Supervised Learning Based Online Filters for Targets Tracking Using Radar Measurements

Jie Deng, Wei Yi, Kai Zeng, Qiyun Peng and Xiaobo Yang
University of Electronic Science and Technology of China, Chengdu, P.R. China
E-mail: jiedeng_sc@outlook.com; kussoyi@gmail.com; 201921010930@std.uestc.edu.cn; qiyunpeng13@std.uestc.edu.cn

***Abstract*—In the field of the radar target tracking, the state filtering plays an important role in estimating the target state. One of the widely-adopted filter is the Bayesian filter, which requires the prior information and an accurate modeling of the real tracking scene. Thus, the matching degree of the dynamic model has a key impact on the state estimation accuracy of the Bayesian filter. However, the target dynamic and radar measurement models cannot be approximated perfectly in an unknown and complicated environment, and the state estimation accuracy of the model-based filter is limited. To address the limitations of the Bayesian filter, a supervised learning based online filter for target tracking is proposed in this paper. In the proposed filter, a mapping among the radar measurements is first established in the context of the polar coordinate system. Then, based on data-driven, the state filtering is directly implemented to obtain the state estimate by using this mapping relationship. As such, the prior information is not required in the proposed filter, hence the proposed filter inherits a good estimation accuracy in unknown and complicated environments. Finally, simulation experiments clarify the effectiveness of our proposed filter via comparing with the traditional filter.**

***Index Terms*—XGBoost, radar measurements, data-driven, filter.**

I. Introduction

In the application of radar target tracking, the state filter is an important module. It is mainly used to address the state estimation problem in time series by using the historical noise measurement data. Herein, the Bayesian filter is a classic filtering framework [1], which is widely used in the target tracking. Nowadays, based on this framework, a series of available filters have been developed. For example, the Kalman filter (KF), the extended Kalman filter (EKF), the unscented Kalman filter (UKF) and the particle filter (PF). More specifically, for the radar target tracking problems, the EKF, the UKF and the PF are often used to solve the filtering problem in nonlinear non-Gaussian systems [1], [2].

For the traditional Bayesian filter, it is necessary to model the dynamic system as well as set and estimate reasonable prior parameters to achieve an accurate description of the system (i.e., the model and the real system are required to match as much as possible). However, the target statistical characteristics cannot be obtained perfectly in practical environments, which makes the estimation of the priori information incorrect. Therefore, the mismatch of the preset parameter model will cause the tracking performance to decline or even diverge in these cases, which is also a limitation of the model-based filter. In order to address this problem, many works were performed. For example, Dang *et al*. [3] and Tripathi *et al*. [4] proposed an adaptive filter for maneuvering targets and unknown noise. An adaptive scheme is proposed in [5] by considering the measurement noise covariance distribution. Although these studies have effects on some specific scenarios, they are not a general method and also increase the complexity of the model, which motivates us to consider a new solution way.

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs [6]. It can avoid modeling the motion system by building a data mapping relationship and get rid of the limitations of model-based filters. Nowadays, supervised learning has been widely used in data prediction, image recognition and text classification, which has achieved good results [7]. There are still some studies on the application of supervised learning in the filters. For example, the combination of the neural network (NN) and KF was first proposed in [8], in which the NN is used to train the residual. Guo *et al*. [9] used the trained NN to compensate the EKF drift in the GPS system. However, they both used supervised learning to train and assist the residuals in [8] and [9]. Subsequently, a random forest (RF) based filter that directly maps the measurement to the estimation was proposed in [10]. On the basis of [10], using the eXtreme Gradient Boosting (XGBoost) to improve the estimation accuracy in [11]. Gao *et al*. [12] considered using Recurrent Neural Network (RNN) for nonlinear target tracking, which has a good estimation ability. However, none of the above-mentioned work [10] – [12] extracts the target motion characteristics, directly used the existing data for training. Therefore, consider extracting the motion change information of training data was proposed in [13], including three aspects of time, space, and angle, then they proposed a XGBoost based filter (XGBF). However, only the Cartesian coordinate system is considered in [13], the polar coordinate system of the radar are not analyzed. If the training data is directly converted from the polar coordinate system to the Cartesian coordinate system for training in [13], there will be influences in the near and far regions and conversion error. The nonlinear conversion of the data will lead to inconsistent training characteristics in different regions, which will affect the training effect.

This work was supported in part by the National Natural Science Foundation of China under Grants 61771110, in part by Chang Jiang Scholars Program, in part by the 111 Project B17008.

(*Corresponding author: Wei Yi*, E-mail: kussoyi@gmail.com.)

