Particle Filter and Support Vector Regression: Individual and collective performance

First Author[1][0000−1111−2222−3333], Second Author[2,3][1111−2222−3333−4444], and Third Author[3][2222−−3333−4444−5555]

[1] Princeton University, Princeton NJ 08544, USA
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
lncs@springer.com
http://www.springer.com/gp/computer-science/lncs
[3] ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
{abc,lncs}@uni-heidelberg.de

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

Los algoritmos Sequential Monte Carlo (SMC) son técnicas utilizados para simular muestras en forma secuencial a partir de distribuciones que evolucionan en el tiempo, se caracterizan por ser altamente flexibles y de extensa aplicabilidad en muchos campos de la ciencia. En particular, los filtros de partículas (PF) es un algoritmo SMC. Los PF fueron desarrollados en la década de los (1990), por Gordon et. al (1993), Kitagawa (1996), Del Moral (1996), Doucet et. al (2000), Doucet et. al (2001), Godsill et. al (2000), Ristic et. al (2003), entre otros, con el propósito de aproximar distribuciones arbitrarias, posiblemente multimodales y en espacios de alta dimensiones. El método consiste en generar un conjunto de muestras pesadas llamadas partículas y una distribución prior para aproximar la densidad a posterior de los estados desconocidos. El paso de predicción supone una recursión Markoviana que se actualiza en forma iterativa sobre la base de los estados predichos en tiempos pasados.

2 Related Work

SABA

$$
\begin{aligned}
dX_t &= (\alpha X_t - \beta X_t Y_t)dt + \sqrt{\sigma_1 X_t} dW_t \\
dY_t &= (-\gamma Y_t + \tau X_t Y_t - \nu Y_t^2)dt + \sqrt{\sigma_2 X_t} dB_t
\end{aligned}
$$

Function	Technique	RMSE	SMSE	MSLL
$x(t)$	GP	$6.96e^{-10}$	$3.67e^{-20}$	$1.63e^{-21}$
$y(t)$	GP	$4.01e^{-10}$	$2.39e^{-20}$	$3.17e^{-21}$

Parameters

$$\begin{aligned}\alpha &= 1.5\\ \beta &= 0.4\\ \sigma_1 &= 0.05\\ \gamma &= 1.5\\ \tau &= 0.4\\ \nu &= 0.2\\ \sigma_2 &= 0.1\end{aligned}$$

3 Materials and Methods

3.1 SARIMA

Time series forecasting is an important research area in machine learning, due to the consideration of information from the past to predict the future. Proper forecasting has significant value in many areas of science such as: economics, meteorology, agriculture, biology, physics and health among many others. When the time series is not stationary, it can be decomposed into deterministic and stochastic components to obtain better prediction performance. The characteristics of the trend and seasonality are treated by the autoregressive integrated moving average model (ARIMA(p, d, q)) where p is the number of auto regressive terms, d is the order of differencing, q is the number of moving average terms. ARIMA is one of the most widely used forecast models due to its flexibility. However, it has some limitations due to the assumption of linearity between the present value and the past value, and random noise in the model; the assumption of additive tendency is considered but it does not take into account the seasonal episodes, for more details see K Abdul Hamid, et., al. (2023), Aisyah et., al. (2021), Dama and Sinoquet (2021) y sus referencias entre otros.
A time series $\{X_t\}$ follows an ARIMA(p, d, q) process if:

$$\Phi_p(L)\,(1-L)^d\,X_t = \Theta_q(L)\epsilon_t$$

where $\{\epsilon_t\}$ is a white noise series, p, d, q are integers, L is the backward shift operator $LX_t = X_{t-1}$, $L^kX_t = X_{t-k}$, Φ_p and Θ_q are polynomials in L, of orders p and q, respectively:

$$\begin{aligned}\Phi_p &= 1 - \phi_1 L - \phi_2 L^2 - \ldots - \phi_p L^p\\ \Theta_q(L) &= 1 - \alpha_1 L - \alpha_2 L^2 - \ldots - \alpha_p L^p\end{aligned}$$

The best known structures of the ARIMA model are summarized in see Dama and Sinoquet (2021):

1. White noise: ARIMA$(0, 0, 0)$,
2. Random walk process: ARIMA$(0, 1, 0)$,
3. Autoregression: ARIMA$(p, 0, 0)$,
4. Moving Avarage: ARIMA$(0, q, 0)$,
5. ARMA: ARIMA$(p, 0, q)$.

