Research Article

Analyzing Machine Learning Models with Gaussian Process for the Indoor Positioning System

Yunxin Xie,¹ Chenyang Zhu ,² Wei Jiang,² Jia Bi ,³ and Zhengwei Zhu²

¹School of Petroleum Engineering, Changzhou University, Changzhou 213100, China

²School of Information Science and Engineering, Changzhou University, Changzhou 213100, China

³Electronics and Computer Science, University of Southampton, University Road, Southampton SO17 1BJ, UK

Correspondence should be addressed to Chenyang Zhu; cz4g16@soton.ac.uk

Received 18 August 2020; Revised 26 September 2020; Accepted 14 October 2020; Published 24 October 2020

Academic Editor: William Guo

Copyright © 2020 Yunxin Xie et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recently, there has been growing interest in improving the efficiency and accuracy of the Indoor Positioning System (IPS). The Received Signal Strength- (RSS-) based fingerprinting technique is essential for indoor localization. However, it is challenging to estimate the indoor position based on RSS's measurement under the complex indoor environment. This paper evaluates three machine learning approaches and Gaussian Process (GP) regression with three different kernels to get the best indoor positioning model. The hyperparameter tuning technique is used to select the optimum parameter set for each model. Experiments are carried out with RSS data from seven access points (AP). Results show that GP with a rational quadratic kernel and eXtreme gradient tree boosting model has the best positioning accuracy compared to other models. In contrast, the eXtreme gradient tree boosting model could achieve higher positioning accuracy with smaller training size and fewer access points.

1. Introduction

Wireless indoor positioning is attracting considerable critical attention due to the increasing demands on indoor location-based services. Examples of this service include guiding clients through a large building or help mobile robots with indoor navigation and localization [1]. However, the global positioning system (GPS) has been used for outdoor positioning in the last few decades, while its positioning accuracy is limited in the indoor environment. Moreover, the GPS signals indoor are also limited so that it is not appropriate for indoor positioning. Generally, the IPS is classified into two types, namely, a radiofrequency-based system and infrared-based system. The radiofrequency-based system utilizes signal strength information at multiple base stations to provide user location services [2]. The infrared-based system uses sensor networks to collect infrared signals and deduce the infrared client's location by checking the location information of different sensors [3]. As the coverage range of infrared-based clients is up to 10 meters while the coverage range of radiofrequency-based clients is

up to 50 meters, radiofrequency has become the most commonly used technique for indoor positioning.

Estimating the indoor position with the radiofrequency technique is also challenging as there are variations of signals due to the motion of the portable unit and dynamics of the changing environment [4]. Moreover, the traditional geometric approach that deduces the location based on the angle and distance estimates from different signal transmitters is problematic as the transmitted signal might be distorted due to reflections and refraction and the indoor environment [5]. Machine learning approaches can avoid the complexity of determining an appropriate propagation model with traditional geometric approaches and adapt well to local variations of indoor environment [6]. Thus, we use machine learning approaches to construct an empirical model that models the distribution of Received Signal Strength (RSS) in an indoor environment. The model can determine the indoor position based on the RSS information in that position. The model-based positioning system involves offline and online phases. The RSS readings from different AP are collected during the offline phase with the machine learning

approach, which captures the indoor environment's complex radiofrequency profile [7]. The model is then trained with the RSS training samples. During the online phase, the client's position is determined by the signal strength and the trained model. Moreover, there is no state-of-the-art work that evaluates the model performance of different algorithms. No guidelines of the size of training samples and the number of AP are provided to train the models.