Although the above-mentioned work can obtain a good filtering effect, it has some deficiencies in the application of radar target filter. Therefore, on the basis of [13], this paper further extends supervised learning based filter to the radar field. More specifically, this paper extracts the measurements from the radar data and directly trains a polar coordinate model. Herein, the training dimension is based on polar coordinates, eliminating the coordinate system conversion error for training. Thus, further address the problem of inconsistent training characteristics in different regions. In this paper, we propose a supervised learning based filter with radar measurements (SLFR), which is still implemented by XGBoost and construct a new filter framework based on supervised learning under radar data. The numerical experiments prove the effectiveness of the proposed filter.

II. Model based Bayesian Filters (MBF)

For the nonlinear filtering, we use the state and measurement equation to describe a dynamic system [1] as follows

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}), \tag{1}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k), \tag{2}$$

where $\mathbf{f}_k$ is the state transition function, $\{\mathbf{w}_{k-1}, k \in \mathbb{N}\}$ is an i.i.d. process noise, $\mathbf{h}_k$ is the measurement function, $\{\mathbf{v}_k, k \in \mathbb{N}\}$ is an i.i.d. measurement noise, $\mathbb{N}$ is the set of natural numbers, $\mathbf{x}_k$ is the target state at time k, $\mathbf{z}_k$ is the measurement at time k.

Meanwhile, we define $\mathbf{z}_{1:k} = \{\mathbf{z}_i, i = 1, ..., k\}$ as the set of all available measurements. From the Bayesian perspective, we construct the posterior probability density function $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. Use the Chapman Kolmogorov equation and Bayesian criterion to recursively update $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ as follows

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right) = \int p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) p\left(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}\right) d\mathbf{x}_{k-1}, \tag{3}$$

$$p\left(\mathbf{x}_k|\mathbf{z}_{1:k}\right) = \frac{p\left(\mathbf{z}_k|\mathbf{x}_k\right) p\left(\mathbf{x}_k|\mathbf{z}_{1:k-1}\right)}{p\left(\mathbf{z}_k|\mathbf{z}_{1:k-1}\right)}, \tag{4}$$

where $p\left(\mathbf{x}_k|\mathbf{x}_{k-1}\right) = p\left(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_{1:k-1}\right)$.

According to the minimum mean square error (MMSE) criterion, the state with the greatest posterior probability density is taken as the optimal estimation of the system state, which can described as

$$\hat{\mathbf{x}}_k = E\left[\mathbf{x}_k \left| \mathbf{z}_{1:k}\right.\right] = \int \mathbf{x}_k p\left(\mathbf{x}_k \left| \mathbf{z}_{1:k}\right.\right) d\mathbf{x}_k. \tag{5}$$

Therefore, we learn that the MBF needs to satisfy the state space equation and the measurement equation models to filter recursively. In other words, the modeling and the estimation of prior parameters need to be as accurate as possible.

III. Supervised Learning Based Filters with Radar Measurements (SLFR)

A. Problem Statement

1) SLF Review: A supervised learning based online tracking filter (SLF) was proposed in [13]. SLF transforms the filtering process into a data training and prediction process based on the sensor data.

Step 1 – *The introduction to supervised learning:* Supervised learning contains the following elements: input features, output variables, hypothesis functions and loss functions. Suppose the input feature is $\mathbf{a}$ and the output variable is $\mathbf{b}$. $h(\mathbf{a}, \boldsymbol{\theta})$ is the estimator corresponding to $\mathbf{b}$, where $\boldsymbol{\theta}$ represents the parameter in h. Use the loss function $J(\boldsymbol{\theta})$ to judge the quality of the parameter $\boldsymbol{\theta}$. $J(\boldsymbol{\theta})$ is defined as

$$J(\boldsymbol{\theta}) = \sum_i (b^{(i)}, h(\mathbf{a}^{(i)}, \boldsymbol{\theta})) + \lambda \boldsymbol{\Omega}(\boldsymbol{\theta}), \tag{6}$$

where $b^{(i)}$ is the real output corresponding to the i-th input feature, $l(\cdot)$ is the training error term (e.g., the square error function, the logistic loss function), $\boldsymbol{\Omega}(\boldsymbol{\theta})$ is the regularization term, which denotes the complexity of the model and prevents over-fitting, λ controls the trade-off between the data fitting error and the model fitting error [14].

Step 2 – *SLF data preprocessing stage:* Take each point in Cartesian coordinate systems as a sample. The data preprocessing is performed for each sample. Thus, forming the input feature to represent the "motion information" of each sample.

Step 3 – *SLF training and application stage:* SLF uses input features and output variables to construct a hypothesis function h. By minimizing the loss function $J(\boldsymbol{\theta})$, iteratively searches for an optimal parameter $\hat{\boldsymbol{\theta}}$. Subsequently, SLF performs the same preprocessing on the new measurement and puts it into the trained hypothesis function to make predictions.

