Abstract

Regularization networks are nonparametric estimators obtained from the application of Tychonov regularization or Bayes estimation to the hypersurface reconstruction problem. Their main drawback is that the computation of the weights scales as $O(n^3)$ where $n$ is the number of data. In this paper we show that for a class of monodimensional problems, the complexity can be reduced to $O(n)$ by a suitable algorithm based on spectral factorization and Kalman filtering. Moreover, the procedure applies also to smoothing splines.

1 Introduction

The problem of learning a map from a training set containing noisy output measures is subject to the so-called bias/variance dilemma. If the neural network is not complex enough, it will lack the flexibility needed to reproduce the unknown map and the estimate will be biased. On the other hand, if a complex network learns the training set "too well", its generalization capabilities will be deteriorated by the overtraining phenomenon. The usual recipe is to adjust the complexity of the model, e.g. the number of weights and neurons, by statistically-based indices such as the Akaike Information Criterion or crossvalidation techniques. However, without an accurate design of the model structure, if the training data are concentrated in a region of the feature

---

1 Corresponding author.
space, the degrees of freedom of the model will be spent to refine the estimate in that region at the cost degraded performances in other regions. To make an example, in the identification of black-box nonlinear ARX (AutoRegressive eXogenous) models for idle speed control of internal combustion engines [8], enriching the training set with additional data collected in a certain experimental condition may even lead to a global deterioration of the model in all other conditions, unless the network structure is carefully re-designed.

As an alternative to methods that try to adjust the model complexity acting on the number of free parameters, there are nonparametric methods, where the number of parameters is proportional or even equal to number of training examples, and overtraining is prevented by smoothing or regularization. A well known example is given by nonparametric spectral estimators, where smoothing is performed through windowing techniques [18]. Other examples are kernel estimators [21] and smoothing splines [24]. The class of neural networks considered in the present paper, namely RNs (Regularization Networks) [19], belongs to the family of nonparametric estimators as well.

Regularization Networks, whose output is a linear combination of basis functions, have a number of attractive properties. First of all, RNs enjoy nice theoretical interpretations in terms of Tychonov regularization theory [19] and Bayesian estimation [25, 27]. A further attractive property is that there is no trade off between the generalization performance in different regions of the feature space since the number of parameters is equal to the number of examples. At the same time, overtraining is avoided because the weights are not learnt through least squares minimization but by minimizing a quadratic functional that includes an additional terms that penalizes "unlikely" solutions, e.g non-smooth ones. Finally, RNs are consistent estimators in the sense that, under mild assumptions, their estimate converges to the true map as the number of data tends to infinity [5, 6].

The main drawback is that, given the one-to-one correspondence between the number \( n \) of examples and the number of neurons, the training of RNs appears to be computationally expensive (\( O(n^3) \) operations). In order to overcome this "\( n^3 \) curse", simpler suboptimal approximations were proposed such as Generalized RBF’s and Hyper Basis Functions networks [19, 12].

The main purpose of the present paper is to show that in some specific, yet significant case the RN weights can be computed with \( O(n) \) complexity. Our result concerns a fairly large class of RNs for the one-dimensional problem where the function to be reconstructed is a scalar map of a scalar variable. Although this is a limitation, there are significant signal reconstruction problems involving very large data sets which can benefit substantially from the introduction of an \( O(n) \) algorithm, see e.g. the anaesthesiology problem studied in [10], where the data for a single subject involve roughly 50,000/100,000
observations collected at a nonuniform sampling rate. The derivation of the $O(n)$ algorithm is based on the Bayesian interpretation of RNs, and, more precisely on a key technical lemma showing that the covariance matrix of an $n$-dimensional random vector obtained by sampling $n$ times a Markovian stochastic process can be inverted in $O(n)$ operations by Kalman filtering. As a particular case, the algorithm can be applied also to smoothing splines that can be regarded as a particular type of RNs.

2 Preliminaries

Consider the regression problem of estimating an unknown function $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$, from the noisy observations

$$y_i = f(t_i) + \epsilon_i, \quad i = 1, \ldots, n$$  \hspace{1cm} (1)

where the additive measurement errors $\epsilon_i$ are uncorrelated zero-mean Gaussian random variables with $\text{Var}[\epsilon_i] = \sigma^2$. The training set will be denoted by $D_n = \{(t, y)\}$, where $t = [t_1, \ldots, t_n]^T$ and $y = [y_1, \ldots, y_n]^T$.

Regularization networks [12] are a class of theoretically grounded nonparametric estimators for solving such regression problems.