Seasonal Autoregressive Integrated Moving Average Model ($SARIMA$) is similar to the Arima model but this model is preferable when the time series exhibits seasonality. The SARIMA can be expressed in terms of a composite model which can be denoted as $(SARIMA(p; d; q)(P; D; Q)_s)$, where p, d and q represent the non-seasonal AR order, no seasonal differencing, non-seasonal MA order respectively. Also, the model parameters P, D, Q and s are corresponding to the seasonal AR order, seasonal differencing, seasonal MA order, and time span of repeating seasonal pattern respectively. The SARIMA is not a linear model. A time series $\{X_t\}$ is generated by a SARIMA process if:

$$\Phi_p(L)\Phi_P\left(L^s\right)(1-L)^d\left(1-L^s\right)^D X_t = \Theta_q(L)\Theta_Q(L^s)\epsilon_t$$

where $\{\epsilon_t\}$ is a white noise series, p, d, q, P, D, Q and s are integers, L is the backward shift operator $L^k X_t = X_{t-k}$, and

$$\begin{aligned}
\Phi_p(L) &= 1 - \phi_1 L - \phi_2 L^2 - \ldots - \phi_p L^p \\
\Phi_p(L^s) &= 1 - \phi_s L - \phi_{2s} L^2 - \ldots - \phi_{Ps} L^{Ps} \\
\Theta_q(L) &= 1 - \alpha_1 L - \alpha_2 L^2 - \ldots - \alpha_p L^p \\
\Theta_Q(L^s) &= 1 - \alpha_s L^s - \alpha_{2s} L^{2s} - \ldots - \alpha_{Qs} L^{Qs}
\end{aligned}$$

are polynomials of degrees p, P, q and Q, respectively, s is the seasonality period, d is the number of classical differentiations and D is the number of seasonal differentiations.
The best known structures of the SARIMA model are summarized in see Dama and Sinoquet (2021):

1. Seasonal ARMA: SARIMA$(0, 0, 0)(P, 0, Q)_s$,
2. ARIMA: SARIMA$(p, d, q)(0, 0, 0)$,
3. Additive trend-seasonality model: SARIMA$(p, d, q)(0, D, 0)_s$.

3.2 Support Vector Regression

The Support Vector Regression (SVR) algorithm was formulated by Vapnik and Chervonenkis (1964) and is a machine-learning algorithm that is perhaps the most elegant of all kernel-learning methods. The SVR consist of a small subset of data points extracted by the learning algorithm from the training sample itself.

SVR models have recently been used to handle problems such as nonlinear, local minimum, and high dimension. This model can even guarantee higher accuracy for long-term predictions compared to other computational approaches in many applications.
Suppose the training data given are $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X}$ where x_i is the input pattern for the $i-$th example $x_i \in \mathcal{X} = \mathbb{R}^n$, and $y_i = f(x_i) \in \mathbb{R}$ is the corresponding desired response (target output), Haykin (2009), Muthukrishnan and Maryam (2020), K Abdul Hamid et. al (2023). The function$f : \mathcal{X} \to \mathbb{R}$ can be described as

$$x \to f(x) := \langle w, x \rangle + b = w^T x + b, \quad w \in \mathcal{X}, \quad b \in \mathbb{R} \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathcal{X}$, x is an input vector, w is an adjustable weight vector, and b is a bias.
Now we define the optimization problem. Given the training sample $\mathcal{D}$, we want find the optimum values of the weight vector w and bias b such that they satisfy the constraints

$$y_i \left(w^T x_i + b \right) \geq 1, \quad i = 1, \ldots, n \tag{2}$$

and the weight vector w minimizes the cost function

$$\min \frac{1}{2} \|w\|^2 = \min \frac{1}{2} w^T w \tag{3}$$

We construct the Lagrangian function, Bertsekas, (1995)

$$J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i \left[y_i \left(w^T x_i + b \right) - 1 \right] \tag{4}$$

where the auxiliary nonnegative variables α_i are called Lagrange multipliers. The solution to the constrained-optimization problem is determined by the saddle point of the Lagrangian function $J(w, b, \alpha)$, i.e

$$\frac{\partial J(w, b, \alpha)}{\partial w} = 0, \quad \frac{\partial J(w, b, \alpha)}{\partial b} = 0 \tag{5}$$