2) Deficiencies in SLF: SLF only trains and predicts the data in Cartesian coordinates, which does not analyze the data in nonlinear scenes. For the filtering problem of radar target tracking, SLF cannot be used directly. It needs to convert the radar data to the Cartesian coordinate system for preprocessing and training. Thus, there is a conversion error. Meanwhile, the nonlinear characteristics of the data will affect the training characteristics of different regions (i.e., the effect of the radar's near and far regions).

B. Data-Driven SLFR Using Radar Measurements

Without the need for accurate matching and parameter estimation of the model, the supervised learning is based on data-driven and directly considers the mapping relationship between the data itself. Based on the advantages of supervised learning and its regression technique for the prediction of continuous response [7], the SLF framework proposed by [13] has a positive effect.

However, SLF didn't consider the filtering problem in the radar scene. To address the filtering problem of radar targets, this paper extends the SLF framework of [13] and proposes a supervised learning based filter with radar measurements (SLFR). For the radar measurement, we construct the input feature that expresses the motion information. More specifically, we directly train a machine model under the polar coordinate system. Since the training area is the polar coordinate system, there is no need to perform the coordinate conversion on the measurement during training. Therefore, there is no effects in the near and far regions during training, which can solve the shortcomings of SLF in polar coordinate systems.

1) SLFR Data Preprocessing Stage: For the radar measurement, $\mathbf{z}_k^j = [\rho_k^j, \alpha_k^j]^\top$ is the j-th radar measurement at time k where "$^\top$" denotes the matrix transpose. Thus, the historical measurement sequence is $\mathbf{z}_{1:k}^j = \{\mathbf{z}_i^j, i = 1, \ldots, k\}$. These radar measurements are converted into input features for training after preprocessing. Let $\mathbf{x}_k^j$ denote the true state of the j-th track, which becomes the output variable after proper preprocessing.

Step 1 – *The time aspect:* For a sample, the "sliding window" measure is adopted to construct a fixed number of features. While ensuring the full expression of the "information" of the sample, it avoids complicated computation and excessive noise introduction. The sliding window processing for $\mathbf{z}_{1:k}^j$ is as follows

$$\mathbf{Z}_{k,\tau}^j = \left\{\mathbf{z}_i^j, i = k-\tau+1, \ldots, k-1, k\right\}, k \geq \tau, \quad (7)$$

where τ represents the sliding window length.

It is indicated in (7) that the τ recent measurements are truncated from the historical measurement of the j-th track. The case of $k < \tau$ is in Section III-C.

Step 2 – *The space aspect:* The "relative measurement" measure is used to eliminate the spatial impact of different initial positions. Since the relative motion information is extracted, all input feature elements only save the relative displacement, which is suitable for more data ranges. The specific processing is given by

$$\begin{aligned}\tilde{\mathbf{Z}}_{k,\tau}^j &= \left\{\tilde{\mathbf{z}}_{k-\tau+1}^j, \ldots, \tilde{\mathbf{z}}_{k-1}^j\right\} \\ &= \left\{\mathcal{F}\left(\mathbf{z}_{k-\tau+1}^j, \mathbf{z}_k^j\right), \ldots, \mathcal{F}\left(\mathbf{z}_{k-1}^j, \mathbf{z}_k^j\right)\right\},\end{aligned} \quad (8)$$

where $\tilde{\mathbf{z}}_i^j = \mathcal{F}\left(\mathbf{z}_i^j, \mathbf{z}_k^j\right), i \neq k$ represents the relative distance $(\Delta\rho, \Delta\alpha)$ between the i-th measurement $\mathbf{z}_i^j$ and the current k-th measurement $\mathbf{z}_k^j$ in the polar coordinate system. Let $z_{i,\rho}^j$ and $z_{i,\alpha}^j$ represent the range and bearing, respectively. $\mathcal{F}(\cdot)$ is

$$\mathcal{F}\left(\mathbf{z}_i^j, \mathbf{z}_k^j\right) = \left(\mathbf{z}_i^j - \mathbf{z}_k^j\right) = \begin{bmatrix} z_{i,\rho}^j - z_{k,\rho}^j \\ z_{i,\alpha}^j - z_{k,\alpha}^j \end{bmatrix}, i \neq k. \quad (9)$$

After the above space-time preprocessing, each sample has $L = 2*(\tau-1)$ input features, which represent the stored motion change features in polar coordinates.

According to the real position $[x_k, y_k]^\top$ in the Cartesian coordinate system, the real position $\mathbf{x}_{k,\mathrm{p}}^j$ in the polar coordinate system can be obtained. If $\mathbf{x}_{k,\mathrm{p}}^j$ is used directly, it will lead to the poor generalization ability. Therefore, the error $\mathbf{r}_k^j = \mathbf{x}_{k,\mathrm{p}}^j - \mathbf{z}_k^j$ is used as the output variable.

