Distributed Learning in Mobile Sensor Networks Using Cross Validation

Songhwai Oh and Jongeun Choi

Abstract— **Mobile sensor networks can increase sensing coverage both in space and time and robustness against dynamic changes in the environment, compared to stationary wireless sensor networks. For operations in a dynamic or unknown environment, mobile sensors need the capability of learning a suitable model during its operations. However, due to the limited communication bandwidth, it is prohibited to share all measurements with other mobile sensors. In this paper, we propose an efficient distributed learning algorithm based on cross validation for mobile sensor networks, which takes the advantage of a multi-agent system and minimizes the communication overhead while achieving excellent performance, and demonstrate its performance in simulation.**

I. Introduction

Mobile sensor networks exhibit superior adaptability and high-resolution sampling capability when compared to stationary wireless sensor networks [1] and they are able to operate under a dynamic environment [2] and unstructured or harsh environments [3]. But, for mobile sensor networks to work under dynamic or unstructured environments, the ability to understand and learn from complex environments is of paramount importance. We can model such complex environments using a nonparametric estimation method, e.g., Gaussian processes (GPs) and support vector machines (SVMs), since they show better expressive and generalizability than parametric estimation methods. Nonparametric estimation methods have shown to be very effective in many real world applications. In particular, Gaussian processes have been used to model complex physical phenomena such as nonstationary geostatistical data [4], wireless signal strength [5], indoor temperature field [6], and terrain mapping [7]. In this paper, we consider the problem of modeling and learning of a complex environment using Gaussian processes.

It is well-known that the performance of the Gaussian process regression algorithm is heavily dependent on the choice of its kernel function [8]. Once a correct kernel function is chosen, the estimation step can be performed (relatively) easily. However, when a mobile sensor network is required to operate under a dynamic or unknown environment, we cannot assume that each agent has a suitable kernel function for its mission in advance. When faced with a new environment, each agent needs to learn a kernel function from collected data while performing its task.

The optimal learning approach for a multi-agent system is to share all data with all agents. But, this is a very costly option for a multi-agent system communicating over wireless channel. Instead, we propose an efficient distributed learning algorithm for mobile sensor networks, which takes the advantage of a multi-agent system and minimizes the communication overhead while achieving excellent performance based on a technique from statistics, called *cross validation* [9], [10].

Cross validation has been widely used in statistics and machine learning to compare the performances of different models (or learning algorithms) by estimating the predictive ability of each statistical model from empirically computed each model's generalization error. Cross validation has been very successful in practice and also enjoys strong theoretical results on its consistency for both classification [11] and regression problems [12].

In this paper, we show that cross validation can be naturally applied to learn models in mobile sensor networks with a limited communication bandwidth. Each agent learns model parameters based on its own measurements. Instead of transmitting all measurements to other agents, each agent transmits the learned model parameters to other agents. Upon receiving model parameters from other agents, each agent applies those model parameters to its own measurements to compute the fitness of each model (i.e., cross validation). Then agents share the fitness results and the model with the best fitness is chosen.

The remainder of this paper is structured as follows. We discuss mobile sensor networks in Section II and Gaussian processes in Section III. The distributed learning algorithm for mobile sensor networks using cross validation is described in Section IV. The performance of the distributed learning algorithm is demonstrated in Section V.

II. Mobile Sensor Networks

First, we explain the mobile sensing network and sensor models used in this paper. Let n be the number of sensing agents distributed over the surveillance region $Q \subset \mathbb{R}^2$. Assume that Q is a convex and compact set. The identity of each agent is indexed by $\mathcal{I} := \{1, 2, \cdots, n\}$. Let $q_i(t) \in Q$ be the location of agent i at time $t \in \mathbb{Z}_{\geq 0}$ and let $q := \text{col}(q_1, q_2, \cdots, q_n) \in \mathbb{R}^{2n}$ be the configuration of the multi-agent system. The discrete time, high-level dynamics of agent i is modeled by

$$\begin{aligned} q_i(t+1) &= q_i(t) + \epsilon p_i(t), \\ p_i(t+1) &= p_i(t) + \epsilon u_i(t), \end{aligned} \tag{1}$$

This work has been supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2010-0013354) and the Research Settlement Fund for the new faculty of SNU.