**Definition 1.** A map $C(\xi_1, \xi_2) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is positive-semidefinite if it is symmetric (i.e., $C(\xi_1, \xi_2) = C(\xi_2, \xi_1)$) and satisfies

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j C(t_i, t_j) \geq 0, \forall \alpha_i, \alpha_j \in \mathbb{R}, \forall t_i, t_j \in \mathbb{R}^d, \forall n \in \mathbb{N}_0.$$

**Definition 2.** Given the training set $D_n$, a positive-semidefinite continuous function $C(\xi_1, \xi_2) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (activation function) and a scalar $\gamma > 0$ (regularization parameter), the associated regularization network (RN) is the mapping $f_{\gamma n}(t) : \mathbb{R}^d \to \mathbb{R}$

$$f_{\gamma n}(t) = \sum_{k=1}^{n} \theta_k C(t, t_k)$$ \hspace{1cm} (2)

where the weights $\theta_k$ are obtained as

$$\theta = \begin{bmatrix} \theta_1 & \cdots & \theta_n \end{bmatrix}^T = (H + \gamma I)^{-1} y,$$

$$[H]_{ij} = C(t_i, t_j).$$

**Remark 1.** If $C(\xi_1, \xi_2)$ is radial, i.e., $C(\xi_1, \xi_2) = h(||\xi_1 - \xi_2||)$, then it can be shown [11] that $f_{\gamma n}(\cdot)$ is the solution of the Tychonov-like regularization
\[ f_{\gamma n}(t) = \arg \min_f \sum_{k=1}^{n} (y_k - f(t_k))^2 + \gamma \int_{\mathbb{R}^d} \frac{|\mathcal{F}(f)(\omega)|^2}{\mathcal{F}(h)(\omega)} d\omega, \]

where \( \mathcal{F} \) denotes the \( d \)-dimensional Fourier transform operator. The two terms in the cost functional express two contrasting objectives. The first term is the usual sum of squared errors and penalizes solutions that are not close to the data of the training set. The second term depends on the choice of the kernel \( h(\cdot) \), which is usually a "smooth" function. Then, its Fourier transform is a low-pass filter and the second term penalizes solutions containing high-frequency components. The positive scalar \( \gamma \) is a tuning parameter that controls the balance between data-fit and smoothness of the solution. For instance, if \( \gamma \) is too small the estimate fits the training set very well at the cost of poor generalization performances. Conversely, a too large \( \gamma \) limits the flexibility of the network which will yield a smooth estimate that does not explain the data. Tychonov regularization is a nonparametric estimation technique and the choice of gamma is analogous to the choice of the model complexity in the usual parametric context. The regularization interpretation holds also for RNs whose kernel \( C(\xi_1, \xi_2) \) is not radial, but we omit the details for the sake of simplicity.

**Remark 2.** If \( h(\cdot) \) is radial, it is immediately seen that the RN (2) has the structure of an RBF neural network with \( n \) neurons, whose weights are computed from (3).

A major drawback of RNs is that, in general, the exact calculation of the vector \( \theta \) scales as \( O(n^3) \) due to the computational complexity of the linear system \( (H + \gamma I)\theta = y \). In order to derive an \( O(n) \) algorithm for the calculation of these coefficients, it is first convenient to introduce the Bayesian interpretation of Regularization Networks.

In Bayes estimation a statistical distribution (the so-called prior) for the unknown function \( f(\cdot) \) must be assigned. In particular, we model \( f(\cdot) \) as a zero-mean \( d \)-dimensional Gaussian (stochastic) process independent of the errors \( \epsilon_i \) and with autocovariance function

\[ E[f(\xi_1)f(\xi_2)] = C(\xi_1, \xi_2). \]

The so-called Bayes estimate \( \hat{f}(t) \) is the posterior expectation given the training set \( \mathcal{D}_n \), whose expression is [25]
\[
\hat{f}(t) \triangleq \mathbb{E}[f|D_n](t) = \sum_{k=1}^{n} \theta_k C(t, t_k) 
\]  
(4)

\[
\mathbb{E}[yy^T] = \mathbb{E}[y] = y, \\
\mathbb{E}[yy^T] = (H + \sigma^2 I).
\]  
(5)

It is apparent that \( \hat{f}(t) = f_{\gamma n}(t) \) if \( \gamma = \sigma^2 \).

Note that there exists a one-to one correspondence between positive semidefinite functions and autocovariance functions: every autocovariance function is positive semidefinite and vice-versa, given a positive semidefinite function \( C(\xi_1, \xi_2) \) there always exists a (Gaussian) process whose autocovariance coincides with \( C(\xi_1, \xi_2) \). When \( C(\xi_1, \xi_2) \) has radial symmetry, i.e. \( C(\xi_1, \xi_2) = h(||\xi_1 - \xi_2||) \) we say that the stochastic process (and its autocovariance) is stationary.

Hereafter, we will restrict our attention to monodimensional regression problems \( (d = 1) \). In other words, we are concerned with the problem of reconstructing a function \( f(\cdot) : \mathbb{R} \rightarrow \mathbb{R} \) given \( n \) noisy samples \( y_i \), see (1). In this case, we assume that the “time samples” \( \{t_i\} \) are ordered, i.e. \( t_1 < t_2 < \ldots < t_n \).

