Time Series Forecasting Using Kernel Regression Methods

José Daniel de Alencar Santos (IFCE)
Allan Kelvin Mendes de Sales (IFCE)
Guilherme de Alencar Barreto (UFC)

Graduate Program in Teleinformatics Engineering (PPGETI)
Federal University of Ceará (UFC), Fortaleza, Ceará, Brazil
jdaniel@ifce.edu.br, kelvin@ifce.edu.br, gbarreto@ufc.br

Fortaleza, October 2022

Outline

Related Publications

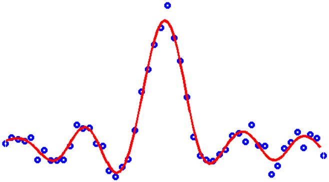
- J. D. A. Santos & G. A. Barreto (2018). Novel sparse LSSVR models in primal weight space for robust system identification with outliers. **Journal of Process Control**, vol. 67, p. 129-140.

- J. D. A. Santos & G. A. Barreto (2017). An outlier-robust kernel RLS algorithm for nonlinear system identification. **Nonlinear Dynamics**, vol. 90, p. 1707-1726.

- A. K. M. Sales & G. A. Barreto (2022). A novel method for sparsification of kernel adaptive filters. **IEEE Transactions on Neural Networks and Learning Systems**, submitted.

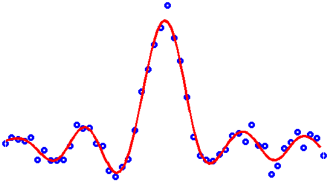
Kernel-Based Regression Models

- In several application domains, a nonlinear model has to be fitted to data samples generated by nonlinear phenomena.
- This a very demanding task because there are many alternatives out there, such as neural networks, polinomial model, Gaussian process (GP) models, and so on.
- A natural (or would it be unnatural?) alternative involves kernel based models, such as support vector regression.



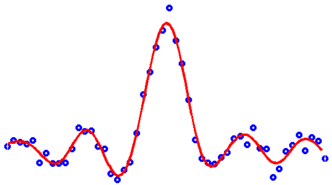
Kernel-Based Regression Models

- In several application domains, a nonlinear model has to be fitted to data samples generated by nonlinear phenomena.
- This a very demanding task because there are many alternatives out there, such as neural networks, polinomial model, Gaussian process (GP) models, and so on.
- A natural (or would it be unnatural?) alternative involves kernel based models, such as support vector regression.

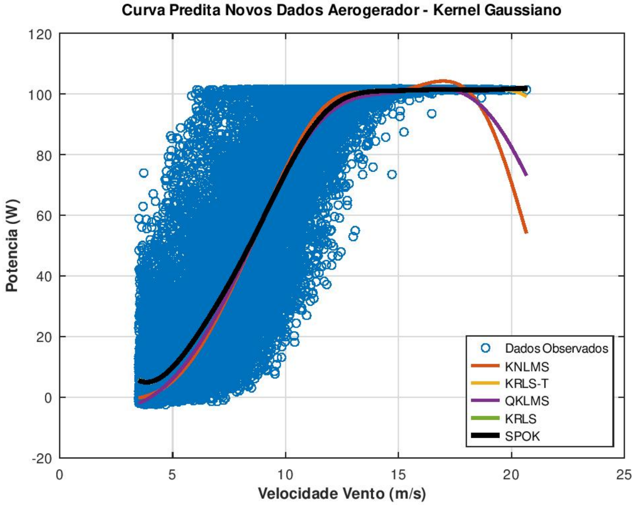


Kernel-Based Regression Models

- In several application domains, a nonlinear model has to be fitted to data samples generated by nonlinear phenomena.
- This a very demanding task because there are many alternatives out there, such as neural networks, polinomial model, Gaussian process (GP) models, and so on.
- A natural (or would it be unnatural?) alternative involves kernel based models, such as support vector regression.

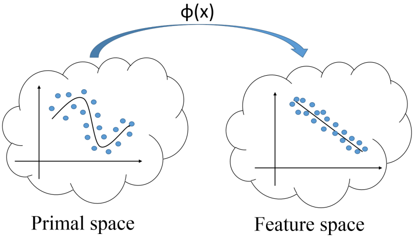


Kernel-Based Regression Models



Kernel-Based Regression Models

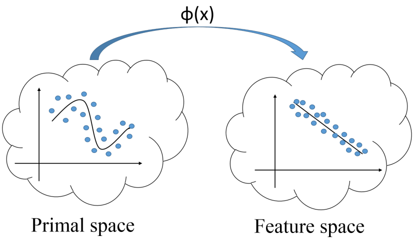
- The least squares support vector regression (**LSSVR**) and the kernel ordinary least squares (KOLS) [1] models are very competitive tools for nonlinear regression tasks.
- It is easy to understand their backgrounds from the basic concepts of linear regression and least squares estimation.
- Kernel models build a linear model in the feature (rkhs) space using an unknown nonlinear mapping ϕ.



[1] Suykens, J. A. K. et al. *Least Squares Support Vector Machines*. New Jersey, USA: World Scientific, 2002.

Kernel-Based Regression Models

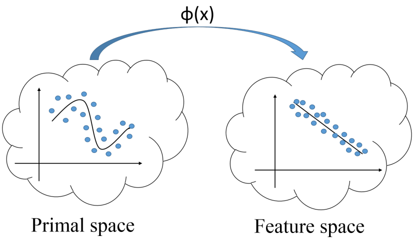
- The least squares support vector regression (**LSSVR**) and the kernel ordinary least squares (KOLS) [1] models are very competitive tools for nonlinear regression tasks.
- It is easy to understand their backgrounds from the basic concepts of linear regression and least squares estimation.
- Kernel models build a linear model in the feature (rkhs) space using an unknown nonlinear mapping ϕ.



