y}_t \eta e_{t-1} \kappa_{\sigma_{t-1}} (x_t, C_{t-1}^{\sigma_{t-1}}) \frac{1}{\sigma_{n-1}^3} \| u_t - C_{t-1}^{\sigma_{t-1}} \|^2 \right)
\]

Factoring we will tend:

\[
\sigma_t = \sigma_{t-1} - \frac{\beta}{2 \lambda^2} \exp \left( - \frac{e_t^2}{2 \lambda^2} \right) 2 \eta e_{t-1} \kappa_{\sigma_{t-1}} (x_t, C_{n-1}^{\sigma_{t-1}}) \frac{1}{\sigma_{t-1}^3} \| x_t - C_{t-1}^{\sigma_{t-1}} \|^2 (-y_t + \tilde{y}_t)
\]

Factoring (\(-\)):

\[
\sigma_t = \sigma_{t-1} + \frac{\beta}{2 \lambda^2} \exp \left( - \frac{e_t^2}{2 \lambda^2} \right) 2 \eta e_{t-1} \kappa_{\sigma_{t-1}} (x_t, C_{t-1}^{\sigma_{t-1}}) \frac{1}{\sigma_{n-1}^3} \| x_t - C_{t-1}^{\sigma_{t-1}} \|^2 (y_t + \tilde{y}_t)
\]

Finally we will have:

\[
\sigma_t = \sigma_{n-1} + \frac{\beta}{\lambda^2 \sigma_{n-1}^3} \exp \left( - \frac{e_t^2}{2 \lambda^2} \right) \eta e_{t-1} e_{t-1} \| x_t - C_{t-1}^{\sigma_{t-1}} \|^2 \kappa_{\sigma_{t-1}} (x_t, C_{n-1}^{\sigma_{t-1}}) \tag{C-4}
\]

Where \( \sigma_t \) is the step size update, \( \sigma_{t-1} \) is the pre-upgrade learning rate, \( \beta \) is the learning rate of the search for the optimal \( \sigma^* \), \( e_t \) it is the error of the KAF, \( e_{t-1} \) it is the previous error of the KAF, \( \eta_{t-1} \) is the previous rate prior to the search and \( \kappa_{\sigma_{t-1}} (x_t, C_{t-1}^{\sigma_{t-1}}) \) is the kernel among the incoming sample \( x_t \) and the last sample stored in the dictionary \( C_{t-1}^{\sigma_{t-1}} \).
Bibliografía