The interpretation of the input \( t \) as a time variable opens the way to the use of Kalman filtering techniques, at least when the autocovariance \( C(\xi_1, \xi_2) \) admits a state-space representation. In particular, in the present and the following Section, we will consider stationary autocovariance functions \( h(\cdot) \) satisfying the following assumption.

**Assumption A1** Letting \( H(s) \) be the bilateral Laplace transform of \( h(\cdot) \), assume that

- \( a. H(0) > 0 \)
- \( b. H(s) \) is a rational fraction of the type
  \[
  H(s) = \frac{s^{2m'} N_H(s)}{D_H(s)} = \frac{s^{2m'} a_0 s^{2m} + a_1 s^{2(m-1)} + \ldots + a_m}{s^{2p} + b_1 s^{2(p-1)} + \ldots + b_p}
  \]
  where \( a_0, a_1, \ldots, a_m, b_1, b_2, \ldots, b_p \) are real coefficients and \( m, m', p \) are suitable nonnegative integers satisfying \( m + m' < p, p > 0 \).
- \( c. \) No root of \( D_H(s) \) and \( N_H(s) \) lies on the imaginary axis.

Assumption A1 is not so restrictive as it may appear at first sight even though it rules out autocovariance functions whose Laplace transform \( H(s) \) is not the ratio of polynomials in the \( s^2 \) variable. For instance, the Squared-Exponential (SE) autocovariance function

\[
h_{SE}(||\xi_1 - \xi_2||) = \exp\left(\frac{-||\xi_1 - \xi_2||^2}{2\tau^2}\right), \quad \tau > 0,
\]  
(6)
is not allowed, since \( H_{SE}(s) = \sqrt{2\pi \tau^2} \exp(s^2 \tau^2/2) \). However, in view of Runge’s theorem [20] one may approximate such (holomorphic) \( H(s) \) with arbitrary precision on compact subsets of the complex plane by using rational fractions.

In particular, the SE autocovariance can be approximated by using the so-called Modified Bessel of order \( r \) (\( MB_r \)) functions kernels [23, 26]. They are defined as

\[
h_r : \mathbb{R} \mapsto \mathbb{R},
\]

\[
h_r(\xi_1 - \xi_2) = \kappa_\nu \left( \frac{\left| \xi_1 - \xi_2 \right|}{\tau} \right)^\nu K_\nu \left( \frac{\left| \xi_1 - \xi_2 \right|}{\tau} \right) = \kappa_\nu \sum_{k=0}^{r-1} \beta_k \left( \frac{\left| \xi_1 - \xi_2 \right|}{\tau} \right)^k \exp \left[ -\frac{\left| \xi_1 - \xi_2 \right|}{\tau} \right],
\]

where \( \tau > 0 \) is a scale parameter, \( K_\nu \) is the modified Bessel function of order \( \nu \) (see e.g. Equation 8.468 in [14]) and \( \nu = r - 1/2 \) for integer \( r \). The constant \( \kappa_\nu \) is set such that \( h_r(0) = 1 \). The constants \( \beta_k \) depend on the order \( \nu \) of the Bessel function. For instance, the analytic expression of the Modified Bessel kernels of orders \( r = 1, 2, 3 \) and of their bilateral Laplace Transform is:

\[
h_1(|\xi_1 - \xi_2|) = e^{-\frac{|\xi_1 - \xi_2|}{\tau}}, \quad H_1(s) = \frac{2\tau}{(1 - s^2 \tau^2)}
\]

\[
h_2(|\xi_1 - \xi_2|) = e^{-\frac{|\xi_1 - \xi_2|}{\tau}} \left( 1 + \frac{|\xi_1 - \xi_2|}{\tau} \right), \quad H_2(s) = \frac{4\tau}{(1 - s^2 \tau^2)^2}
\]

\[
h_3(|\xi_1 - \xi_2|) = e^{-\frac{|\xi_1 - \xi_2|}{\tau}} \left( 1 + \frac{|\xi_1 - \xi_2|}{\tau} + \frac{1}{3} \left( \frac{|\xi_1 - \xi_2|}{\tau} \right)^2 \right), \quad H_3(s) = \frac{16 \tau}{(1 - s^2 \tau^2)^3}.
\]

In [17] it is shown that \( MB_r \) kernels are positive semidefinite. To see how \( MB_r \) functions approximate the \( SE \) one [26], we consider the bilateral Laplace Transform of \( MB_r \)

\[
H_r(s) \propto \frac{\tau}{(1 - s^2 \tau^2)^{r}}, \quad (7)
\]