[1] Suykens, J. A. K. et al. *Least Squares Support Vector Machines*. New Jersey, USA: World Scientific, 2002.

Kernel-Based Regression Models

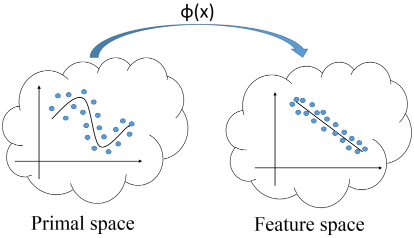
- The least squares support vector regression (**LSSVR**) and the kernel ordinary least squares (KOLS) [1] models are very competitive tools for nonlinear regression tasks.
- It is easy to understand their backgrounds from the basic concepts of linear regression and least squares estimation.
- Kernel models build a linear model in the feature (rkhs) space using an unknown nonlinear mapping ϕ.



[1] Suykens, J. A. K. et al. *Least Squares Support Vector Machines*. New Jersey, USA: World Scientific, 2002.

Kernel-Based Regression Models

- The least squares support vector regression (**LSSVR**) and the kernel ordinary least squares (KOLS) [1] models are very competitive tools for nonlinear regression tasks.
- It is easy to understand their backgrounds from the basic concepts of linear regression and least squares estimation.
- Kernel models build a linear model in the feature (rkhs) space using an unknown nonlinear mapping ϕ.



[1] Suykens, J. A. K. et al. *Least Squares Support Vector Machines*. New Jersey, USA: World Scientific, 2002.

General and Specific Objectives

General Objective

The overall objective of this talk is to introduce kernel-based nonlinear regression models and their applications to time series forecasting.

Specific Objectives

1. To describe the basics of the LSSVR model.
2. To describe the basics of the kernel ordinary least squares (KOLS).
3. To describe the basics of the kernel regularized least squares (KRELS).
4. To discuss Octave/Matlab codes of the aforementioned models.
5. To present some applications on time series forecasting.

General and Specific Objectives

General Objective

The overall objective of this talk is to introduce kernel-based nonlinear regression models and their applications to time series forecasting.

Specific Objectives

1. To describe the basics of the LSSVR model.
2. To describe the basics of the kernel ordinary least squares (KOLS).
3. To describe the basics of the kernel regularized least squares (KRELS).
4. To discuss Octave/Matlab codes of the aforementioned models.
5. To present some applications on time series forecasting.

General and Specific Objectives

General Objective

The overall objective of this talk is to introduce kernel-based nonlinear regression models and their applications to time series forecasting.

Specific Objectives

1. To describe the basics of the LSSVR model.
2. To describe the basics of the kernel ordinary least squares (KOLS).
3. To describe the basics of the kernel regularized least squares (KRELS).
4. To discuss Octave/Matlab codes of the aforementioned models.
5. To present some applications on time series forecasting.

General and Specific Objectives

General Objective

The overall objective of this talk is to introduce kernel-based nonlinear regression models and their applications to time series forecasting.

Specific Objectives

1. To describe the basics of the LSSVR model.
2. To describe the basics of the kernel ordinary least squares (KOLS).
3. To describe the basics of the kernel regularized least squares (KRELS).
4. To discuss Octave/Matlab codes of the aforementioned models.
5. To present some applications on time series forecasting.

General and Specific Objectives

General Objective

The overall objective of this talk is to introduce kernel-based nonlinear regression models and their applications to time series forecasting.

Specific Objectives

1. To describe the basics of the LSSVR model.
2. To describe the basics of the kernel ordinary least squares (KOLS).
3. To describe the basics of the kernel regularized least squares (KRELS).
4. To discuss Octave/Matlab codes of the aforementioned models.
5. To present some applications on time series forecasting.

General and Specific Objectives

General Objective

The overall objective of this talk is to introduce kernel-based nonlinear regression models and their applications to time series forecasting.

Specific Objectives

1. To describe the basics of the LSSVR model.
2. To describe the basics of the kernel ordinary least squares (KOLS).
3. To describe the basics of the kernel regularized least squares (KRELS).
4. To discuss Octave/Matlab codes of the aforementioned models.
5. To present some applications on time series forecasting.

General and Specific Objectives

General Objective

The overall objective of this talk is to introduce kernel-based nonlinear regression models and their applications to time series forecasting.

Specific Objectives

1. To describe the basics of the LSSVR model.
2. To describe the basics of the kernel ordinary least squares (KOLS).
3. To describe the basics of the kernel regularized least squares (KRELS).
4. To discuss Octave/Matlab codes of the aforementioned models.
5. To present some applications on time series forecasting.

Outline

LSSVR Primal Problem

- Given an estimation (a.k.a. training) dataset $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$, with $\boldsymbol{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$, the kernel regression problem is given by

$$f(\boldsymbol{x}_n) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b, \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^{d_h}$ is the unknown parameter vector, $b \in \mathbb{R}$ is a bias and $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d_h}$ is a nonlinear map into the feature space.