2) SLFR Training Stage: Based on the radar data, we construct a training data set Φ of N tracks with time T as follows

$$\mathbf{\Phi} = \left\{\left(\mathbf{z}_{1:k}^i, \mathbf{x}_k^i\right), i = 1, ..., N; k = 1, ..., T\right\}. \quad (10)$$

For $\mathbf{\Phi}$, "1) SLFR data preprocessing" is used to effectively extract the target motion information to form input features $\tilde{\mathbf{Z}}_{k,\tau}^j$ and output variables $\mathbf{r}_k^j$. Therefore, we can construct an available training set.

The hypothesis function h establishes the mapping relationship between $\tilde{\mathbf{Z}}_{k,\tau}^j$ and $\mathbf{r}_k^j$ is given by

$$h\left(\tilde{\mathbf{Z}}_{k,\tau}^j, \boldsymbol{\theta}\right) : \tilde{\mathbf{Z}}_{k,\tau}^j \to \mathbf{r}_k^j. \quad (11)$$

We want the state estimation $\hat{\mathbf{r}}_k^j$ output by the hypothesis function h can approximate the output variable $\mathbf{r}_k^j$ as much as possible. Thus, the optimal parameter $\hat{\boldsymbol{\theta}}$ is found by minimizing the loss function as follows

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{j=1}^{N}\sum_{k=1}^{T} l\left(\mathbf{r}_k^j, h\left(\tilde{\mathbf{Z}}_{k,\tau}^j, \boldsymbol{\theta}\right)\right) + \lambda\mathbf{\Omega}\left(\boldsymbol{\theta}\right). \quad (12)$$

Because the filtering problem belongs to the regression problem, choose the root mean square error (RMSE) as the loss function training error term $l(\cdot)$ [11] as follows

$$l\left(h\left(\tilde{\mathbf{Z}}_{k,\tau}^j, \hat{\boldsymbol{\theta}}\right), \mathbf{r}_k^j\right) = \sqrt{\frac{1}{NT}\sum_{j=1}^{N}\sum_{k=1}^{T}\left\|\mathbf{r}_k^j - h\left(\tilde{\mathbf{Z}}_{k,\tau}^j, \hat{\boldsymbol{\theta}}\right)\right\|_2^2}, \quad (13)$$

Based on (12) and (13) continuous iterative search to find the optimal parameter $\hat{\boldsymbol{\theta}}$, a machine model network is trained.

3) SLFR Estimation and Application Stage: Assuming that a batch of new radar measurements $\mathbf{z}_{1:k}^{j,\mathrm{n}}$ is obtained, where "n" stands for new. The "1) SLFR data preprocessing" is also performed to form the input features $\tilde{\mathbf{Z}}_{k,\tau}^{j,\mathrm{n}}$. Then, we put $\tilde{\mathbf{Z}}_{k,\tau}^{j,\mathrm{n}}$ into the trained hypothesis function h for filtering, which can be described as

$$\hat{\mathbf{r}}_k^{j,\mathrm{n}} = h\left(\tilde{\mathbf{Z}}_{k,\tau}^{j,\mathrm{n}}, \hat{\boldsymbol{\theta}}\right). \quad (14)$$

Perform an "inverse transformation" on the estimated new error $\hat{\mathbf{r}}_k^{j,\mathrm{n}}$ as shown in (15). Furthermore, the error compensation is performed on the measurement to obtain the corresponding state estimation as follows

$$\hat{\mathbf{x}}_{k,\mathrm{p}}^{j,\mathrm{n}} = \hat{\mathbf{r}}_k^{j,\mathrm{n}} + \mathbf{z}_k^{j,\mathrm{n}}. \quad (15)$$

Based on this stage, the filtering of SLFR is realized. Therefore, the state filtering value $\hat{\mathbf{x}}_{k,\mathrm{p}}^{j,\mathrm{n}}$ in the polar coordinate system can be obtained.

C. XGBoost Based Training Algorithm Implementation

The XGBoost model is proposed by Tianqi Chen in recent years, which has been widely applied in the data mining field [15]. XGBoost is one of the supervised learning algorithms. It is a tree model based on integrated thoughts, which has excellent performance in prediction [15] and classification [16]. Herein, we choose XGBoost as the training algorithm to implement SLFR.

In the data preprocessing stage, it is still processed according to the SLFR framework. For the case of $k < \tau$ in (7), XGBoost can specify the default direction of the branch for the missing value and automatically handle it [17].

In the training stage, because XGBoost is a tree-based algorithm, the parameter $\boldsymbol{\theta}$ of h consists of two parts: one is the structure of the tree, the other is the score of each leaf node [17]. Therefore, we need to find the optimal tree structure and the corresponding optimal leaf node by minimizing the loss function.