Since,

\[
\lim_{r \to \infty} \left( 1 - \frac{s^2 \tau^2}{2r} \right)^{-r} = \exp \left[ \frac{s^2 \tau^2}{2} \right],
\]

the \( MB_r \) behaves like the Gaussian kernel, for large \( r \), provided that \( \tau \) is suitably rescaled as a function of \( r \). Note also that, from (7), \( MB_r \) functions satisfy Assumption A1.
As it will be shown below, the advantage of using autocovariance functions satisfying Assumption A1 is that the corresponding Gaussian process \( f(\cdot) \) can be represented as the output \( \tilde{f}(t) \) of a linear time-invariant system admitting a state-space representation of the type:

\[
T: \begin{cases}
\dot{x}(t) = Ax(t) + Bw(t) \\
\tilde{f}(t) = Cx(t) \\
x(t_0) = x_0
\end{cases}, \tag{8}
\]

where \( A \in \mathbb{R}^{p \times p}, B \in \mathbb{R}^{p \times 1}, C \in \mathbb{R}^{1 \times p} \) are suitable matrices, \( w(\cdot) \) is a continuous-time white Gaussian noise, and the initial state \( x_0 \) is a suitable random vector. The next three Lemmas show how to obtain the state-space representation (8) from the spectrum \( H(s) \) as well as how to derive the corresponding discrete time model at the sampling points. The reader who is not familiar with stochastic system theory can find the proofs in the Appendix.

**Lemma 1.** Let Assumption A1 holds. Then, there exists a complex function

\[
G(s) = \frac{N_G(s)}{D_G(s)} = \frac{\tilde{a}_1 s^{p-1} + \tilde{a}_2 s^{p-2} + \ldots + \tilde{a}_p}{s^p + \tilde{b}_1 s^{p-1} + \ldots + \tilde{b}_p}
\]

where \( \tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_p, \tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_p \) are real coefficients, such that \( H(j\omega) = |G(j\omega)|^2 \) and \( D_G(s) \) is stable (i.e. its roots belong to the open left-half plane).

**Lemma 2.** Let \( f(\cdot) \) be a zero-mean stationary Gaussian process whose autocovariance function \( h(\cdot) \) satisfies Assumption A1. Consider the system \( T \) defined by (8) with

\[
A = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
-\tilde{b}_p & -\tilde{b}_{p-1} & -\tilde{b}_{p-2} & \cdots & -\tilde{b}_1
\end{bmatrix}, \quad B = \begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
\tilde{a}_p & \tilde{a}_{p-1} & \cdots & \tilde{a}_2 & \tilde{a}_1
\end{bmatrix}.
\]

where the coefficients \( \tilde{a}_i \) and \( \tilde{b}_i \) are those of Lemma 1. Let \( w(t) \) be a continuous-time white Gaussian noise with unit intensity and \( x_0 \sim N(0, P) \), where \( P \) is
the unique positive-definite solution of the Lyapunov equation

\[ AP + PA^T + BB^T = 0. \quad (9) \]

Then \( \tilde{f}(t) \) has the same statistics as \( f(t) \), \( \forall t \geq t_0 \), i.e. \( (8) \) is a stochastic state-space realization of \( f(\cdot) \).

To make an example, consider the \( MB_1 \) autocovariance function. From Lemma 2 we obtain \( A = -1/\tau, B = 1, C = \sqrt{2}/\tau \). Moreover the Lyapunov equation \( (9) \) gives \( P = \frac{\tau}{2} \). Similarly, for the \( MB_2 \) function we obtain

\[
A = \begin{bmatrix}
0 & 1 \\
\frac{-1}{\tau^2} & -\frac{2}{\tau}
\end{bmatrix}, \quad B = \begin{bmatrix}
0 \\
1
\end{bmatrix}, \quad C = \begin{bmatrix}
\sqrt{\frac{4}{\tau^3}} & 0
\end{bmatrix}, \quad P = \begin{bmatrix}
\frac{\tau^3}{4} & 0 \\
0 & \frac{\tau}{4}
\end{bmatrix}.
\]

The next step is to obtain a state-space stochastic realization of the discrete-time process \( y_k \).

**Lemma 3.** Let \( y_k \) be given by \( (1) \), where \( f(\cdot) \) is a zero-mean stationary Gaussian process whose autocovariance function \( h(\cdot) \) satisfies Assumption A1. Consider the discrete-time time-varying system

\[
W : \begin{cases}
x_{k+1} = F_k x_k + G_k w_k \\
\tilde{y}_k = C x_k + \epsilon_k \\
x_0 \sim N(0, P)
\end{cases} \quad (10)
\]

where \( w_k \sim WGN(1) \) (i.e. \( w_k \) is a discrete-time white Gaussian noise with unit variance), \( \epsilon_k \) are uncorrelated zero-mean Gaussian random variables with \( \text{Var}[\epsilon_k] = \sigma^2 \), and \( C \) and \( P \) are defined as in Lemma 2, \( F_k = e^{A(t_{k+1} - t_k)} \), and \( G_k \in \mathbb{R}^{p \times 1} \) is such that

\[
G_k G_k^T = \int_{t_k}^{t_{k+1}} e^{A(t_{k+1} - t)} BB^T e^{A^T(t_{k+1} - t)} dt.
\]

Moreover, assume that \( w_k, \epsilon_k, x_0 \) are mutually independent. Then, the samples \( \tilde{y}_k \) and \( y_k, k = 1, 2, \ldots, n \), have the same statistics, i.e. \( (10) \) is a stochastic state-space realization of the discrete-time process \( y_k \).