Application (4) and (5) yields the following:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i, \quad \textit{and} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{6}$$

The solution vector w is defined in terms of an expansion that involves the n training examples.
The SVR can be viewed as a kernel machine inner product kernel. Let x denote a vector drawn from the input space of dimension m. Let $\{\varphi_i(x)\}_{i=1}^{\infty}$ denote a set of nonlinear functions that, between them, transform the input space of dimension m to a feature space of infinite dimensionality. Given this transformation, we

may define a hyperplane acting as the decision surface in accordance with the formula

$$\sum_{i=1}^{\infty} w_i \varphi_i(x) = 0$$

where $\{w_i\}_{i=1}^{\infty}$ denotes an infinitely large set of weights that transforms the feature space to the output space. Using matrix notation:

$$w^T \Phi(x) = 0 \tag{7}$$

where $\Phi(x)$ is the feature vector and w is the corresponding weight vector. Now we may expressing the weight vector as

$$w = \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i) \tag{8}$$

where the feature vector is expressed as

$$\Phi(x_i) = (\varphi_1(x_i), \varphi_2(x_i), \ldots)^T \tag{9}$$

Then we may express the decision surface in the output space as

$$\sum_{i=1}^{n} \alpha_i y_i \Phi(x_i)^T \Phi(x) = 0$$

where n is the number of support vectors and $\Phi(x_i)^T \Phi(x)$ represents an inner product, where this inner-product term be denoted as

$$k\left(x, x_i\right) = \Phi(x_i)^T \Phi(x) = \sum_{j=1}^{\infty} \varphi_j(x_i) \varphi_j(x) \tag{10}$$

The optimal decision surface in the output space as

$$\sum_{i=1}^{n} \alpha_i y_i k\left(x, x_i\right) = 0 \tag{11}$$

The kernel $k\left(x, x_i\right)$ is a function that computes the inner product of the images produced in the feature space under the embedding Φ of two data points in the input space, Shawe-Taylor and Cristianini, (2004).
A covariance function , also called a kernel, kernel function, or covariance kernel, is a positive-definite function of two inputs x and x'. When we select a specific covariance function, we select if our solution should be smooth, linear, periodic and polynomial.
For a given training dataset $\mathcal{D}$, the covariance function generates the covariance

matrix, $\mathcal{K}(x, x')$ and this describes the correlation between different points in the process.

$$\mathbb{C}ov\left(f(x), f(x')\right) = \mathcal{K}(x, x') = \begin{pmatrix} \mathbb{C}ov\left(x_1, x_1\right) & \dots & \mathbb{C}ov\left(x_1, x_n\right) \\ \mathbb{C}ov\left(x_2, x_1\right) & \dots & \mathbb{C}ov\left(x_2, x_n\right) \\ \vdots & \vdots & \vdots \\ \mathbb{C}ov\left(x_n, x_1\right) & \dots & \mathbb{C}ov\left(x_n, x_n\right) \end{pmatrix} \tag{12}$$

where $\mathbb{C}ov\left(f(x_i), f(x_j)\right) = k\left(x_i, x_j\right)$.
We summarize the kernels for three common types of support vector machines: polynomial learning machine, radial-basis-function network, and two-layer perceptron

1. Polynomial learning machine kernel:

$$k\left(x, x_i\right) = \left(x^T x_i + 1\right)^p, \quad i = 1, \dots, n$$

   power p is specified a prior.
2. Radial-basis-function network kernel:

$$k\left(x, x_i\right) = \exp\left(-\frac{1}{2\sigma^2}\|x - x_i\|^2\right), \quad i = 1, \dots, n$$

   The width σ^2, common to all the kernels, is specified a prior.
3. Two-layer perceptron:

$$k\left(x, x_i\right) = \tanh\left(\beta_0 x^T x_i + \beta_1\right), \quad i = 1, \dots, n$$