Here, we only focus on the training of h in the SLFR. The specific formula theory of XGBoost is in [17]. XGBoost is based on the classification and regression tree (CART) for learning and training. Therefore, for the filtering problem of the target tracking, the CART model is

$$\hat{\mathbf{r}}_k^j = \sum_{m=1}^{M} \mathbf{f}_m \left(\tilde{\mathbf{Z}}_{k,\tau}^j\right), \mathbf{f}_m \in \boldsymbol{\Gamma}, \tag{16}$$

where M represents the number of trees, $\boldsymbol{\Gamma}$ represents all CARTs, $\mathbf{f}_m$ represents a specific CART.

We consider the filtering problem of N tracks with time T, the total number of samples is $S = NT$. Rewrite $\tilde{\mathbf{Z}}_{k,\tau}^j$ to $\tilde{\mathbf{Z}}_{s,\tau}$, $\hat{\mathbf{r}}_k^j$ to $\hat{\mathbf{r}}_s$, and $\mathbf{r}_k^j$ to $\mathbf{r}_s$, where $s = 1, 2, ...S$. Therefore, the general loss function in XGBoost is

$$J = \sum_{s=1}^{S} l\left(\mathbf{r}_s, \hat{\mathbf{r}}_s\right) + \sum_{m=1}^{M} \boldsymbol{\Omega}\left(\mathbf{f}_m\right). \tag{17}$$

For SLFR, the training error term $l(\cdot)$ selects RMSE as shown in (13). In addition, since the tree model is an addition model, we add an optimal CART $\mathbf{f}_t$ at step t. $\mathbf{f}_t$ is given by

$$J^{(t)} = \sum_{s=1}^{S} l\left(\mathbf{r}_s, \hat{\mathbf{r}}_s^{(t-1)} + \mathbf{f}_t\left(\tilde{\mathbf{Z}}_{s,\tau}\right)\right) + \boldsymbol{\Omega}\left(\mathbf{f}_t\right) + \mathcal{C}, \tag{18}$$

where $\hat{\mathbf{r}}_s^{(t)}$ represents the output of the t-th tree, $\mathcal{C}$ refers to the regular term of the previous $t-1$ trees. Perform the second-order Taylor expansion on (18) and remove the constant term $\mathcal{C}$, we can get

$$J^{(t)} \approx \sum_{s=1}^{S} \left[\mathbf{e}_s \mathbf{f}_t\left(\tilde{\mathbf{Z}}_{s,\tau}\right) + \frac{1}{2}\mathbf{u}_s \mathbf{f}_t^2\left(\tilde{\mathbf{Z}}_{s,\tau}\right)\right] + \boldsymbol{\Omega}\left(\mathbf{f}_t\right), \tag{19}$$

where

$$\begin{cases} \mathbf{e}_s = \partial_{\hat{\mathbf{r}}_s^{(t-1)}} l\left(\mathbf{r}_s, \hat{\mathbf{r}}_s^{(t-1)}\right) \\ \mathbf{u}_s = \partial^2_{\hat{\mathbf{r}}_s^{(t-1)}} l\left(\mathbf{r}_s, \hat{\mathbf{r}}_s^{(t-1)}\right) \end{cases}. \tag{20}$$

For the regularization term $\boldsymbol{\Omega}\left(\mathbf{f}_t\right)$, XGBoost chooses it in [17] as

$$\boldsymbol{\Omega}\left(\mathbf{f}_t\right) = \gamma \mathcal{P} + \frac{1}{2}\lambda \sum_{j=1}^{\mathcal{P}} \boldsymbol{\omega}_j^2, \tag{21}$$

where $\mathcal{P}$ is the number of leaf nodes, the value of the $\mathcal{P}$ leaf node constitutes a $\mathcal{P}$-dimensional vector $\boldsymbol{\omega}$, γ and λ represent self-set parameter.

We obtain the loss function J and hypothesis function h of XGBoost in SLFR. The specific method to obtain the optimal tree structure and the corresponding optimal leaf node is in [17], this paper no longer conducts formula derivation.

IV. Simulations and Results

A. Radar Scene Training Data Set

In the field of machine learning, there is currently no data set for the filter of radar target tracking. Therefore, we again adopt the processing measures in [10] – [13] (i.e., select a specific model to generate the training data). It is worth noting that in actual scenes, GPS is often used to obtain the true target tracks. In general, most similar studies simulate the target motion tracks and measurements according to dynamics and measurement models. In addition, this paper mainly focuses on the problem of single-target filtering, the clutter and multi-target measurement correlation issues are not considered for the time being.