Now, we can take advantage of the state-space model \( (10) \) in order to derive an \( O(n) \) algorithm based on Kalman filtering for the computation of the coefficients \( \theta_i \). For this purpose, a basic property of the Kalman filter is first recalled [1].

**Lemma 4.** The system
\[ \begin{align*}
\mathbf{K} : & \quad \begin{cases}
\hat{x}_{k+1} = (F_k - K_k C) \hat{x}_k + K_k \tilde{y}_k \\
\nu_k = \left(R_k^e\right)^{-\frac{1}{2}} (\tilde{y}_k - C \hat{x}_k) \\
\hat{x}_1 = 0
\end{cases}, \\
R_k^e & = C \Sigma_k C^T + \sigma^2 \\
K_k & = F_k \Sigma_k C^T \left[C \Sigma_k C^T + \sigma^2\right]^{-1} \\
\Sigma_{k+1} & = F_k \Sigma_k F_k^T - F_k \Sigma_k C^T \left[C \Sigma_k C^T + \sigma^2\right]^{-1} C \Sigma_k F_k^T + G_k G_k^T \\
\Sigma_1 & = F_0 P F_0^T + G_0 G_0^T,
\end{align*} \]

is a "whitening filter" for the sequence \( \tilde{y}_k, k = 1, 2, \ldots, n \), defined in Lemma 3, in the sense that its output \( \nu_k \) is a white Gaussian noise.

**Remark 3.** The expression of the initial covariance matrix \( \Sigma_1 \) seems to depend on the initial time \( t_0 \) which has no correspondence in the training set \( D_n \). However, by exploiting the Lyapunov equation (9), it can be seen that \( \Sigma_1 = P \), so that the dependence on \( t_0 \) vanishes. We did not introduce this simplification in Lemma 4 because the present statement has a more general validity. In fact, it holds also for nonstationary covariance functions, (see Section 4), in which case \( P \) may not be the solution of an algebraic Lyapunov equation.

We are now in a position to state the main result.

**Theorem 1.** The weights \( \theta_i, i = 1, \ldots, n \), in (4), (5), can be computed by means of the following algorithm:

\[ \begin{align*}
\hat{x}_{k+1} & = (F_k - K_k C) \hat{x}_k + K_k \tilde{y}_k, \quad x_1 = 0 \quad (11) \\
\nu_k & = \left(R_k^e\right)^{-\frac{1}{2}} (\tilde{y}_k - C \hat{x}_k), \quad k = 1, 2, \ldots, n \quad (12) \\
\hat{\xi}_{j-1} & = (F_j - K_j C)^T \hat{\xi}_j - C^T \left(R_j^e\right)^{-\frac{1}{2}} \nu_j, \quad \hat{\xi}_n = 0 \quad (13) \\
\theta_j & = K_j^T \hat{\xi}_j + \left(R_j^e\right)^{-\frac{1}{2}} \nu_j, \quad j = n, n-1, \ldots, 1. \quad (14)
\end{align*} \]

where the matrices \( K_k \) and \( R_k^e \) are defined as in Lemma 4.

**Proof.** See Appendix. \( \square \)

Summarizing the results of this section, the proposed algorithm is:
Algorithm 1.

1. Compute the bilateral Laplace transform
   \[ H(s) = \mathcal{L}(h)(s); \]
2. Use the constructive procedure outlined in the proof of Lemma 3 to compute the polynomials \( N_G(s), D_G(s); \)
3. Construct matrices \( A, B, C, \) according to Lemma 2;
4. Compute the \( F_k \) matrices as \( F_k = e^{A(t_{k+1}-t_k)}, k = 1, 2, \ldots, n; \)
5. Compute the \( P \) matrix by solving equation (9) (this requires \( O(p^3) \) operations [15]);
6. Compute the matrices
   \[ Q_k = G_k G_k^T = \int_{0}^{t_{k+1}-t_k} e^{A\tau} B B^T e^{A^T \tau} d\tau, k = 1, 2, \ldots, n; \]
   This may be done by the means of the following procedure [22]
   \[ \tilde{F} = \begin{bmatrix} -A & BB^T \\ 0 & A \end{bmatrix}, \quad \begin{bmatrix} \Phi_1 & \Psi \\ 0 & \Phi_2 \end{bmatrix} e^{F(t_{k+1}-t_k)}, \quad \Phi_1 \in \mathbb{R}^{p \times p}, \]
   \[ Q_k = \Phi_2^T \Psi; \]
7. Compute the \( \theta_k \) coefficients using the algorithm of Theorem 1.