- The primal optimization problem of the LSSVR model is given by

$$\min_{\boldsymbol{w},b,\boldsymbol{e}} J_p(\boldsymbol{w}, \boldsymbol{e}) = \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{smoothness}} + \underbrace{\gamma \frac{1}{2} \sum_{n=1}^{N} e_n^2}_{\text{training errors}}, \tag{2}$$

$$\text{subject to} \left\{ y_n = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b + e_n, \text{ for } n = 1, \ldots, N, \right. \tag{3}$$

where γ is the regularization parameter and e_n is the n-th error.

LSSVR Primal Problem

- Given an estimation (a.k.a. training) dataset $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$, with $\boldsymbol{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$, the kernel regression problem is given by

$$f(\boldsymbol{x}_n) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b, \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^{d_h}$ is the unknown parameter vector, $b \in \mathbb{R}$ is a bias and $\boldsymbol{\phi}(\cdot) : \mathbb{R}^d \to \mathbb{R}^{d_h}$ is a nonlinear map into the feature space.

- The primal optimization problem of the LSSVR model is given by

$$\min_{\boldsymbol{w}, b, \boldsymbol{e}} J_p(\boldsymbol{w}, \boldsymbol{e}) = \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{smoothness}} + \underbrace{\gamma \frac{1}{2} \sum_{n=1}^N e_n^2}_{\text{training errors}}, \tag{2}$$

$$\text{subject to} \left\{ \; y_n = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b + e_n, \;\; \text{for } n = 1, \ldots, N, \right. \tag{3}$$

where γ is the regularization parameter and e_n is the n-th error.

LSSVR Dual Problem

- The LSSVR dual problem is obtained by building the Lagrangian, as

$$\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{e}, \boldsymbol{\alpha}) := \frac{1}{2}\|\boldsymbol{w}\|^2 + \gamma\frac{1}{2}\sum_{n=1}^{N} e_n^2 - \sum_{n=1}^{N} \alpha_n(\boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}_n) + b + e_n - y_n), \tag{4}$$

where α_n are the Lagrange multipliers.

- From the optimality conditions[2], one gets

$$\underbrace{\begin{bmatrix} 0 & \mathbf{1}_N^\top \\ \mathbf{1}_N & \boldsymbol{K} + \gamma^{-1}\boldsymbol{I}_N \end{bmatrix}}_{\boldsymbol{\Omega}} \underbrace{\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix}}_{\boldsymbol{\alpha}_o} = \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{y} \end{bmatrix}}_{\boldsymbol{y}_o}. \tag{5}$$

- $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ is the **kernel matrix**, whose entries are

$$K_{i,j} := k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)^\top\boldsymbol{\phi}(\boldsymbol{x}_j), \text{ for } i, j = 1, \ldots, N, \tag{6}$$

where $k(\cdot,\cdot)$ is the chosen kernel function.

[2] FLETCHER, R. **Practical methods of optimization**. New Jersey, USA: John Wiley & Sons, 2013.

LSSVR Dual Problem

- The LSSVR dual problem is obtained by building the Lagrangian, as

$$\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{e}, \boldsymbol{\alpha}) := \frac{1}{2}\|\boldsymbol{w}\|^2 + \gamma\frac{1}{2}\sum_{n=1}^{N} e_n^2 - \sum_{n=1}^{N} \alpha_n(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b + e_n - y_n), \quad (4)$$

  where α_n are the Lagrange multipliers.
- From the optimality conditions[2], one gets

$$\underbrace{\begin{bmatrix} 0 & \mathbf{1}_N^\top \\ \mathbf{1}_N & \boldsymbol{K} + \gamma^{-1}\boldsymbol{I}_N \end{bmatrix}}_{\boldsymbol{\Omega}} \underbrace{\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix}}_{\boldsymbol{\alpha}_o} = \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{y} \end{bmatrix}}_{\boldsymbol{y}_o}. \quad (5)$$

- $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is the **kernel matrix**, whose entries are

$$K_{i,j} := k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j), \text{ for } i, j = 1, \ldots, N, \quad (6)$$

  where $k(\cdot, \cdot)$ is the chosen kernel function.

[2] FLETCHER, R. **Practical methods of optimization**. New Jersey, USA: John Wiley & Sons, 2013.

LSSVR Dual Problem

- The LSSVR dual problem is obtained by building the Lagrangian, as

$$\mathscr{L}(\boldsymbol{w}, b, \boldsymbol{e}, \boldsymbol{\alpha}) := \frac{1}{2}\|\boldsymbol{w}\|^2 + \gamma\frac{1}{2}\sum_{n=1}^{N} e_n^2 - \sum_{n=1}^{N} \alpha_n(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b + e_n - y_n), \quad (4)$$

where α_n are the Lagrange multipliers.
- From the optimality conditions[2], one gets

$$\underbrace{\begin{bmatrix} 0 & \mathbf{1}_N^\top \\ \mathbf{1}_N & \boldsymbol{K} + \gamma^{-1}\boldsymbol{I}_N \end{bmatrix}}_{\boldsymbol{\Omega}} \underbrace{\begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix}}_{\boldsymbol{\alpha}_o} = \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{y} \end{bmatrix}}_{\boldsymbol{y}_o}. \quad (5)$$

- $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is the **kernel matrix**, whose entries are

$$K_{i,j} := k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j), \text{ for } i, j = 1, \ldots, N, \quad (6)$$

where $k(\cdot, \cdot)$ is the chosen kernel function.

[2] FLETCHER, R. **Practical methods of optimization**. New Jersey, USA: John Wiley & Sons, 2013.