In this paper, we compare three machine learning models, namely, Support Vector Regression (SVR), Random Forest (RF), and eXtreme Gradient Tree Boosting (XGBoost), with the Gaussian Process Regression (GPR) to find the best model for indoor positioning. Each model is trained with the optimum parameter set obtained from the hyperparameter tuning procedure. Besides, the GPR is trained with three kernels, namely, Radial-Basis Function (RBF) kernel, Matérn kernel, and Rational Quadratic (RQ) kernel, and evaluated with the average error and standard deviation. We used the hyperparameter tuning procedure to tune the parameter for each model and get the optimal parameter set for each model and then compare the performances. The prediction results are evaluated with different sizes of training samples and numbers of AP. Results show that the XGBoost model outperforms all the other models and related work in positioning accuracy. Moreover, the XGBoost model can also achieve high positioning accuracy with smaller training size and fewer APs. We design experiment and use results to show the optimal number of access points and the size of RSS data for the optimal model.

This paper is organized as follows. Section 2 summarizes the related work that constructs models for indoor positioning. Section 3 introduces the background of machine learning approaches as well as the kernel functions for GPR. Sections 4 and 5 describe procedure and experiment result we carried out for the indoor positioning with different approaches. Section 6 concludes the paper and outlines some future work.

2. Related Work

A great deal of previous research has focused on improving the indoor positioning accuracy with machine learning approaches. Brunato evaluated the k-nearest-neighbor approach for indoor positioning with wireless signals from several access points [8], which has an average uncertainty of two meters. Battiti et al. compared the neural network- (NN-) based model and k-nearest-neighbor model to determine the mobile terminal under the wireless LAN environment [9]. Results show that the NN model performs better than the k-nearest-neighbor model and can achieve a standard average of 1.8 meters. Wu et al. compared different kernel functions of the support vector regression to estimate locations with GSM signals [6]. Their results show that the SVR models have better positioning performance compared with NN models. As SVR has the best prediction performance in the current work, we select SVR as a baseline model to evaluate the performance of the other three machine learning approaches and the GPR approach with different kernels.

Besides machine learning approaches, Gaussian process regression has also been applied to improve the indoor positioning accuracy. Schwaighofer et al. built Gaussian process models with the Matérn kernel function to solve the localization problem in cellular networks [5]. Bekkali et al. compared the kernel functions for GPR and developed a location sensing system based on RSS data [7]. Alfakih et al. proposed a Gaussian Mixture Model to approximate the distribution of RSS for indoor localization [10]. Their approach reaches the mean error of 1.6 meters. Less work has been done to compare the GPR with traditional machine learning approaches. Our work assesses the positioning performance of different models and experiments on the size of training samples and the number of APs for the optimum model.

3. Machine Learning Models and Gaussian Process Regression

In the past decade, machine learning played a fundamental role in artificial intelligence areas such as lithology classification, signal processing, and medical image analysis [11–13]. More recently, there has been extensive research on supervised learning to predict or classify some unseen outcomes from some existing patterns. Given a set of data points $\{x_1, x_2, \dots, x_n\}$ associated with set of labels $\{y_1, y_2, \dots, y_n\}$, supervised learning could build a regressor or classifier to predict or classify the unseen y from x . Here each x_i is a feature vector with size n and each y_i is the labeled value. A model h_θ is built with supervised learning for the given input x_i and the predicted value is $h_\theta(x_i)$. The training process of supervised learning is to minimize the difference between predicted value $h_\theta(x_i)$ and the actual value y_i with a loss function $L(h_\theta(x_i), y_i)$. The model performance of supervised learning is usually assessed by $\sum_{i=1}^M L(h_\theta(x_i), y_i)$.

3.1. Support Vector Regression. The support vector machine (SVM) model is usually used to construct hyperplane to separate high-dimensional feature space and distinguish data from different classes [14]. Drucker et al. proposed a support vector regression (SVR) algorithm that applies a soft margin of tolerance ξ in SVM to approximate and predict values [15]. ξ is used to define the soft margin allowed for the model. The weights w of the model are calculated given that model function $h_\theta(x)$ is at most ϵ from the target y ; formally, $\forall i \cdot y_i - w * x_i - b \leq \epsilon + \xi_i \wedge w * x_i + b - y_i \leq \epsilon + \xi_i^*$. In SVR, the goal is to minimize the function in equation (1). Here, C is the penalty parameter of the error term $\sum_{i=1}^n (\xi_i + \xi_i^*)$:

$$\frac{1}{2} \|w\|^2 + C * \sum_{i=1}^n (\xi_i + \xi_i^*). \quad (1)$$

SVR uses a linear hyperplane to separate the data and predict the values. However, in some cases, the distribution of data is nonlinear. Thus, kernel functions map the nonlinear separable feature space to linear separable feature space with kernel functions [16]. Equation (2) shows the

Radial Basis Function (RBF) kernel for the SVR model, where σ defines the standard deviation of the data. $\|x - y\|^2$ defines the squared Euclidean distance between feature vectors x and y :

$$K(x, y) = \exp\left(-\frac{1}{2\sigma^2}\|x - y\|^2\right). \quad (2)$$

3.2. Random Forest. In supervised learning, decision trees are commonly used as classification models to classify data with different features. Classification and Regression Trees (CART) [17] are usually used as algorithms to build the decision tree. However, using one single tree to classify or predict data might cause high variance. Thus, ensemble methods are proposed to construct a set of tree-based classifiers and combine these classifiers' decision with different weighting algorithms [18]. Random Forest (RF) algorithm is one of the ensemble methods that build several regression trees and average the result of the final prediction of each regression tree [19]. Algorithm 1 shows the procedure of the RF algorithm. Given the feature space and its corresponding labels, the RF algorithm takes a random sample from the features and constructs the CART tree with randomly selected features. During the procedure, N trees are built to generate the forest. During the test phase, test data are fed into the forest, and each CART tree in the forest predicts a value $\hat{y}_i$ based on the tree structure. The final output of the forest is the average vote of all the predicted values. During the training process, the number of trees and the trees' parameter are required to be determined to get the best parameter set for the RF model.

3.3. Boosting Approaches. Besides SVR and RF, boosting is also useful in supervised learning to reduce bias and variance of the model by constructing strong models from weak models step by step [20]. In each step, the model's weakness is obtained from the data pattern, and the weak model is then altered to fit the data pattern. In recent years, there has been a greater focus placed upon eXtreme Gradient Tree Boosting (XGBoost) models [21]. Friedman et al. proposed to use gradient descent in the boosting approach to minimize the loss function $\sum_{i=1}^n L(y_i, h_{(x_i)})$ [22] and refined the boosting model with regression trees in [23]. In their approach, the first-order Taylor expansion is used in the loss function to approximate the regression tree learning. In Chen and Guestrin's approach, they proposed to use a higher-order approximation to get a better regression tree structure [21]. The XGBoost algorithm works as Algorithm 2. The model is initialized with a function $f_0(x) = a_0$ which minimizes the loss function $\sum_{i=1}^n L(y_i, a_0)$. In each boosting step, the multipliers p_{ik} and q_{ik} are calculated as first-order Taylor expansion and higher-order Taylor expansion of loss function $L(y_i, f(x_i))$ to calculate the leaf weights which build the regression tree structure. Then the current model $f_k(x)$ is updated with the previous model $f_{k-1}(x)$ with the shrunk base model $\rho a_k h_k(x)$. At last, the

weak models $f_k(x)$ are combined to generate the strong model $f(x)$. In XGBoost, the number of boosting iterations and the structure of regression trees affect the performance of the model. Thus, these parameters are tuned to with cross-validation to get the best XGBoost model.