4. A nonstationary neural network kernel:

$$k\left(x_i, x_j\right) = \frac{2}{\pi}\sin^{-1}\left(\frac{2\hat{x}_i^T \Sigma_* \hat{x}_j^T}{\left(1 + 2\hat{x}_i^T \Sigma_* \hat{x}_i^T\right)\left(1 + 2\hat{x}_j^T \Sigma_* \hat{x}_j^T\right)}\right)$$

   where $\hat{x} = (1, x_1, \dots, x_n)$ is the input vector and $\Sigma_* = diag\left(\sigma_0^2, \sigma_1^2, \dots, \sigma_n^2\right)$ is a diagonal weight prior, σ_0^2 is a variance for bias parameter.
5. Periodic kernel:

$$k\left(x_i, x_j\right) = \sigma_p \exp\left\{-\frac{2}{l^2}\sin^2\left[\pi\left(\frac{x_i - x_j}{p}\right)\right]\right\}$$

   where l is a parameter that controls the smoothness of the function and p governs the inverse length of the periodicity.

For the symmetric kernel $k\left(x, x_i\right)$ is an special case of Mercer's theorem, Mercer, (1909); Courant and Hilbert, (1970). Let $k(x, x')$ be a continuous symmetric kernel that is defined in the closed interval $a \leq x \leq b$. The kernel $k(x, x')$ can be expanded in the series:

$$k\left(x, x'\right) = \sum_{i=1}^{\infty} \lambda_i \varphi_i(x) \varphi_i(x')$$

with positive coefficients $\lambda_i > 0$ for all i. For this expansion to be valid and for it to converge absolutely and uniformly, it is necessary and sufficient that the condition

$$\int_b^a \int_b^a k(x, x')\,\psi(x)\psi(x')dxdx' \geq 0$$

holds for all $\psi(.)$, for which we have $\int_b^a \psi^2(x)dx < \infty$. where a and b are the constants of integretion. The features $\varphi_i(x)$ are called eigenfunctions of the expansion, and the numbers λ_i are called eigenvalues, Haykin (2009).

3.3 Particle Filtering

Considérese el problema de estimación de observaciones generadas en forma secuencial, mediante una ecuación de transición que describe la distribución de un proceso de Markov oculto denotado por $\{x_t, \quad t \in \mathbb{N}\}$, llamado vector de estados latentes (estados no observados), y una ecuación de observación que describe la verosimilitud de los datos medidos en tiempos discretos denotado por $\{y_t, \quad t \in \mathbb{N}\}$. El modelo es definido en términos de las densidades de probabilidades:

$$\begin{aligned} x_t &= f(x_{t-1}) + u_t \quad u_t \sim N(0, \sigma_u^2) \quad \text{(state evolution density)} \\ y_t &= h(x_t) + v_t \quad v_t \sim N(0, \sigma_v^2) \quad \text{(observation density)} \end{aligned} \tag{13}$$

dónde $x_t \in \mathbb{R}^{n_x}$: representan los estados no observados del sistema, $y_t \in \mathbb{R}^{n_y}$: representan las observaciones en el tiempo t, $f(.), h(.)$: representan las funciones no lineales de los estados y de las observaciones, $u_t \in \mathbb{R}^{n_u}, v_t \in \mathbb{R}^{n_v}$: representa a los procesos de ruido blanco. El interés recae en estimar los estados desconocidos $\mathbf{x_{1:t}} = \{x_1, \ldots x_t\}$ a partir de las mediciones $\mathbf{y_{1:t}} = \{y_1, \ldots y_t\}$. La distribución conjunta de los estados y las observaciones puede obtenerse directamente por la regla de la cadena de probabilidad:

$$p(x_{1:t}, y_{1:t}) = f(x_1) \left(\prod_{k=2}^{t} f(x_k|x_{k-1}) \right) \left(\prod_{k=1}^{t} h(y_k|x_k) \right)$$

where $f(x_1)$ is the distribution of the initial state.
Para hacer inferencia basados en modelos espacio de estados, se lleva acabo mediante una estimación secuencial de la distribución filtrada $p(\mathbf{x}_{1:t}|y_{1:t})$, en particular, interesa $p(x_t|\mathbf{y_{1:t}})$, que puede ser estimada en dos etapas:

1. **Analysis step:**

$$p(x_t|\mathbf{y_{1:t-1}}) = \int p(x_{t-1}|\mathbf{y_{1:t-1}}) f(x_t|x_{t-1}) dx_{t-1} \tag{14}$$