The Kernel Matrix

- The kernel matrix $\boldsymbol{K} = [K_{i,j}]_{N\times N}$ is defined as

$$\boldsymbol{K} = \begin{bmatrix} \phi(\boldsymbol{x}_1)^\top\phi(\boldsymbol{x}_1) & \phi(\boldsymbol{x}_1)^\top\phi(\boldsymbol{x}_2) & \cdots & \phi(\boldsymbol{x}_1)^\top\phi(\boldsymbol{x}_N) \\ \phi(\boldsymbol{x}_2)^\top\phi(\boldsymbol{x}_1) & \phi(\boldsymbol{x}_2)^\top\phi(\boldsymbol{x}_2) & \cdots & \phi(\boldsymbol{x}_2)^\top\phi(\boldsymbol{x}_N) \\ \vdots & \vdots & \cdots & \vdots \\ \phi(\boldsymbol{x}_N)^\top\phi(\boldsymbol{x}_1) & \phi(\boldsymbol{x}_N)^\top\phi(\boldsymbol{x}_2) & \cdots & \phi(\boldsymbol{x}_N)^\top\phi(\boldsymbol{x}_N) \end{bmatrix}_{N\times N} \tag{7}$$

$$= \begin{bmatrix} k(\boldsymbol{x}_1,\boldsymbol{x}_1) & k(\boldsymbol{x}_1,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ k(\boldsymbol{x}_2,\boldsymbol{x}_1) & k(\boldsymbol{x}_2,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2,\boldsymbol{x}_N) \\ \vdots & \vdots & \cdots & \vdots \\ k(\boldsymbol{x}_N,\boldsymbol{x}_1) & k(\boldsymbol{x}_N,\boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{bmatrix}_{N\times N} \tag{8}$$

The Kernel Matrix

- The kernel matrix $\boldsymbol{K}$ must be positive-definite.
- It is a Gram matrix; that is, a matrix of dot products.
- It is symmetric. Thus, its computation can be optimized for speed.
- There are lots of kernel functions that can be used.

The Kernel Matrix

- The kernel matrix $\boldsymbol{K}$ must be positive-definite.
- It is a Gram matrix; that is, a matrix of dot products.
- It is symmetric. Thus, its computation can be optimized for speed.
- There are lots of kernel functions that can be used.

The Kernel Matrix

- The kernel matrix $\boldsymbol{K}$ must be positive-definite.
- It is a Gram matrix; that is, a matrix of dot products.
- It is symmetric. Thus, its computation can be optimized for speed.
- There are lots of kernel functions that can be used.

The Kernel Matrix

- The kernel matrix $\boldsymbol{K}$ must be positive-definite.
- It is a Gram matrix; that is, a matrix of dot products.
- It is symmetric. Thus, its computation can be optimized for speed.
- There are lots of kernel functions that can be used.

The Kernel Matrix

- The linear kernel function: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$.
- The Gaussian (or RBF) kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\lambda^2}\right), \tag{9}$$

  where $\lambda > 0$ is the width of the function.
- The polynomial kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (c + \boldsymbol{x}_i^\top \boldsymbol{x}_j)^d, \tag{10}$$

  where c is a constant and $d \geq 1$ is the order of the polynomial.
- The sigmoidal (or MLP) kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(a\boldsymbol{x}_i^\top \boldsymbol{x}_j + r), \tag{11}$$

  where a (slope) and r (bias) are constants.

The Kernel Matrix

- The linear kernel function: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$.
- The Gaussian (or RBF) kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\lambda^2}\right), \tag{9}$$

  where $\lambda > 0$ is the width of the function.
- The polynomial kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (c + \boldsymbol{x}_i^\top \boldsymbol{x}_j)^d, \tag{10}$$

  where c is a constant and $d \geq 1$ is the order of the polynomial.
- The sigmoidal (or MLP) kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(a\boldsymbol{x}_i^\top \boldsymbol{x}_j + r), \tag{11}$$

  where a (slope) and r (bias) are constants.

The Kernel Matrix

- The linear kernel function: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$.
- The Gaussian (or RBF) kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\lambda^2}\right), \tag{9}$$

  where $\lambda > 0$ is the width of the function.
- The polynomial kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (c + \boldsymbol{x}_i^\top \boldsymbol{x}_j)^d, \tag{10}$$

  where c is a constant and $d \geq 1$ is the order of the polynomial.
- The sigmoidal (or MLP) kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(a\boldsymbol{x}_i^\top \boldsymbol{x}_j + r), \tag{11}$$

  where a (slope) and r (bias) are constants.

The Kernel Matrix

- The linear kernel function: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$.
- The Gaussian (or RBF) kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\lambda^2}\right), \tag{9}$$

  where $\lambda > 0$ is the width of the function.
- The polynomial kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (c + \boldsymbol{x}_i^\top \boldsymbol{x}_j)^d, \tag{10}$$

  where c is a constant and $d \geq 1$ is the order of the polynomial.
- The sigmoidal (or MLP) kernel function:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(a\boldsymbol{x}_i^\top \boldsymbol{x}_j + r), \tag{11}$$

  where a (slope) and r (bias) are constants.