The computations in steps 1,2,3 and 5 do not depend on the training set but only on the adopted autocovariance function \( h(t). \) This means that these calculations can be done off-line once for all and the results stored for future use. The computational cost of steps 4 and 6 scales as \( O(p^3 n) \), where the factor \( p^3 \) arises from the calculation of the exponential matrices [13]. However, we stress that \( p \) is fixed (and usually \( p \ll n \)) so that the complexity of these steps scales in a linear way with the size of the training set. Moreover, it is easily seen by inspection that the Kalman filter calculations involved in Theorem 1 take \( O(p^3 n) \) operations [1]. Hence, the overall complexity of the proposed algorithm is linear with respect to the number \( n \) of the collected datapoints.
A careful examination of Lemmas 3 and 4 and Theorem 1 reveals that their proofs are independent of the structure of the matrices \(A, B, C\). In fact, their statements still hold true provided that the Bayesian prior is assigned in state-space (i.e. Markovian) form: more precisely one has to assume that \(f(t) = \tilde{f}(t)\) with \(\tilde{f}(t)\) defined by (8) where \(w(t)\) is a Gaussian white noise with unitary intensity and \(x_0 \sim N(0, P)\), where \(P \geq 0\) is an arbitrary matrix (it need not satisfy the Lyapunov equation (9)). Note that, when \(P\) does not satisfy (9), the process \(\tilde{f}(t)\), is no more stationary. An interesting case is given by the following choice of the system \(T\):

\[
A = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix},
\]

where \(A \in \mathbb{R}^{p \times p}\). If \(P = 0\), it is easy to recognize that \(\tilde{f}(t)\) is a \((p - 1)\) fold integrated Wiener process and the associated Bayes estimate turns out to be a \(p\)-th order regression spline with zero boundary conditions [24]. Then, steps 4, 6 and 7 of the algorithm presented in the previous section provide an \(O(n)\) procedure for the computation of the spline weights.

Unfortunately, it is not possible to extend the \(O(n)\) algorithm to the general multidimensional case \((d > 1)\). However, as explained in [7], it is possible to take advantage of the proposed algorithm when computing the weights of a particular class of multidimensional RNs, namely additive RNs [16].

5 Examples

We considered the problem of reconstructing the function

\[
f(t) = \frac{1}{2} \exp \left( -2t + 20t^3 - 43t^5 + 26t^7 \right) + \sin \left( 1 - 7t - 7t^2 + 21t^3 \right),
\]

on the interval \([-1, 1]\). This function exhibits both low-frequency and oscillating components. To generate the training set, 100 samples \(t_i\) from the uniform distribution on \([-1, 1]\) are drawn. The data \(y_i\) are created by adding Gaussian noise \(\epsilon_i\) with \(\text{Var}[\epsilon_i] = 0.01\). The function and the training set are depicted in figure 1-a. To illustrate Algorithm 1, we fixed \(\gamma = 1\) and we used the \(MB_2\) au-
The original function and the training set $D_{100}$.

(b) The estimated function.

Figure 1. Regression with $MB_{2}$ kernels.

tocovariance function with $\tau = 0.2$. The RN obtained for $\text{Var}[\epsilon_i] = \sigma^2 = 0.01$ yields the estimate drawn in figure 1-b. As discussed later, even better results can be obtained if we tune $\gamma$ in a proper way.

The computational benefits of Algorithm 1 with respect to the “standard method” (i.e. solving the linear system (5)), are illustrated in figure 2. All the experiments were performed on a Pentium II - 400 Mhz machine with 128 Mb of RAM using Matlab 5.3. In figure 2-a we compare the computational complexity of the two methods. The number of flops was calculated by taking into account the operations required for generating the matrices that depend on the dataset and for calculating the regression coefficients. Therefore, for the standard regression algorithm, we considered the computation of $(H + \sigma^2 I)$ and the solution of the linear system (5). For Algorithm 1 we considered steps 4, 6 and 7. From the log-log plot of figure 2-a it is apparent that the computational burden for the standard method grows as $O(n^3)$.
Figure 3. Comparison between regression with MB\textsubscript{2} and Gaussian autocovariances whereas the burden associated with Algorithm 1 grows as \(O(n)\). Algorithm 1 is advantageous also for memory occupation. In figure 2-b it is shown that the memory occupation scales linearly for Algorithm 1 and quadratically for the standard method that requires the full storage of the matrix \((H + \sigma^2 I)\). A comparison between the computational times of the regression algorithms for various sizes of the training set is reported in table 1. When using Algorithm 1 with a training set consisting of 50000 samples, the computational time was 41.7 min. (85.398.457 flops).

<table>
<thead>
<tr>
<th>N. of samples</th>
<th>100</th>
<th>600</th>
<th>1100</th>
<th>1600</th>
<th>2100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 1</td>
<td>0.22s</td>
<td>1.32s</td>
<td>2.53s</td>
<td>4.12s</td>
<td>6.1s</td>
</tr>
<tr>
<td>Standard method</td>
<td>0.05s</td>
<td>1.92s</td>
<td>9.77s</td>
<td>29.22s</td>
<td>119.57s</td>
</tr>
</tbody>
</table>

Table 1
Comparison of the computational times.