2. **Forecast step::**

$$p(x_t|\mathbf{y_{1:t}}) = \frac{h(y_t|x_t)p(x_t|\mathbf{y_{1:t-1}})}{p(y_t|\mathbf{y_{1:t-1}})} \tag{15}$$

donde:

$$p(y_t|\mathbf{y_{1:t-1}}) = \int h(y_t|x_t)p(x_t|\mathbf{y_{1:t-1}})dx_t \tag{16}$$

Para que esta inferencia se pueda realizar en modelos de alta dimensión, y con estructuras no lineales se han propuesto muchos técnicas de aproximaciones; en particular, se proponen utilizar los particle filtering methods, the filtering density is approximated with an empirical distribution formed from point masses, or particles. Suppose that we have at time $t-1$ weighted particles

$$\left\{\mathbf{x_{1:t-1}^{(i)}}, \omega_{t-1}^{(i)}, i = 1, \ldots, N\right\} \tag{17}$$

drawn from the smoothing density $p(x_{t-1}|\mathbf{y_{1:t-1}})$, We can consider this an empirical approximation for the density made up of point masses,

$$p_N\left(x_{t-1}|\mathbf{y_{1:t-1}}\right) \approx \sum_{i=1}^{N} \omega_t^{(i)} \delta_{x_{t-1}^{(i)}}(x_{t-1}), \quad \sum_{i=1}^{N} \omega_t^{(i)} = 1, \quad \omega_t^{(i)} \geq 0 \tag{18}$$

where $\delta_{x_{t-1}^{(i)}}(x_{t-1})$ denotes the Dirac-delta function. Si $\{\omega_t^{(i)}, i = 1, \ldots, N\}$, son elegidos adecuadamente Crisan and Doucet, 2002, probaron que:

$$\lim_{N\to\infty} p_N\left(x_t|\mathbf{y_{1:t}}\right) = p(x_t|\mathbf{y_{1:t}}) \tag{19}$$

To update the smoothing density from time $t-1$ to time t, factorize it as

$$p_N\left(x_t|\mathbf{y_{1:t}}\right) = p_N\left(x_{t-1}|\mathbf{y_{1:t-1}}\right) \frac{h\left(y_t|x_t\right) f\left(x_t|x_{t-1}\right)}{p\left(y_t|\mathbf{y_{1:t-1}}\right)} \tag{20}$$

We select N trajectories are drawn at random with replacement from $\left\{\mathbf{x_{1:t}^{(i)}}, i = 1, \ldots, N\right\}$ with probabilities $\left\{\omega_{\mathbf{t}}^{(\mathbf{i})}, i = 1, \ldots, N\right\}$. A new state is then generated randomly from an importance distribution, $q(x_t|x_{t-1}, y_t)$, and appended to the corresponding trajectory, x_{t-1}. The importance weight is updated to:

$$\omega_t^{(i)} = \frac{h(y_t|x_t^{(i)}) f(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|x_{t-1}^{(i)}, y_t)} \omega_{t-1}^{(i)} \tag{21}$$

where:

$$\omega_{t-1}^{(i)} = \frac{p(x_{t-1}^{(i)}|\mathbf{y_{1:t-1}})}{q(x_{t-1}^{(i)}|\mathbf{y_{1:t-1}})} \tag{22}$$

then

$$p_N(x_t|\mathbf{y_{1:t}}) \approx \sum_{i=1}^{N} \omega_t^{(i)} \delta_{x_t^{(i)}}(x_t) \tag{23}$$

Given at time $t-1$, $N \in \mathbb{N}$ random samples $\{\mathbf{x}_{1:t-1}^{(i)}\}$ distributed approximately according to $p\left(x_{t-1}|\mathbf{y_{1:t-1}}\right)$, the Monte Carlo filter proceeds as follows at time t:

Step 1: Sequential Importance Sampling

– Generate N i.i.d. samples $\{\tilde{x}_t^{(i)}, i = 1, \ldots, N\}$ from the proposal density $q(x)$:

$$\tilde{x}_t^{(i)} \sim q(x_t|\mathbf{x_{1:t-1}^{(i)}}, \mathbf{y_{1:t}}) = f(x_t|\tilde{x}_{t-1}^{(i)}) + u_t^{(i)} \quad , \quad u_t^{(i)} \sim N(0, \sigma_u^2)$$

and set $\tilde{\mathbf{x}}_{\mathbf{1:t}}^{\mathbf{(i)}} = \{\mathbf{x_{1:t-1}^{(i)}}, \tilde{x}_t^{(i)}\}$.
– For $i = 1, ..., N$, evaluate the importance weights up to a normalizing constant

$$\omega_t^{(i)} \propto \frac{h\left(y_t|\mathbf{y_{1:t-1}}, \tilde{\mathbf{x}}_{\mathbf{1:t}}^{\mathbf{(i)}}\right) f\left(\tilde{x}_t^{(i)}|\tilde{x}_{t-1}^{(i)}\right)}{q(x_t|\mathbf{x_{1:t-1}^{(i)}}, \mathbf{y_{1:t}})}$$