LSSVR Dual Problem

- Kernel-based predictors:

$$\underbrace{f(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + b,}_{\text{primal space}} \tag{12}$$

or

$$\underbrace{\begin{aligned} f(\boldsymbol{x}) &= \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b, \\ &= \boldsymbol{\alpha}^\top \boldsymbol{K}(\boldsymbol{x}, \mathcal{D}) + b. \end{aligned}}_{\text{dual space}} \tag{13}$$

LSSVR Dual Problem

- Kernel-based predictors:

$$\underbrace{f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + b,}_{\text{primal space}} \tag{12}$$

or

$$\underbrace{\begin{aligned} f(\boldsymbol{x}) &= \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b, \\ &= \boldsymbol{\alpha}^\top \boldsymbol{K}(\boldsymbol{x}, \mathcal{D}) + b. \end{aligned}}_{\text{dual space}} \tag{13}$$

LSSVR Dual Problem

- Kernel-based predictors:

$$\underbrace{f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + b,}_{\text{primal space}} \tag{12}$$

or

$$\underbrace{\begin{aligned} f(\boldsymbol{x}) &= \sum_{n=1}^{N} \alpha_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b, \\ &= \boldsymbol{\alpha}^\top \boldsymbol{K}(\boldsymbol{x}, \mathcal{D}) + b. \end{aligned}}_{\text{dual space}} \tag{13}$$

Outline

The KOLS Models

- The KRLS model ($f(\boldsymbol{x}_i) = \boldsymbol{\phi}^\top(\boldsymbol{x}_i)\boldsymbol{w}$) minimizes the cost function

$$J(\boldsymbol{w}) = \sum_{i=1}^{N}(y_i - f(\boldsymbol{x}_i))^2 = \|\boldsymbol{y} - \boldsymbol{\Phi}^\top \boldsymbol{w}\|^2, \tag{14}$$

which can be rewritten as

$$J(\boldsymbol{\alpha}) = \|\boldsymbol{y} - \boldsymbol{K}_t \boldsymbol{\alpha}\|^2. \tag{15}$$

where $\boldsymbol{K} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ is the kernel matrix built using the N training samples.

- Theoretically, the KOLS model solution can be given by $\boldsymbol{\alpha} = \boldsymbol{K}^{-1}\boldsymbol{y}$, by simply setting $b = 0$ and $\gamma \to \infty$ in Eq. (5).

The KOLS Models

- The KRLS model $(f(\boldsymbol{x}_i) = \boldsymbol{\phi}^\top(\boldsymbol{x}_i)\boldsymbol{w})$ minimizes the cost function

$$J(\boldsymbol{w}) = \sum_{i=1}^{N}(y_i - f(\boldsymbol{x}_i))^2 = \|\boldsymbol{y} - \boldsymbol{\Phi}^\top \boldsymbol{w}\|^2, \tag{14}$$

  which can be rewritten as

$$J(\boldsymbol{\alpha}) = \|\boldsymbol{y} - \boldsymbol{K}_t\boldsymbol{\alpha}\|^2. \tag{15}$$

  where $\boldsymbol{K} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ is the kernel matrix built using the N training samples.
- Theoretically, the KOLS model solution can be given by $\boldsymbol{\alpha} = \boldsymbol{K}^{-1}\boldsymbol{y}$, by simply setting $b = 0$ and $\gamma \to \infty$ in Eq. (5).

Sparsifying the KOLS Model

- As mentioned before, the size of the kernel matrix and, hence, the dimension of the vector of parameters $\boldsymbol{\alpha}$ is determined by the number of training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$.
- To avoid inversion of huge kernel matrices, we must use some kind of spasification method to select a subset of relevant sample pairs to compose a dictionary.
- In the ALD[3] criterion, when a new incoming sample $\boldsymbol{x}_t$ is available, one must verify if $\phi(\boldsymbol{x}_t)$ is ALD on the **dictionary** $\mathcal{D}_{t-1}^{sv} = \{\tilde{\boldsymbol{x}}_j\}_{j=1}^{m_t-1}$.
- One should estimate $\boldsymbol{a} = [a_1, \dots, a_{m_{t-1}}]^\top$ satisfying the ALD criterion

$$\delta_t \triangleq \min_{\boldsymbol{a}} \left\| \sum_{m=1}^{m_{t-1}} a_m \phi(\tilde{\boldsymbol{x}}_m) - \phi(\boldsymbol{x}_t) \right\|^2 \leq \nu, \qquad (16)$$

where ν is the sparsity level (hyper-)parameter.

[3] ENGEL, Y.; MANNOR, S.; MEIR, R. **Sparse online greedy support vector regression**. In: ELOMAA, T.; MANNILA, H.; TOIVONEN, H. (Ed.). Proceedings of the 13th European Conference on Machine Learning (ECML'2002). Helsinki, Finland, 2002. p. 84–96.

Sparsifying the KOLS Model

- As mentioned before, the size of the kernel matrix and, hence, the dimension of the vector of parameters $\boldsymbol{\alpha}$ is determined by the number of training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$.
- To avoid inversion of huge kernel matrices, we must use some kind of spasification method to select a subset of relevant sample pairs to compose a dictionary.
- In the ALD[3] criterion, when a new incoming sample $\boldsymbol{x}_t$ is available, one must verify if $\phi(\boldsymbol{x}_t)$ is ALD on the **dictionary** $\mathcal{D}_{t-1}^{sv} = \{\tilde{\boldsymbol{x}}_j\}_{j=1}^{m_t-1}$.
- One should estimate $\boldsymbol{a} = [a_1, \ldots, a_{m_{t-1}}]^\top$ satisfying the ALD criterion

$$\delta_t \triangleq \min_{\boldsymbol{a}} \left\| \sum_{m=1}^{m_{t-1}} a_m \phi(\tilde{\boldsymbol{x}}_m) - \phi(\boldsymbol{x}_t) \right\|^2 \leq \nu, \qquad (16)$$

where ν is the sparsity level (hyper-)parameter.