Songhwai Oh is with the School of Electrical Engineering and Computer Science and ASRI, Seoul National University, Seoul 151-744, Korea. songhwai@snu.ac.kr.

Jongeun Choi is with the Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824-1226, USA. jchoi@egr.msu.edu.

where $q_i, p_i, u_i \in \mathbb{R}^2$ are, respectively, the position, the velocity, and the input of the mobile agent and ϵ is the iteration step size (or sampling time). We assume that the measurement $y(q_i(t), t)$ of sensor i includes the scalar value of the Gaussian process $z(q_i(t), t)$ and sensor noise $w(t)$, at its position $q_i(t)$ and measurement time t,

$$y(q_i(t), t) := z(q_i(t), t) + w(t). \tag{2}$$

A. Graph-Theoretic Representation

The interactions among mobile agents are represented as a graph. Let $G(q) := (\mathcal{I}, \mathcal{E}(q))$ be a communication graph such that an edge $(i, j) \in \mathcal{E}(q)$ if agent i can communicate with agent $j \neq i$. We assume that each agent can communicate with its neighboring agents within a limited transmission range given by a radius of r. We define the neighborhood of agent i in a configuration of q by $N(i, q) := \{j : (i, j) \in \mathcal{E}(q),\ i \in \mathcal{I}\}$. The adjacency matrix $A := [a_{ij}]$ of an undirected graph G is a symmetric matrix such that $a_{ij} = k_3 > 0$ if vertex i and vertex j are neighbors and $a_{ij} = 0$ otherwise, where k_3 is a positive scalar. The scalar graph Laplacian $L = [l_{ij}] \in \mathbb{R}^{n\times n}$ is a matrix defined as $L := D(A) - A$, where $D(A)$ is a diagonal matrix whose diagonal entries are row sums of A, i.e., $D(A) := \text{diag}(\sum_{j=1}^{n} a_{ij})$. The 2-dimensional graph Laplacian is defined as $\hat{L} := L \otimes I_2$, where $\otimes$ is the Kronecker product. A quadratic disagreement function can be obtained via the Laplacian $\hat{L}$ [13]:

$$\Psi_G(p) := p^T \hat{L} p = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}(q)} a_{ij} \|p_j - p_i\|^2, \tag{3}$$

where $p := \text{col}(p_1, p_2, \cdots, p_n) \in \mathbb{R}^{2n}$.

B. Swarming Behavior

In order for agents to sample measurements of a scalar field at spatially distributed locations simultaneously, a group of mobile agents will be coordinated by a flocking algorithm [14]. We use attractive and repulsive smooth potentials similar to those used in [13], [15] to generate a swarming behavior. To enforce a group of agents to satisfy a set of algebraic constraints $\|q_i - q_j\| = d$ for all $j \in N(i, q)$, we introduce a collective potential

$$U_1(q) := \sum_i \sum_{j\neq i} U_{ij}(\|q_i - q_j\|^2) = \sum_i \sum_{j\neq i} U_{ij}(r_{ij}), \tag{4}$$

where $r_{ij} := \|q_i - q_j\|^2$. U_{ij} in (4) is defined by

$$U_{ij}(r_{ij}) := \frac{1}{2}\left(\log(\alpha + r_{ij}) + \frac{\alpha + d^2}{\alpha + r_{ij}}\right), \text{ if } r_{ij} < d_0^2, \tag{5}$$

otherwise (i.e., $r_{ij} \geq d_0^2$), it is defined according to the gradient of the potential, which will be described shortly. Here $\alpha, d \in \mathbb{R}_{>0}$ and $d < d_0$. The gradient of the potential with respect to q_i for agent i is given by