– For $i = 1, ..., N$, normalize the importance weights:

$$\tilde{\omega}_t^{(i)} = \frac{\omega_t^{(i)}}{\sum_{j=1}^{N} \omega_t^{(j)}} \quad , \quad \sum_{i=1}^{N} \tilde{\omega}_t^{(i)} = 1$$

– Evaluate $\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} [\tilde{w}_t^{(i)}]^2}$

Step 2 Resampling
– If $\hat{N}_{eff} \geq N_{thres}$,

$$x_{1:t}^{(i)} = \tilde{x}_{1:t}^{(i)}, \quad for \quad i = 1, \ldots, N$$

otherwise
– For $i = 1, ..., N$, sample an index $j(i)$ distributed according to the discrete distribution with N elements satisfying

$$p\left(j(i) = l\right) = \tilde{\omega}_t^{(l)}, \quad for \quad l = 1, \ldots, N$$

– For $i = 1, ..., N$, $\mathbf{x_{1:t}^{(i)}} = \tilde{\mathbf{x}}_{\mathbf{1:t}}^{j(i)}$ and $\tilde{w}_t^{(i)} = \frac{1}{N}$.

3.4 SVR-PF

JUAN This is the proposed approach...

3.5 Data sets

JUAN CPI of Ecuador. CPI India, CPI china

3.6 Metrics

En la literatura existe una variedad de métricas que se utilizan para medir la performance de los modelos, que se basan en la diferencia entre valor verdadero y el valor estimado $(y - \hat{y})$, o entre la diferencia al cuadrado $(y - \hat{y})^2$. Estas métricas están relacionadas con las funciones de pérdida en las normas L_1 y L_2 que

minimizan el error cuando se suman todas las diferencias. Estas medidas ponen énfasis en los errores, debido a la utilización de una norma L_2, las predicciones que se alejan de los valores reales se penalizan en mayor medida en comparación con las predicciones más cercanas.
In the study of time series, the statistical measurement that are commonly used to determine the appropriate lag length in the time series are: the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). Selection is based on the model with the lowest AIC and BIC values.

Mean Squared Error and Root Mean Squared Error: The MSE (Mean Squared Error), and RMSE (Root Mean Squared Error), often referred to as quadratic loss or L_2 loss is a standard metrics used in model evaluation. For a sample of n observations (y_i) and n corresponding model predictions $\hat{y}_i$, the MSE is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

and the RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} = \sqrt{MSE}$$

La raíz no afecta a los rangos relativos de los modelos, pero produce una métrica con las mismas unidades de y, lo cual es conveniente para estimar el error típico bajo errores distribuidos normalmente. The RMSE has been used as a standard statistical metric to measure model performance in research studies, Chai and Draxler (2014).

Mean Absolute Error: The Mean Absolute Error (MAE) measures the average of the sum of absolute differences between observation values and predicted values. The MAE is another useful measure widely used in model evaluation. Then, MAE is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Absolute Percentage Error: The mean absolute percentage error (MAPE) is one of the most popular measures of the forecast accuracy due to its advantages of scale-independency and interpretability. MAPE is the average of absolute percentage errors (APE). Let y_i At and $\hat{y}_i$ denote the actual and forecast values at data point i, respectively. Then, MAPE is defined as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

where n is the number of data points.

4 Results

JUAN

5 Discussion

JUAN

6 Conclusions

Acknowledgements Please place your acknowledgments at the end of the paper, preceded by an unnumbered run-in heading (i.e. 3rd-level heading).