[3] ENGEL, Y.; MANNOR, S.; MEIR, R. **Sparse online greedy support vector regression**. In: ELOMAA, T.; MANNILA, H.; TOIVONEN, H. (Ed.). Proceedings of the 13th European Conference on Machine Learning (ECML'2002). Helsinki, Finland, 2002. p. 84–96.

Sparsifying the KOLS Model

- As mentioned before, the size of the kernel matrix and, hence, the dimension of the vector of parameters $\boldsymbol{\alpha}$ is determined by the number of training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$.
- To avoid inversion of huge kernel matrices, we must use some kind of spasification method to select a subset of relevant sample pairs to compose a dictionary.
- In the ALD[3] criterion, when a new incoming sample $\boldsymbol{x}_t$ is available, one must verify if $\phi(\boldsymbol{x}_t)$ is ALD on the **dictionary** $\mathcal{D}_{t-1}^{sv} = \{\tilde{\boldsymbol{x}}_j\}_{j=1}^{m_t-1}$.
- One should estimate $\boldsymbol{a} = [a_1, \ldots, a_{m_{t-1}}]^\top$ satisfying the ALD criterion

$$\delta_t \triangleq \min_{\boldsymbol{a}} \left\| \sum_{m=1}^{m_{t-1}} a_m \phi(\tilde{\boldsymbol{x}}_m) - \phi(\boldsymbol{x}_t) \right\|^2 \leq \nu, \quad (16)$$

where ν is the sparsity level (hyper-)parameter.

[3] ENGEL, Y.; MANNOR, S.; MEIR, R. **Sparse online greedy support vector regression**. In: ELOMAA, T.; MANNILA, H.; TOIVONEN, H. (Ed.). Proceedings of the 13th European Conference on Machine Learning (ECML'2002). Helsinki, Finland, 2002. p. 84-96.

Sparsifying the KOLS Model

- As mentioned before, the size of the kernel matrix and, hence, the dimension of the vector of parameters $\boldsymbol{\alpha}$ is determined by the number of training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$.
- To avoid inversion of huge kernel matrices, we must use some kind of spasification method to select a subset of relevant sample pairs to compose a dictionary.
- In the ALD[3] criterion, when a new incoming sample $\boldsymbol{x}_t$ is available, one must verify if $\phi(\boldsymbol{x}_t)$ is ALD on the **dictionary** $\mathcal{D}_{t-1}^{sv} = \{\tilde{\boldsymbol{x}}_j\}_{j=1}^{m_t-1}$.
- One should estimate $\boldsymbol{a} = [a_1, \ldots, a_{m_{t-1}}]^\top$ satisfying the ALD criterion

$$\delta_t \triangleq \min_{\boldsymbol{a}} \left\| \sum_{m=1}^{m_{t-1}} a_m \boldsymbol{\phi}(\tilde{\boldsymbol{x}}_m) - \boldsymbol{\phi}(\boldsymbol{x}_t) \right\|^2 \leq \nu, \qquad (16)$$

where ν is the sparsity level (hyper-)parameter.

[3] ENGEL, Y.; MANNOR, S.; MEIR, R. **Sparse online greedy support vector regression**. In: ELOMAA, T.; MANNILA, H.; TOIVONEN, H. (Ed.). Proceedings of the 13th European Conference on Machine Learning (ECML'2002). Helsinki, Finland, 2002. p. 84-96.

Sparsifying the KOLS Model

- In terms of the kernel matrix, the Eq. (16) can be rewritten as

$$\boldsymbol{a}_t = \tilde{\boldsymbol{K}}_{t-1}^{-1}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t), \quad \text{and} \quad \delta_t = k_{tt} - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top \boldsymbol{a}_t \leq \nu, \tag{17}$$

  where ν is the sparsity level parameter. The 1st element in the dictionary is chosen at random.

- Once a pass of the ALD criterion is concluded, we can rewrite the problem as

$$J(\tilde{\boldsymbol{\alpha}}) = \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{A}}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\alpha}}\|^2, \tag{18}$$

  where $\tilde{\boldsymbol{A}} = [\boldsymbol{a}_1 \ \boldsymbol{a}_2 \ \ldots \ \boldsymbol{a}_{m_t}]^\top \in \mathbb{R}^{t \times m_t}$ and $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^{m_t}$.

- Then, computing $\partial J(\tilde{\boldsymbol{\alpha}})/\partial \tilde{\boldsymbol{\alpha}} = \mathbf{0}$, one gets

$$\tilde{\boldsymbol{\alpha}} = \tilde{\boldsymbol{K}}^{-1}\tilde{\boldsymbol{P}}\tilde{\boldsymbol{A}}^\top\tilde{\boldsymbol{y}}, \tag{19}$$

  where $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{A}}^\top\tilde{\boldsymbol{A}})^{-1}$.

Sparsifying the KOLS Model

- In terms of the kernel matrix, the Eq. (16) can be rewritten as

$$\boldsymbol{a}_t = \tilde{\boldsymbol{K}}_{t-1}^{-1}\tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t), \text{ and } \delta_t = k_{tt} - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top \boldsymbol{a}_t \leq \nu, \quad (17)$$

  where ν is the sparsity level parameter. The 1st element in the dictionary is chosen at random.

- Once a pass of the ALD criterion is concluded, we can rewrite the problem as

$$J(\tilde{\boldsymbol{\alpha}}) = \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{A}}\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\alpha}}\|^2, \quad (18)$$

  where $\tilde{\boldsymbol{A}} = [\boldsymbol{a}_1 \ \boldsymbol{a}_2 \ \ldots \ \boldsymbol{a}_{m_t}]^\top \in \mathbb{R}^{t \times m_t}$ and $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^{m_t}$.