$$\nabla_s U_1|_{s=q_i} := \frac{\partial U_1(q)}{\partial \tilde{q}_i}\Big|_{\tilde{q}_i=q_i} = \sum_{j\neq i} \frac{\partial U_{ij}(r)}{\partial r}\Big|_{r=r_{ij}} 2(q_i - q_j)$$

$$= \begin{cases} \sum_{j\neq i} \frac{(r_{ij}-d^2)(q_i-q_j)}{(\alpha+r_{ij})^2} & \text{if } r_{ij} < d_0^2 \\ \sum_{j\neq i} \rho\left(\frac{\sqrt{r_{ij}}-d_0}{|d_1-d_0|}\right) \frac{\|d_0^2-d^2\|}{(\alpha+d_0^2)^2}(q_i - q_j) & \text{otherwise,} \end{cases} \tag{6}$$

where $\rho : \mathbb{R}_{>0} \to [0, 1]$ is the bump function that smoothly varies from 1 to 0 as the scalar input increases [13]. In equations (4), (5), and (6), α was introduced to prevent the reaction force from diverging at $r_{ij} = \|q_i - q_j\|^2 = 0$. This potential yields a reaction force that is attracting when the agents are too far and repelling when a pair of two agents are too close. It has an equilibrium point at a distance of d. We also introduce a potential U_2 to model the environment. U_2 enforces each agent to stay inside the closed and connected surveillance region Q and prevents collisions with obstacles in Q. Define the total artificial potential by

$$U(q) := k_1 U_1(q) + k_2 U_2(q), \tag{7}$$

where $k_1 > 0$ and $k_2 > 0$ are weighting factors.

We use the distributed control law developed in [14], where distributed control for agent i is decided based on gradients of the potential fields. More precisely, the input to the dynamic model (1) is

$$u_i(t) = -\nabla U(q_i(t)) - k_{di} p_i(t) - \nabla \Psi_G(p_i(t)), \tag{8}$$

where we use $\nabla g(x)$ to represent $\nabla_s g(s)|_{s=x}$ and k_{di} is a gain for the velocity feedback. For more detail, see [14].

III. Gaussian Processes

A Gaussian process[1] defines a distribution over a space of functions and it is completely specified by its mean function and covariance function. Let $X \in \mathcal{D} \subseteq \mathbb{R}^d$ denote the index vector. If $z(X) \in \mathbb{R}$ is a Gaussian process, it can be written as

$$z(X) \sim \mathcal{GP}(\mu(X), \mathcal{K}(X, X')), \tag{9}$$

where $\mu(X) = \mathbb{E}(z(X))$ is a mean function and $\mathcal{K}(X, X') = \mathbb{E}\left[(z(X) - \mu(X))(z(X') - \mu(X'))\right]$ is a covariance function of $z(X)$. From noisy measurements $Y = [y_1, y_2, \ldots, y_n]^T \in \mathbb{R}^n$, at $X_1, X_2, \cdots, X_n \in \mathcal{D}$, where $y_i = z(X_i) + w_i$ and w_i is a white Gaussian noise with variance σ_w^2, we can predict the value of the Gaussian process at an unobserved location X_* and the predicted value has a Gaussian distribution $z(X_*) \sim \mathcal{N}(\hat{z}(X_*), \mathbf{var}(z(X_*)))$, where

$$\begin{aligned} \hat{z}(X_*) &= k_*^T (\Sigma + \sigma_w^2 I)^{-1} Y \\ \mathbf{var}(z(X_*)) &= \mathcal{K}(X_*, X_*) - k_*^T (\Sigma + \sigma_w^2 I)^{-1} k_* \end{aligned}$$

with $k_* = [\mathcal{K}(X_1, X_*), \ldots, \mathcal{K}(X_n, X_*)]^T$ [8]. The predicted variance $\mathbf{var}(z(X_*))$ measures the uncertainty we have about the new location X_* and can be used in mobile sensor networks for exploration, e.g. [16].