3.4. Gaussian Process Regression. Gaussian process (GP) is a distribution over functions with a continuous domain, such as time and space [24]. In recent years, Gaussian process has been used in many areas such as image thresholding, spatial data interpolation, and simulation metamodeling. A GP $g(x)$ is usually parameterized by a mean function $\mu(x)$ and a covariance function $K(x, x')$, formalized in equations (3) and (4). Given a set of data points $\{x_1, x_2, \dots, x_n\}$ associated with set of labels $\{y_1, y_2, \dots, y_n\}$, each label y_i can be seen as a Gaussian noise model as in equation (5). Here, $GP((\mu(x), K(x, x')))$ defines the stochastic map for each data point and its label and ε_i defines the measurement noise assumed to satisfy the Gaussian noise with σ standard deviation:

$$\mu(x) = E[g(x)], \quad (3)$$

$$K(x, x') = E[(g(x) - \mu(x))(g(x') - \mu(x')))], \quad (4)$$

$$y_i = g(x_i) + \varepsilon_i, \text{ where } g(x_i) \\ = GP(\mu(x), K(x, x')), \{ \varepsilon \} \sim N(0, \sigma_n^2). \quad (5)$$

Given the training data x with its corresponding labels y as well as the test data x^* with its corresponding labels y^* with the same distribution, then equation (6) is satisfied. Here, $K(x, x)$ is the covariance matrix based on training data points x , $K(x^*, x)$ is the covariance matrix between the test data points and training points, and $K(x^*, x^*)$ is the covariance matrix between test points. Then, the conditional probability of y^* can be formalized as equation (7):

$$\begin{bmatrix} y \\ y^* \end{bmatrix} / x, x^* \sim N\left(\begin{bmatrix} \mu(x) \\ \mu(x^*) \end{bmatrix}, \begin{bmatrix} K(x, x) + \sigma_n^2 I & K(x, x^*)^T \\ K(x^*, x) & K(x^*, x^*) \end{bmatrix}\right), \quad (6)$$

$$p(y^* | y, x, x^*) \sim N(\hat{\mu}, \hat{v}), \quad (7)$$

where

$$\begin{aligned} \hat{\mu} &= K(x, x^*)^T (K(x, x) + \sigma_n^2 I)^{-1} (y - \mu(x)) + \mu(x^*), \\ \hat{v} &= K(x^*, x^*) - K(x, x^*)^T (K(x, x) + \sigma_n^2 I)^{-1} K(x, x^*) + \sigma_n^2 I. \end{aligned} \quad (8)$$

Maximum likelihood estimation (MLE) has been used in statistical models, given the prior knowledge of the data distribution [25]. Thus, given the training data points x with label y , the estimated y^* of target x^* can be calculated by maximizing the joint likelihood $\log(p(y^* | y, x, x^*))$ in equation (7).

- (1) Given: $(x_1, y_1), \dots, (x_n, y_n)$ where $x_i \in X, y_i \in \mathbb{R}$ the number of CART trees: N
- (2) For $n = 1$ to N :
 - (a) Construct a CART tree $\text{CART}(x)$ with randomly selected data $x \in x_i$ with randomly selected features
 - (b) Get the prediction $\hat{y}$ of each CART tree and add the CART tree to the forest $F = F \cup \text{CART}(x)$
- (3) Predict the final result y from the forest: $\hat{y} = \sum_{i=1}^N \hat{y}_i / N$

ALGORITHM 1: Random Forest algorithm.

- (1) Given: $(x_1, y_1), \dots, (x_n, y_n)$ where $x_i \in X, y_i \in \mathbb{R}$
The number of iterations: N
- (2) Initialize $f_0(x) = a_0 = \arg\min \sum_{i=1}^n L(y_i, a_0)$
- (3) For $k = 1$ to N :
 - (a) Compute

$$p_{ik} = -[\partial L(y_i, f(x_i)) / \partial f(x_i)]_{f(x)=f_{k-1}(x)}, \text{ for } i = 1, \dots, n$$

$$q_{ik} = -[\partial^2 L(y_i, f(x_i)) / \partial f(x_i)^2]_{f(x)=f_{k-1}(x)}, \text{ for } i = 1, \dots, n$$
 - (b) Fit base model $h_k(x)$

$$h_k(x) = \sum_{j=1}^k b_{jk} I(x \in R_{jk})$$
 - (c) Determine the leaf weight for the learnt structure with p_{ik} and q_{ik}
 - (d) Update current model $f_k(x)$ with previous model $f_{k-1}(x)$ and the constrained $\rho a_k h_k(x)$

$$f_k(x) = f_{k-1}(x) + \rho a_k h_k(x) (x \in R_{jk})$$
- (4) Calculate the final boosting model $f(x) = \sum_{k=0}^N f_k(x)$

ALGORITHM 2: XGBoost algorithm, modified from [21].