In radar target tracking scenarios, we consider that the track is transformed according to the constant velocity (CV) model. The radar's location $r = [r_x, r_y]^\top$. Based on these, a radar scene training data set is generated. The state and measurement equations are presented as

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{w}_k, \tag{22}$$

$$\mathbf{z}_k = \begin{bmatrix} \sqrt{\left(x_k - r_x\right)^2 + \left(y_k - r_y\right)^2} \\ \tan^{-1}\left(x_k - r_x, y_k - r_y\right) \end{bmatrix} + \mathbf{v}_k, \tag{23}$$

where $\mathbf{x}_k = [x_k, \dot{x}_k, y_k, \dot{y}_k]^\top$, $\mathbf{z}_k = [\rho_k, \alpha_k]^\top$, $\{\mathbf{w}_k, k \in \mathbb{N}\}$ and $\{\mathbf{v}_k, k \in \mathbb{N}\}$ are the i.i.d process noise and the measurement noise, respectively. $\mathbf{F}$ can be written as

$$\mathbf{F} = \mathbf{I}_2 \otimes \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \tag{24}$$

where $\mathbf{I}_2$ denotes the 2×2 identity matrix, $\otimes$ is the Kronecker product, Δt is the radar scan interval.

Meanwhile, it is reasonable to assume that the process noise and the measurement noise follow the zero-mean Gaussian distribution, $\mathbf{Q}$ and $\mathbf{R}$ are the covariances as follows

$$\mathbf{Q} = q_s \mathbf{I}_2 \otimes \begin{bmatrix} \Delta t^3/3 & \Delta t^2/2 \\ \Delta t^2/2 & \Delta t \end{bmatrix}, \tag{25}$$

$$\mathbf{R} = \begin{bmatrix} v_\rho^2 & 0 \\ 0 & v_\alpha^2 \end{bmatrix}, \tag{26}$$

where q_s is the process noise intensity, v_ρ and v_α are the measurement noise standard deviations of range and bearing, respectively.

Because this paper still uses XGBoost as the training algorithm to implement SLFR, an XGBoost based Filter (XGBF) is proposed. In general, we can get a series of measurements and true tracks based on the above formulas. Therefore, they are used as XGBF training sets to verify the effect of XGBF.

B. XGBF Hyper-Parameter Setting

Hyper-parameter is a key task to optimize the performance of machine learning algorithm. Similarly, the XGBF in this paper also needs to be adjusted and set for hyper-parameters. In order to limit the complexity of the search iteration and refer to the setting method in [13], this paper selects four hyper-parameters and sets their values (other hyper-parameters use the default values) [17] as shown in Table I.

TABLE I
XGBF HYPER-PARAMETER SETTING

Parameter	Definition	Value
eta	Learning rate	0.05
τ	Sliding window length	20
nrounds	Number of trees to be used in the model	500
max depth	Maximum depth of a tree	8

C. Performance Comparison

1) General Cases: Suppose the radar is located at $r = [0, 0]^\top$, the scan period Δt = 1s, the time T = 50s. The process noise satisfies the Gaussian distribution $w_k \sim \mathcal{N}(0, \mathbf{Q})$, where q_s = 1. The measurement noise satisfies the Gaussian distribution $v_k \sim \mathcal{N}(0, \mathbf{R})$, where $\mathbf{R}$ = diag{20, 0.00001}.

For the XGBF, we choose 10000 training samples and 5000 test samples in order to save computation. For the PF, we select the SIR filter and the number of tracks N = 5000 (i.e., 5000 Monte Carlo simulations), the number of particles P = 3000. Meanwhile, the initial state of the track is generated uniformly in [± 5000m, ± 25m/s, ±5000m, ± 30m/s]. Choose the RMSE as the estimation accuracy evaluation standard.

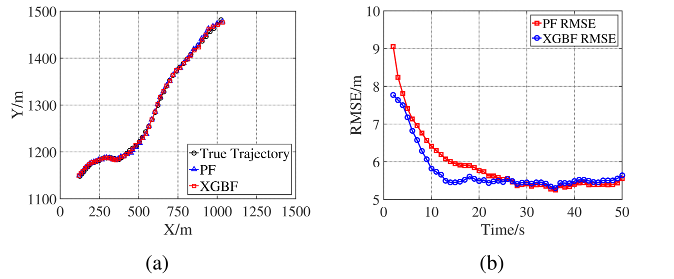


Fig. 1. (a) The target track in the nonlinear system. (b) The RMSE of the PF and the XGBF.

In Fig. 1, the PF belongs to the optimal filter when the number of particles is large enough, the XGBF in this paper can approximate the estimation accuracy of the PF in the radar scene. More specifically, Fig. 1 (b) shows that the XGBF is better than the PF in the early stage of filtering, while the PF is better than the XGBF in the later stage. This is because the Bayesian filter can gradually converge the covariance under ideal conditions, which gradually reduces the RMSE. However, in the complicated and unknown environment, the XGBF has unique advantages in below.