As already mentioned, our Algorithm does not apply to the Gaussian kernel which does not satisfy Assumption A1. However, the Gaussian kernel can be approximated by MB\textsubscript{r} autocovariances. In figure 3 we compared the RNs obtained by using the MB\textsubscript{2} (i.e. and MB kernel of low order) and Gaussian autocovariance functions

\[
\hat{h}_2(\xi_1 - \xi_2) = \alpha \exp \left( -\frac{|\xi_1 - \xi_2|}{0.2} \right) \left( 1 + \frac{|\xi_1 - \xi_2|}{0.2} \right),
\]

\[
\hat{h}_{SE}(|\xi_1 - \xi_2|) = \beta \exp \left( -\frac{|\xi_1 - \xi_2|^2}{2 \cdot 0.2^2} \right).
\]

We used the same training set as in figure 1-a and we tuned \(\sigma^2\) and the scale parameters \(\alpha > 0\) and \(\beta > 0\) by using Generalized Cross Validation [24]. This is a popular data-based algorithm that estimates the “best” values for the parameters. Figures 3-a and 3-b highlight that MB\textsubscript{2} kernel reconstructs more
effectively the oscillating part, whereas the Gaussian RN reproduces better the low-frequency components. This is not surprising, since, roughly speaking, the use of Gaussian kernels amounts to assuming a very high degree of smoothness of the unknown function [26, 9]. Conversely, the $MB_2$ kernel is less smooth and therefore more flexible with respect to fast oscillations. In any case, in order to obtain a smoother estimate it is sufficient to adopt an $MB_r$ kernel with a large enough value of $r$.

6 Concluding remarks

It has been shown that for a fairly large class of monodimensional Regularization Networks the weights can be computed in $O(n)$ operations via a Kalman filtering algorithm. Although not included in the considered class, the Gaussian activation function can be approximated arbitrarily well by resorting to the family of $MB_r$ kernels. As illustrated by numerical examples, the new numerical scheme brings a definite advantage in terms of computational time and memory occupation, thus opening the way to the use of RNs for on-line processing of large data sets.

It is worth noting that the so-called ARMA splines can be computed with $O(n)$ complexity by applying Kalman filtering to a suitable state-space model [24, 2] (ARMA splines are roughly equivalent to the RNs considered in the present paper). However these $O(n)$ algorithms are oriented towards the pointwise computation of the regression function over a prespecified grid rather than providing directly the weights of the RN. Hence, they lose the network structure of the estimator together with its possible advantages in terms of parallel implementation.

Acknowledgments

This paper has been partially supported by MURST Project “Identification and Control of Industrial Systems”, ICE-CNR and NIH grant RR11095.

Appendix

Proof of Lemma 1 First, note that the roots of $N_H(s)$ and $D_H(s)$ are symmetric w.r.t. the origin, and, in view of Assumption A1(c), they do not lie on the imaginary axis. Let $z_1, z_2, \ldots, z_m$ ($\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_p$) be the roots of $N_G(s)$,
(\(D_G(s)\)) belonging to the left half plane. It is easy to verify that
\[
G(s) = \sqrt{H(0)} \left( s - z_1 \right) \left( s - z_2 \right) \cdots \left( s - z_m \right) \left( s - \hat{z}_1 \right) \left( s - \hat{z}_2 \right) \cdots \left( s - \hat{z}_p \right),
\]
satisfies \(H(j\omega) = |G(j\omega)|^2\). The fact that \(\tilde{a}_i, \tilde{b}_i, i = 1, 2, \ldots p\) are real follows from the symmetry of the roots of \(N_H(s)\) and \(D_H(s)\) w.r.t. the real axis.

**Proof of Lemma 2** Letting \(x_0 = 0\), a standard result of linear system theory [4] shows that the system \(\mathbf{T}\) is a (deterministic) realization of \(G(s)\). This means that, given a generic (deterministic) input \(w(t)\) s.t. \(w(t) = 0, t < t_0\), it holds
\[
\mathcal{L}(\tilde{f})(s) = G(s)\mathcal{L}(w)(s),
\]
where \(\mathcal{L}\) denotes the bilateral Laplace transform operator. Moreover, recalling that \(\mathbf{T}\) is stable, it can be shown [3] that, if \(x_0 \sim N(0, P)\), the stochastic process \(\tilde{f}(t), t \geq t_0\) defined by (8) is zero-mean, Gaussian and stationary with
\[
E\left[ \tilde{f}(t)\tilde{f}(t + \tau) \right] = \tilde{h}(\tau)
\]
where \(\tilde{h}(\tau)\) is such that
\[
\mathcal{L}(\tilde{h})(j\omega) = |G(j\omega)|^2.
\]
Finally, from Lemma 1, \(\mathcal{L}(h)(j\omega) = |G(j\omega)|^2\). Hence, \(\mathcal{L}(h)(s) = \mathcal{L}(\tilde{h})(s), \forall s \in \mathbb{C}\) implying \(h(\tau) = \tilde{h}(\tau)\) so that \(f(\cdot)\) and \(\tilde{f}(\cdot)\) have the same autocovariance. Since \(f(\cdot)\) and \(\tilde{f}(\cdot)\) are zero-mean and Gaussian, the thesis follows.