In GPR, covariance functions are also essential for the performance of GPR models. This paper mainly evaluates three covariance functions, namely, Radial Basis Function (RBF) kernel, Matérn kernel, and Rational Quadratic kernel. The RBF kernel is a stationary kernel parameterized by a scale parameter l that defines the covariance function's length scale. Equation (2) shows the kernel function for the RBF kernel. The Matérn kernel adds parameter ν that controls the resulting function's smoothness, which is given in equation (9). Equation (10) shows the Rational Quadratic kernel, which can be seen as a mixture of RBF kernels with different length scales. In the equation, the α parameter controls the mixture of the length scales:

$$K(x_i, x_j) = \sigma^2 \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(\gamma \sqrt{2\nu} \left(\frac{x_i - x_j}{l} \right) \right)^\nu \cdot K_\nu \left(\gamma \sqrt{2\nu} \left(\frac{x_i - x_j}{l} \right) \right), \quad (9)$$

$$K(x_i, x_j) = \left(1 + \frac{(x_i - x_j)^2}{2\alpha l^2} \right)^{-\alpha}. \quad (10)$$

4. Experiment with Offline Training

In this paper, we use the RSS-based modeling technique that explores the relationship between the specific location and its corresponding RSS. Figure 1 shows the procedure that

builds indoor positioning model by comparing the performance of different machine learning models. In the offline phase, RSS data from several APs are collected as the training data set. There are two procedures to train the offline RSS-based model. In the first step, cross-validation (CV) is used to test whether the model is suitable for the given machine learning model. The CV can be used for feature selection and hyperparameter tuning. By using the 5-fold CV, the training data is split into fivefold. During the training process, the model is trained with the four folds of data and test with the left fold of data. The training procedure is repeated five times to calculate the average accuracy of the model with the specific parameter. After we get the model with the optimum parameter set, the second step of the offline phase trains the model with the RSS data. Then, we got the final model that maps the RSS to its corresponding position in the building. Later in the online phase, we can use the generated model for indoor positioning.

4.1. Data Collection. To construct the fingerprinting database and evaluate the machine learning models, we collect RSS data in an indoor environment whose floor plan is shown in Figure 2. In the building, we place 7 APs represented as red pentagram on the floor with an area of $21.6 \text{ M} \times 15.6 \text{ m}$. The RSS measurements are taken at each point in a grid of 0.6 m spacing between each other. The RSS data are measured in dBm, which has typical negative values ranging between 0 dBm and -110 dBm . The RSS data of seven APs are taken as seven features. The output is the

FIGURE 1: Indoor positioning modeling procedure with offline phase and online phase.

FIGURE 2: Indoor floor plan with access points marked by red pentagram.

coordinates of the location on the two-dimensional floor. We write Android applications to collect RSS data at reference points within the test area marked by the seven APs, whereas the RSS comes from the Nighthawk R7000P commercial router. During the field test, we collect 799 RSS data as the training set. The training set's size could be adjusted accordingly based on the model performance, which would be discussed in the following section. The 200 RSS data are collected during the day with people moving or environment changes, which are used to evaluate the model performance.