In addition, the insufficient number of training samples is also a reason why the XGBF's estimation accuracy is lower than the PF as shown in Fig 2. We can know that if the size of the training set is increased, the estimation accuracy of the XGBF can be closer to the PF.

2) Special Cases: In special cases, we consider the following three different cases:

- Case 1: the real situation is the time-varying process noise environment. In other words, the process noise changes continuously with time.

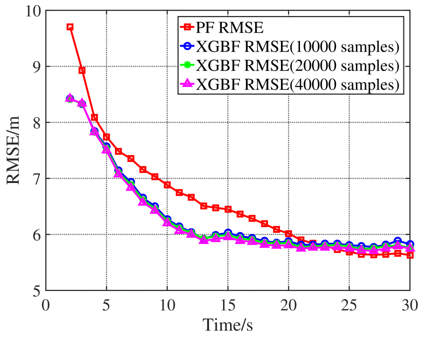


Fig. 2. The RMSE of the PF and the XGBF in larger training samples.

- Case 2: the real situation is the compound noise environment, More specifically, the process noise is an additive composite noise of Gaussian + exponent, the measurement noise is still Gaussian.
- Case 3: the real situation is the compound noise environment, the process noise is an additive composite noise of Gaussian + exponent, the measurement noise is a multiplicative noise of Gaussian ∗ exponent.

Case 1: T = 30s, $w_k \sim \mathcal{N}(0, \mathbf{Q})$ and the q_s changes with time, other conditions remain unchanged. Suppose q_s = 0.05t, but the model incorrectly estimates q_s = 0.1 in the PF. The estimation accuracy is shown in Fig. 3 (a).

Case 2: the state equation is

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{w}_k + \gamma_k, \tag{27}$$

where $w_k \sim \mathcal{N}(0, \mathbf{Q})$ and q_s = 1, γ_k follows the exponential distribution of the parameter $\lambda = 2.5$ (i.e., $\gamma_k \sim E(2.5)$). For the PF, the model is incorrectly modeled as only the Gaussian noise. Other conditions remain unchanged. At this moment, the estimation accuracy is shown in Fig. 3 (b).

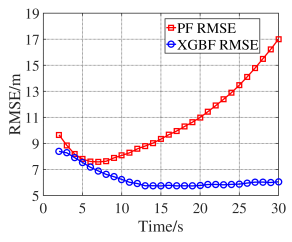
Case 3: the state equation is still (27), the measurement equation is

$$\mathbf{z}_k = \begin{bmatrix} \sqrt{(x_k - r_x)^2 + (y_k - r_y)^2} \\ \tan^{-1}(x_k - r_x, y_k - r_y) \end{bmatrix} + \mathbf{v}_k * \gamma_k, \tag{28}$$

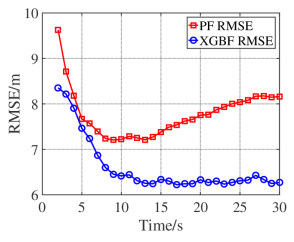
where $v_k \sim \mathcal{N}(0, \mathbf{R})$, $\gamma_k \sim E(3)$. For PF, the process noise and the measurement noise are modeled as only the Gaussian noise. The result is shown in Fig. 3 (c).

The results of Fig. 3 highlight that the XGBF of SLFR can be well applied to unknown and complicated noise environments in the radar filtering scenario. Because the XGBF is based on data-driven, the mapping relationship between data is learned to avoid the influence of model parameters. However, the estimation accuracy of the PF depends heavily on the prior information of the model, the model mismatch will cause the accuracy of the PF to drop sharply.

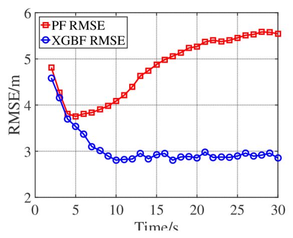
3) The Effect of the Number of Particles P: For the XGBF, the simulation conditions remain the same as above. For the PF, set the number of particles P = 1500 and P = 3000.



(a)



(b)



(c)

Fig. 3. The RMSE of the XGBF and the PF in different cases. (a) Case 1. (b) Case 2. (c) Case 3.

Comparing the estimation accuracy and execution time of the different P. The execution time is the time when all the track filtering is completed (for the XGBF execution time, only the estimation and application stage of new radar measurement is considered). Therefore, we can get Fig. 4 and Table II.

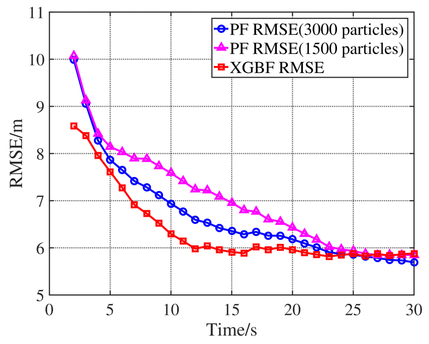


Fig. 4. The RMSE of the PF and the XGBF in the different particles P.