- Then, computing $\partial J(\tilde{\alpha})/\partial\tilde{\alpha} = \mathbf{0}$, one gets

$$\tilde{\alpha} = \tilde{\boldsymbol{K}}^{-1}\tilde{\boldsymbol{P}}\tilde{\boldsymbol{A}}^\top\tilde{\boldsymbol{y}}, \quad (19)$$

  where $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{A}}^\top\tilde{\boldsymbol{A}})^{-1}$.

Sparsifying the KOLS Model

- In terms of the kernel matrix, the Eq. (16) can be rewritten as

$$\boldsymbol{a}_t = \tilde{\boldsymbol{K}}_{t-1}^{-1} \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t), \quad \text{and} \quad \delta_t = k_{tt} - \tilde{\boldsymbol{k}}_{t-1}(\boldsymbol{x}_t)^\top \boldsymbol{a}_t \leq \nu, \tag{17}$$

  where ν is the sparsity level parameter. The 1st element in the dictionary is chosen at random.
- Once a pass of the ALD criterion is concluded, we can rewrite the problem as

$$J(\tilde{\boldsymbol{\alpha}}) = \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{A}} \tilde{\boldsymbol{K}} \tilde{\boldsymbol{\alpha}}\|^2, \tag{18}$$

  where $\tilde{\boldsymbol{A}} = [\boldsymbol{a}_1 \ \boldsymbol{a}_2 \ \ldots \ \boldsymbol{a}_{m_t}]^\top \in \mathbb{R}^{t \times m_t}$ and $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^{m_t}$.
- Then, computing $\partial J(\tilde{\boldsymbol{\alpha}})/\partial \tilde{\boldsymbol{\alpha}} = \mathbf{0}$, one gets

$$\tilde{\boldsymbol{\alpha}} = \tilde{\boldsymbol{K}}^{-1} \tilde{\boldsymbol{P}} \tilde{\boldsymbol{A}}^\top \tilde{\boldsymbol{y}}, \tag{19}$$

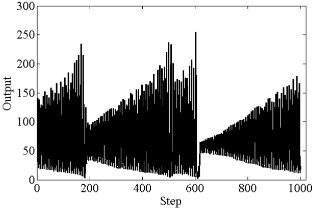
  where $\tilde{\boldsymbol{P}} = (\tilde{\boldsymbol{A}}^\top \tilde{\boldsymbol{A}})^{-1}$.

Outline

Computational Experiments - Laser Time Series

Dataset	Task	Prediction Type	N	N'	$\hat{L}_u$	$\hat{L}_y$
Laser	Time series prediction	Free Simulation	1,000	100/500	-	50

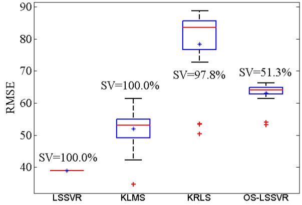
- The chaotic laser time series[4] [5] is a benchmarking dataset with a total of 10,093 samples.
- Scenarios of test: (i) using the next 100 samples;
  (ii) using the next 500 samples.



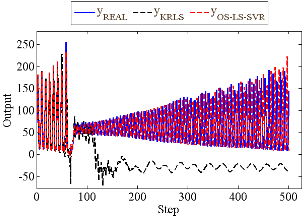
[4] GERSHENFELD, N. A.; WEIGEND, A. S. The future of time series. In: WEIGEND, A. S.; GERSHENFELD, N. A. (Ed.). **Times Series Prediction: Forecasting the Future and Understanding the Past.** Reading, MA: Addison-Wesley, 1993.

[5] Available for download at `www-psych.stanford.edu/$\sim$andreas/Time-Series/SantaFe.html`.

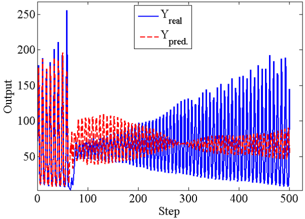
Computational Experiments - Laser Time Series



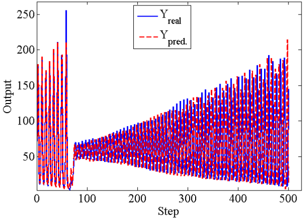
(a) RMSE values.



(b) KRLS and OS-LSSVR.



(c) LSSVR.



(d) OS-LSSVR.

Computational Experiments - Monthly Rainfall

- Average monthly rainfall at the seashore of Fortaleza from 1983 to 2021.
- Training (1983-2016), Validation (2017-2020), Testing (2021).
- Validation for model selection and Testing for actual prediction.
- Validation and Testing in free simulation mode (recursive prediction).
- Initial 8 regressor values taken from 1982.