4.2. Hyperparameter Tuning. As is shown in Section 2, the machine learning models require hyperparameter tuning to get the best model that fits the data. Table 1 shows the parameters requiring tuning for each machine learning model. Tuning is a process that uses a performance matrix to rank the regressors with different parameters to optimize a parameter for each specific model [11]. In this paper, we use the distance error as the performance matrix to tune the parameters. Given the predicted coordinates of the location as $(\hat{x}, \hat{y})$ and the true coordinates of the location as (x, y) , the Euclidean distance error is calculated as follows:

$$d = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2}. \quad (11)$$

Underfitting and overfitting often affect model performance. In this paper, we use the validation curve with 5-fold cross-validation to show the balanced trade-off between the bias and variance of the model. In the validation curve, the training score is higher than the validation score as the model will be a better fit to the training data than test data. The increasing of the validation scores indicates that the model is underfitting. When the validation score decreases, the model is overfitting. Thus, validation curves can be used to select the best parameter of a model from a range of values. Results show that nonlinear models have better prediction accuracy compared with linear models, which is evident as the distribution of RSS over distance is not linear. Table 1 shows the optimal parameter settings for each model, which we use to train different models.

Figure 3 shows the tuning process that determines the optimum value for the penalty parameter C and kernel coefficient parameter γ for the SVR with RBF and linear kernels. Results show that RBF has better prediction accuracy compared with linear kernels in SVR. It is evident, as the distribution of RSS over distance is not linear. Thus, linear models cannot describe the model correctly. Moreover, the selection of coefficient parameter γ of the SVR with RBF kernel is critical to the performance of the model. The validation curve shows that when γ is 0.01, the SVR has the best performance in predicting the position.

TABLE 1: Hyperparameter tuning for different machine learning models.

Machine learning model	Parameter	Optimal setting	Distance error
SVR with linear kernel	Penalty C	2	4.58
	Kernel coefficient	0.01	
SVR with RBF kernel	Penalty C	2	1.75
	Kernel coefficient	0.01	
Random Forest	Number of trees	400	0.93
	Max depth of individual tree	10	
	Min samples split of individual tree	8	
	Min samples leaf of individual tree	4	
XGBoost	Number of boosting iterations	500	0.85
	Learning rate	0.25	
	Max depth of individual tree	15	
	Subsample	0.9

FIGURE 3: Hyperparameter tuning for SVR with linear and RBF kernel. (a) γ of RBF kernel. (b) C of RBF kernel. (c) γ of linear kernel. (d) C of linear kernel.

Figure 4 shows the tuning process that calculates the optimum value for the number of trees in the random forest as well as the tree structure of the individual tree in the forest. The validation curve shows that the maximum depth of the tree might affect the performance of the RF model. When the maximum depth of the individual tree reaches 10,

the model comes to the best performance. The number of boosting iterations and other parameters concerning the tree structure do not affect the prediction accuracy a lot.

Figure 5 shows the tuning process that calculates the optimum value for the number of boosting iterations and the learning rate for the AdaBoost model. Results show that a

FIGURE 4: Hyperparameter tuning for Random Forest model. (a) Number of estimators. (b) Max depth. (c) Min samples split. (d) Min samples leaf.

higher learning rate would lead to better model performance. While the number of iterations has little impact on prediction accuracy, 300 could be used as the number of boosting iterations to train the model to reduce the training time.

Figure 6 shows the tuning process that calculates the optimum value for the number of boosting iterations, the learning rate, and the individual tree structure for the XGBoost model. To avoid overfitting, we also tune the *subsample* parameter that controls the ratio of training data before growing trees. The validation curve suggests that a higher learning rate and the number of boosting iterations could have a better model performance. The individual tree structure and the ratio of training data have less impact on prediction accuracy.