**Proof of Lemma 3** With reference to system (8), the Lagrange formula yields
\[
x(t_{k+1}) = F_k x(t_k) + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-t)} B w(t) dt, \quad k = 0, 1, \ldots, n\tag{15}
\]
Hence,
\[
\text{Cov} \left[ x(t_{k+1}) \right] = F_k \text{Cov} \left[ x(t_k) \right] F_k^T + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-t)} B B^T e^{A^T(t_{k+1}-t)} dt,
\]
with \(\text{Cov} \left[ x(t_0) \right] = P\). On the other hand, from (10)
\[
\text{Cov} \left[ x_{k+1} \right] = F_k \text{Cov} \left[ x_k \right] F_k^T + G_k G_k^T, \quad \text{Cov} \left[ x_0 \right] = P,
\]
so that \(x(t_k)\) and \(x_k\) have the same covariance. In a similar way, resorting again to (15), it is possible to show that the whole autocovariance functions of \(x(t_k)\) and \(x_k\) coincide, see e.g. [3]. Then, the thesis easily follows by inspecting the definitions of \(y_k\) and \(\tilde{y}_k\). \(\Box\)
The following notion of input-output matrix and the subsequent Lemma are instrumental to the proof of Theorem 1.

**Definition 3.** Let $L_S \in \mathbb{R}^{n \times n}$ be a lower triangular matrix and consider the single-input ($u_k \in \mathbb{R}$) single-output ($y_k \in \mathbb{R}$) linear time-varying system $S$:

$$
S : \begin{cases} 
  x_{k+1} = A_k x_k + B_k u_k \\
  y_k = C_k x_k + D_k u_k, \quad x_k \in \mathbb{R}^p.
\end{cases}
$$

Then, $S = (A_k, B_k, C_k, D_k)$ is said a realization of $L_S$ if,

$$
\begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}^T = L_S \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix}^T, \tag{16}
$$

for all the sequences $u_k$. Conversely, given $S = (A_k, B_k, C_k, D_k)$, the associated matrix $L_S$ such that (16) holds is termed $(n$-step) input-output matrix of the system.

Note that the input-output matrix of a system $S$ is well defined since, due to linearity, it does not depend on the input sequence.

**Lemma 5.** Let $L_S \in \mathbb{R}^{n \times n}$ be the $(n$-step) input-output matrix of the single input-single output system $S = (A_k, B_k, C_k, D_k)$. Then the system

$$
S^T : \begin{cases} 
  \xi_{k-1} = A_k^T \xi_k + C_k^T u_k \\
  \eta_k = B_k^T \xi_k + D_k u_k, \quad \xi_k \in \mathbb{R}^p \\
  \xi_n = 0
\end{cases}
$$

is a realization of $L_S^T$ in the sense that

$$
\begin{bmatrix} \eta_1 & \eta_2 & \cdots & \eta_n \end{bmatrix}^T = L_S^T \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix}^T.
$$

The proof of Lemma 5 is easily carried out by inspection.

**Proof of Theorem 1.** In view of Lemma 4, the system (11)-(12) is the whitening filter of the sequence $\tilde{y}_k$ and, by Lemma 3, of $y_k$ as well. Then, letting $\nu = \begin{bmatrix} \nu_1 & \nu_2 & \cdots & \nu_n \end{bmatrix}^T$ it follows that $E[\nu \nu^T] = I$. Now let $L_K$ be the input-output matrix of the system $K$ given in Lemma 4. Then $\nu = L_K y$ and

$$
I_n = E[\nu \nu^T] = E[L_K y y^T L_K^T] = L_K E[y y^T] L_K^T. \tag{17}
$$
Recalling that the system $\mathcal{K}$ admits a causal inverse [1], it follows that $L_{\mathcal{K}}$ is nonsingular. Then, (17) implies

$$E[yy^T]^{-1} = L_{\mathcal{K}}^T L_{\mathcal{K}}.$$  \hfill (18)

Since Lemma 5 shows that the system (13)-(14) is the realization of $L_{\mathcal{K}}^T$, it follows that the input-output matrix of the overall system (11)-(14) having $y_k$ as input and $\theta_k$ as output is $L_{\mathcal{K}}^T L_{\mathcal{K}}$. Hence it follows that

$$\theta = L_{\mathcal{K}}^T L_{\mathcal{K}} y.$$  

This, in view of (18) and (5), concludes the proof.

References