Computational Experiments - Monthly Rainfall

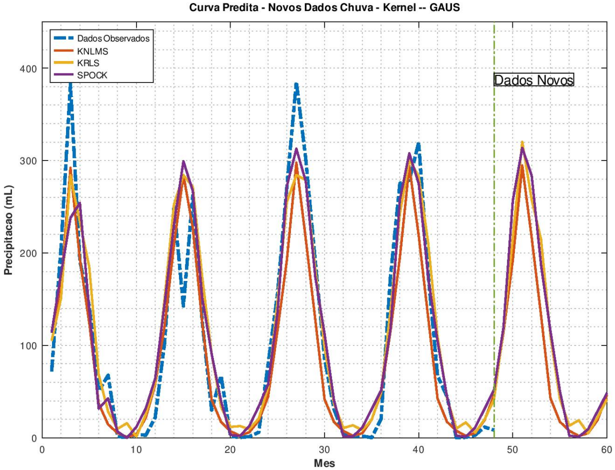
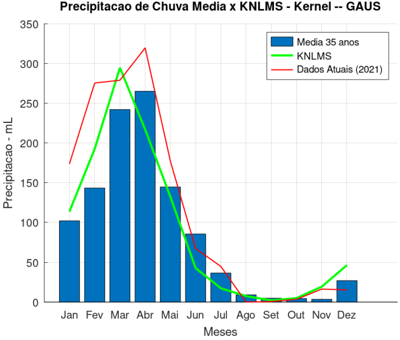
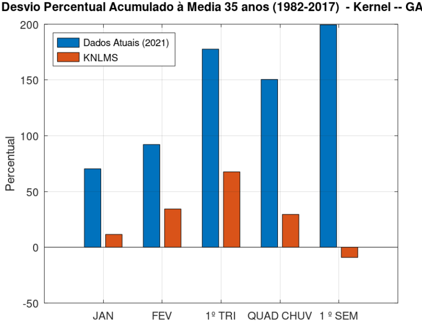


Figure 1: Predicted time series in free simulation mode.

Computational Experiments - Monthly Rainfall



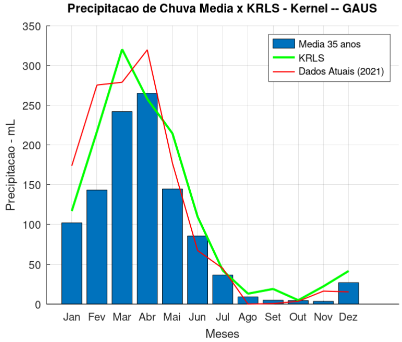
(a) Predicted Rainfall.



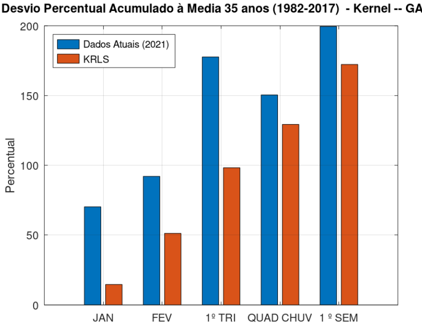
(b) Relative Error.

Figure 2: Results for the KNLMS model.

Computational Experiments - Monthly Rainfall



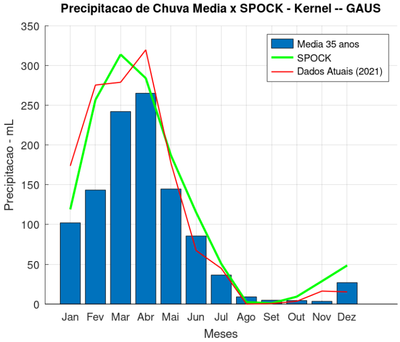
(a) Predicted Rainfall.



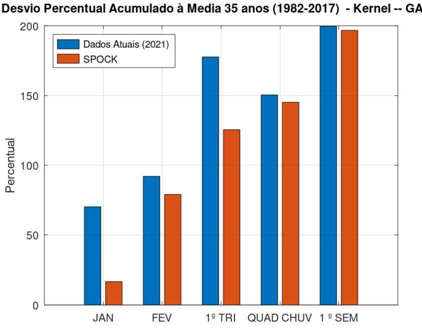
(b) Relative Error.

Figure 3: Results for the KRLS model.

Computational Experiments - Monthly Rainfall



(a) Predicted Rainfall.



(b) Relative Error.

Figure 4: Results for the SPOCK model.

Outline

Conclusions

- We have introduced some kernel-based nonlinear regression models.
- These models were successfully applied to time series forecasting.
- They can be made more compact by means of sparsification techniques, such as the ALD.
- They can be easily extended to online learning.
- They can be extended in order to handle outlier in the data.

Conclusions

- We have introduced some kernel-based nonlinear regression models.
- These models were successfully applied to time series forecasting.
- They can be made more compact by means of sparsification techniques, such as the ALD.
- They can be easily extended to online learning.
- They can be extended in order to handle outlier in the data.

Conclusions

- We have introduced some kernel-based nonlinear regression models.
- These models were successfully applied to time series forecasting.
- They can be made more compact by means of sparsification techniques, such as the ALD.
- They can be easily extended to online learning.
- They can be extended in order to handle outlier in the data.

Conclusions

- We have introduced some kernel-based nonlinear regression models.
- These models were successfully applied to time series forecasting.
- They can be made more compact by means of sparsification techniques, such as the ALD.
- They can be easily extended to online learning.
- They can be extended in order to handle outlier in the data.

Conclusions

- We have introduced some kernel-based nonlinear regression models.
- These models were successfully applied to time series forecasting.
- They can be made more compact by means of sparsification techniques, such as the ALD.
- They can be easily extended to online learning.
- They can be extended in order to handle outlier in the data.

Conclusions

- We have introduced some kernel-based nonlinear regression models.
- These models were successfully applied to time series forecasting.
- They can be made more compact by means of sparsification techniques, such as the ALD.
- They can be easily extended to online learning.
- They can be extended in order to handle outlier in the data.

Acknowledgements

UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARÁ

SPIRAL
Signal and Information
Processing for Data Analysis
and Learning Systems

THANK YOU!!!