4.3. Kernel Selection for Gaussian Process Regression. Besides the typical machine learning models, we also analyze the GPR with different kernels for the indoor positioning

problem. Table 2 shows the distance error with a confidence interval for different kernels with length scale bounds. In statistics, 1.96 is used in the constructing of 95% confidence intervals [26]. We calculate the confidence interval by multiplying the standard deviation with 1.96. Overall, the three kernels have similar distance errors. However, the confidence interval has a huge difference between the three kernels. The RBF and Matérn kernel have the 4.4 m and 8.74 m confidence interval with 95% accuracy while the Rational Quadratic kernel has the 0.72 m confidence interval with 95% accuracy. Rational Quadratic kernel is the most stable model for the GPR algorithm. Thus, we select this as the kernel of the GPR model to compare with other machine learning models.

5. Model Evaluation and Experiment Results

In the previous section, we train the machine learning models with the 799 RSS samples. In this section, we evaluate

FIGURE 5: Hyperparameter tuning for AdaBoost model. (a) Number of estimators. (b) Learning rate.

FIGURE 6: Hyperparameter tuning for XGBoost model. (a) Number of estimators. (b) Max depth. (c) Subsample. (d) Learning rate.

TABLE 2: Distance error with confidence interval for different Gaussian progress regression kernels.

Kernel	Length scale bounds	Distance error	Confidence interval with 95% probability
RBF	0.1–10	0.865	4.4
Rational Quadratic	0.1–10	0.864	0.72
Matérn	1	0.865	8.74
Hline

FIGURE 7: Features that affect model performance of indoor positioning. (a) Impact of the number of RSS samples. (b) Impact of the number of access points.

the result by evaluating the performance of the models with 200 collected RSS samples with location coordinates. Tables 1 and 2 show the distance error of different machine learning models. Overall, the GPR with Rational Quadratic kernel has the lowest distance error among all the GP models, and XGBoost has the lowest distance error compared with other machine learning models. Specifically, XGBoost model achieves a 0.85 m error, which is better than the RF model. XGBoost also outperforms the SVR with RBF kernel. However, the XGBoost and the GPR with Rational Quadratic have similar performance concerning the distance error. In this section, we evaluate the impact of the size of training samples and the number of APs to get the model with high indoor positioning accuracy but requires fewer resources such as training samples and the number of APs.

5.1. Size of Training Samples. Figure 7(a) shows the impact of the training sample size on different machine learning models. During the training process, we restrict the training size from 400 to 799 and evaluate the distance error of different trained machine learning models. Results reveal that there has been a gradual decrease in distance error with the increasing of the training size for all machine learning models. In all stages, XGBoost has the lowest distance error compared with all the other models. The RF model has a similar performance with a slightly higher distance error. With the increase of the training size, GPR gets the better

performance, while its performance is still slightly weaker compared with the XGBoost model.

5.2. Number of Access Points. Figure 7(b) reveals the impact of the size of APs on different machine learning models. In the training process, we use the RSS collected from different APs as features to train the model. The size of the APs determines the size of the features. Results show that the distance error decreases gradually for the SVR model. The graph also shows that there has been a sharp drop in the distance error in the first three APs for XGBoost, RF, and GPR models. Then the distance error of the three models comes to a steady stage. This trend indicates that only three APs are required to determine the indoor position. More APs are not helpful as the indoor positioning accuracy is not improving with more APs. Overall, XGBoost still has the best performance among RF and GPR models.

6. Conclusion and Future Work

In this paper, we evaluate different machine learning approaches for indoor positioning with RSS data. The models include SVR, RF, XGBoost, and GPR with three different kernels. Hyperparameter tuning is used to select the optimum parameter set for each model. Then the performance of different models is evaluated using the Euclidean distance error between the predicted coordinates and real

coordinates. Results show that XGBoost has the best performance compared with all the other machine learning models. Also, 600 is enough for the RSS training size as the distance error does not change dramatically after the training size reaches 600. Results also reveal that 3 APs are enough for indoor positioning as the distance error does not decrease with more APs.

Indoor position estimation is usually challenging for robots with only built-in sensors. Accumulated errors could be introduced into the localization process when the robot moves around. However, based on our proposed XGBoost model with RSS signals, the robot can predict the exact position without the accumulated error. Thus, more work can be done to decrease the positioning error by using the extended Kalman filter localization algorithm to fuse the built-in sensor data and the RSS data.