References

1. Chai, T. and Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?. Arguments against avoiding RMSE in the literature, Geosci. Model Dev., 7, 1247-1250, https://doi.org/10.5194/gmd-7-1247-2014.
2. Doucet, A; de Freitas, J; Gordon, N. (2001). An introduction to sequential Monte Carlo methods, in Sequential Monte Carlo Methods in Practice, A. Doucet, J. F. G. de Freitas, and N. J. Gordon, Eds. New York: Springer-Verlag.
3. Arulampalam, M; Maskell, S; Gordon, N; Clapp, T. (2002). A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 50, NO. 2.
4. Ma, X; Karkus, P; Hsu, D; Lee, W. (2020). Particle Filter Recurrent Neural Networks. The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20).
5. Kitagawa, G. (1996). Monte Carlo filter and smoother for Non-Gaussian nonlinear state space models. Journal of Computational and Graphical Statistics, 5(1), 1–25.
6. Jung, S; Schlangen, I; Charlish, A. (2019). Sequential Monte Carlo Filtering with Long Short-Term Memory Prediction. Conference: 22nd International Conference on Information FusionAt: Ottawa.
7. Gordon, N ; Salmond, D; Smith, A.F.M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. Volume 140, Issue 2, p. 107-113. DOI: 10.1049/ip-f-2.1993.0015 , Print ISSN 0956-375X. Online ISSN 2053-9045.
8. Doucet, A; Godsill, S; Andrieu, C. (2000). On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. Statistics and Computing. Volume 10, 3,197-208.
9. Godsill, S; Arnaud Doucet; West, M. (2004). Monte Carlo smoothing for nonlinear time series. Journal of the American Statistical Association, vol. 99, no. 465.

10. Ristic, B; Arulampalam, S; Gordon, N. (2003). Beyond the Kalman filter: Particle filters for tracking applications. Artech house.
11. Choe, Y; Shin, J; Spencer, N. (2017). Probabilistic Interpretations of Recurrent Neural Networks. Final Report, 10-708 Probabilistic Graphical Models.
12. Cho, K; Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179.
13. Hochreiter, S; and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8):1735–1780.
14. Liu, Y; Cheng, J; Zhang, H; Zou, H; Xion, N. (2020). Long Short-Term Memory Networks Based on Particle Filter for Object Tracking. Digital Object Identifier 10.1109/ACCESS.2020.3041294.
15. V. N. Vapnik, and A. Y. Chervonenkis. (1964). A note on one class of perceptrons, Automation and Remote Control, vol.25, pp. 821 837.
16. Muthukrishnan, R; Maryam Jamila, S .(2020). Predictive Modeling Using Support Vector Regression. International Journal of Scientific and Technology Research. 9(2):4863-4865
17. Bertsekas,D.P.(1995). Nonlinear Programming. Belmont, MA:Athenas Scientific.
18. Shawe-Taylor, J., and N. Cristianini. (2004). Kernel Methods for Pattern Analysis, Cambridge, U.K., and New York: Cambridge University Press.
19. Mercer, J., (1909).Functions of positive and negative type, and their connection with the theory of integral equations,"Transactions of the London Philosophical Society (A), vol. 209, pp. 415–446.
20. Courant, R., and D. Hilbert. (1970). Methods of Mathematical Physics, vol. I and II, New York:Wiley Interscience
21. Aisyah, W.I.W.M.N.; Muhamad Safiih, L.; Razak, Z.; Nurul Hila, Z.; Abd Aziz, K.A.H.; Elayaraja, A.; Nor Shairah, A.Z, (2021). Improved of Forecasting Sea Surface Temperature based on Hybrid ARIMA and Vector Machines Model. Malays. J. Fundam. Appl. Sci,17, 609–620.
22. K Abdul Hamid, A.A.; Wan Mohamad Nawi, W.I.A.; Lola, M.S.; Mustafa, W.A.; Abdul Malik, S.M.; Zakaria, S.; Aruchunan, E.; Zainuddin, N.H.; Gobithaasan, R.U.; Abdullah, M.T. (2023). Improvement of Time Forecasting Models Using Machine Learning for Future Pandemic Applications Based on COVID-19 Data 2020–2022. Diagnostics, 13, 1121. https://doi.org/10.3390/diagnostics13061121.
23. Dama, F; Sinoquet C. (2021). Analysis and modeling to forecast in time series: a systematic review. arXiv:2104.00164v1 [cs.LG].
24. Haykin, S. (2009). Neural Networks and Learning Machines. Third Edition, Pearson Education, Inc., McMaster University, Hamilton.