TABLE II
RMSE AND RUNTIMES FOR ALL TEST TRACKS

Algorithm	**RMSE(average)**	**Runtime**
XGBF	6.332	17.250s
PF(1500 particles)	6.929	91.546s
PF(3000 particles)	6.740	222.995s

In Fig. 4 and Table II, the XGBF can obtain better estimation accuracy with shorter execution time compared with the PF. Compared with the PF (1500 particles), the XGBF's estimation accuracy is improved by 8.6% and the execution time is reduced by 81.16%. For 3000 particles, the accuracy of PF in the later stage will be better than XGBF from Fig. 1 (b). However, when it comes to the execution time and the special cases of Fig. 3, the XGBF of SLFR still has a good filtering effect. After training, the XGBF has a fast execution speed. With the help of the GPUs, the execution time of XGBF can further reduced [12], which can achieve the real-time performance required by the radar system.

V. Conclusions

In this paper, we propose SLFR for radar target filtering scenarios. Meanwhile, we use the XGBoost to implement SLFR. Compared with MBF, it can avoid the estimation accuracy degradation caused by the model mismatch and the prior parameter estimation error in the unknown complicated noise environment. More specifically, Our method introduces the supervised learning in the filter of radar target tracking and provides a new way to solve the radar target filtering problem. In complicated and unknown noise environments, the simulation results show the advantages of proposed XGBF in the execution time and the estimation accuracy.

References

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, 2002.
[2] P. Rohal and J. Ochodnicky, "Radar target tracking by kalman and particle filter," in *Proc. Commun. Inf. Technol (KIT).*, 2017, pp. 1–4.
[3] Van Tho Dang, "An adaptive kalman filter for radar tracking application," in *Proc. Micro., Radar Rem. Sens. Sympo.*, 2008, pp. 261–264.
[4] R. P. Tripathi, S. Ghosh, and J. O. Chandle, "Tracking of object using optimal adaptive kalman filter," in *Proc. IEEE Int. Conf. Eng. Technol(ICETECH).*, 2016, pp. 1128–1131.
[5] A. Assa and K. N. Plataniotis, "Adaptive kalman filtering by covariance sampling," *IEEE Signal Processing Lett.*, vol. 24, no. 9, pp. 1288–1292, 2017.
[6] N. Pete and R. Stuart, *Artificial intelligence: a modern approach*. New York: Prentice hall, 2009.
[7] T. D. Phan, "Housing price prediction using machine learning algorithms: The case of melbourne city, australia," in *Proc. Int. Conf. Mach. Learn. Data. Eng.*, 2018, pp. 35–42.
[8] L. Chin, "Application of neural networks in target tracking data fusion," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 30, no. 1, pp. 281–287, 1994.
[9] H. Guo, "Neural network aided kalman filtering for integrated GPS/INS navigation system," *TELKOMNIKA Indonesian Journal of Electrical Engineering.*, vol. 11, no. 3, pp. 1221–1226, 2013.
[10] K. Thormann, F. Sigges, and M. Baum, "Learning an object tracker with a random forest and simulated measurements," in *Proc. Int. Conf. Inform. Fusion.*, 2017, pp. 1–4.
[11] B. Zhai, W. Yi, M. Li, H. Ju, and L. Kong, "Data-driven XGBoost-based filter for target tracking," *The Journal of Engineering.*, vol. 2019, no. 20, pp. 6683–6687, 2019.
[12] C. Gao, J. Yan, S. Zhou, B. Chen, and H. Liu, "Long short-term memory-based recurrent neural networks for nonlinear target tracking," *Signal Processing.*, vol. 164, 2019.
[13] J. Deng and W. Yi, "Supervised learning based online tracking filters: An XGBoost implementation," arXiv:2004.04975, 2020.
[14] T. Li, H. Chen, S. Sun, and J. M. Corchado, "Joint smoothing and tracking based on continuous-time target trajectory function fitting," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1476–1483, 2019.
[15] J. Zhong, Y. Sun, W. Peng, M. Xie, J. Yang, and X. Tang, "XGBFEMF: An XGBoost-based framework for essential protein prediction," *IEEE Trans. Nanobiosci.*, vol. 17, no. 3, pp. 243–250, 2018.
[16] W. Chen, K. Fu, J. Zuo, X. Zheng, T. Huang, and W. Ren, "Radar emitter classification for large data set based on weighted-XGBoost," *IET Radar, Sonar Navigation*, vol. 11, no. 8, pp. 1203–1207, 2017.
[17] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. ACM. Int. Conf. Knowledge Discovery Data Mining.*, 2016, pp. 785–794.