Data Availability

The data are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the work.

References

- [1] A. Serra, D. Carboni, and V. Marotto, "Indoor pedestrian navigation system using a modern smartphone," in *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, pp. 397–398, ACM, San Francisco, CA, USA, 2010.
- [2] P. Bahl, V. N. Padmanabhan, V. Bahl, and V. Padmanabhan, "Radar: an in-building rf-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000*, Tel Aviv, Israel, 2000.
- [3] A. Harter and A. Hopper, "A distributed location system for the active office," *IEEE Network*, vol. 8, no. 1, pp. 62–70, 1994.
- [4] H. Hashemi, "The indoor radio propagation channel," *Proceedings of the IEEE*, vol. 81, no. 7, pp. 943–968, 1993.
- [5] A. Schaighofer, M. Grigorov, V. Tresp, and C. Hoffmann, "Ggps: a Gaussian process positioning system for cellular networks," *Advances in Neural Information Processing Systems*, pp. 579–586, 2004.
- [6] Z. L. Wu, C. H. Li, J. K. Y. Ng, and K. R. Leung, "Location estimation via support vector regression," *IEEE Transactions on Mobile Computing*, vol. 6, no. 3, pp. 311–321, 2007.
- [7] A. Bekkali, T. Masuo, T. Tominaga, N. Nakamoto, and H. Ban, "Gaussian processes for learning-based indoor localization," in *Proceedings of the 2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pp. 1–6, IEEE, Xi'an, China, 2011.
- [8] M. Brunato and C. Kiss Kallo, "Transparent location fingerprinting for wireless services," Technical Report, University of Trento, Trento, Italy, 2002.
- [9] R. Battiti, A. Villani, and T. Le Nhat, "Neural network models for intelligent networks: deriving the location from signal patterns," in *Proceedings of AINS*, Kennesaw, GA, USA, 2000.
- [10] M. Alfakih, M. Keche, and H. Benoudnine, "Gaussian mixture modeling for indoor positioning wifi systems," in *Proceedings of the 2015 3rd International Conference on Control*, pp. 1–5, IEEE, Tlemcen, Algeria, 2015.
- [11] Y. Xie, C. Zhu, W. Zhou, Z. Li, X. Liu, and M. Tu, "Evaluation of machine learning methods for formation lithology identification: a comparison of tuning processes and model performances," *Journal of Petroleum Science and Engineering*, vol. 160, pp. 182–193, 2018.
- [12] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2017.
- [13] G. Litjens, T. Kooi, B. E. Bejnordi et al., "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017.
- [14] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [15] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," *Advances in Neural Information Processing Systems*, pp. 155–161, 1997.
- [16] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear svm," *Journal of Machine Learning Research*, vol. 11, pp. 1471–1490, 2010.
- [17] L. Breiman, *Classification and Regression Trees*, Routledge, Abingdon, UK, 2017.
- [18] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the International workshop on multiple classifier systems*, pp. 1–15, Springer, Cagliari, Italy, 2000.
- [19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [20] R. E. Schapire, "The boosting approach to machine learning: an overview," in *Nonlinear Estimation and Classification*, pp. 149–171, Springer, Berlin, Germany, 2003.
- [21] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, New York, NY, USA, 2016.
- [22] J. H. Friedman, "machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [23] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.
- [24] C. E. Rasmussen, *Gaussian Processes in Machine Learning in Summer School on Machine Learning*, pp. 63–71, Springer, Berlin, Germany, 2003.
- [25] R. J. Rossi, *Mathematical Statistics: An Introduction to Likelihood Based Inference*, John Wiley & Sons, Hoboken, NJ, USA, 2018.
- [26] L. J. Savage, *The Foundations of Statistics*, Courier Corporation, North Chelmsford, MA, USA, 1972.