[1] A *Gaussian process* is a collection of random variables, any finite number of which have a joint Gaussian distribution [8].

A popular choice for a covariance function is the squared exponential kernel function

$$\mathcal{K}(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2\sigma_l^2}\sum_{m=1}^{n}(x_m - x'_m)^2\right), \qquad (10)$$

where σ_f and σ_l are hyperparameters of the kernel, which can be learned from data [8]. Learning in Gaussian processes implies (1) choosing the right kernel function from a set of candidates, and (2) estimating hyperparameters of the chosen kernel function.

IV. Distributed Learning Using Cross Validation

Cross validation is a widely used empirical technique for estimation and model selection [9], [10]. When there is a number of candidate models, cross validation estimates the predictive ability of each statistical model by empirically computing each model's generalization error. In k-fold cross validation, we partition the data set into k folds of equal size. Then we run k experiments. For experiment i, we train our model using all the data except the i-th fold and test on the i-th fold of data. We finally choose the model which performs best on the test set since this model minimizes the generalization error most. The performance of a model can be measured by the likelihood or classification error on the test set, which empirically approximates the generalizability of each model. For classification problems, k-fold estimate is strictly more accurate than a single hold-out estimate [11]. For regression problems (both parametric and nonparametric), cross validation is consistent in the sense of selecting the better model (procedure) with probability approaching one [12].

The overall architecture of distributed learning algorithm using cross validation designed for mobile sensor networks is shown in Figure 1 and the distributed CV learning algorithm is given in Algorithm 1. Each agent collects data from its own sensors and learns the model, for example, hyperparameters of the kernel function or a kernel from a set of possible kernels function and its associated hyperparameters . Then agents communicate with each other about the learned models over wireless communication channel. Once model candidates from the other agents are received, each agent tests the fitness of each model on its own data (i.e., cross validation) and advertises each model's fitness to other agents. Agents then choose the model that performs the best in the model fitness test.

The main difference between the distributed CV learning algorithm and the traditional k-fold cross validation algorithm is that, under distributed CV learning, a model is trained using a single fold of data collected by an agent. Assuming each agent collects the same number of measurements m/n, where m be the total number of available measurements from all agents and n is the number of agents. Each agent trains a model using its own data of size m/n and the trained model is tested by other agents using their own data. This scheme is designed to avoid transmissions of measurements. By transmitting only model information, not the data, we can greatly reduce the communication load

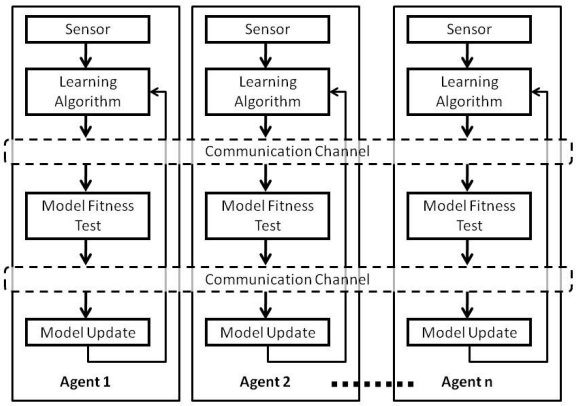


Fig. 1. An architecture of distributed learning for mobile sensor networks.

of the network and speed up the overall learning process by learning different models in parallel. When several models are considered, the parameter set $\hat{\theta}_i$ can also contain the model index information in order to solve the model selection problem (e.g., Section V-C).

Algorithm 1 Distributed CV Learning (agent i at time t)

Input: $y_i(1), \ldots, y_i(t)$ (data), $q_i(1), \ldots, q_i(t)$ (locations), N_i (neighbors of agent i and $i \notin N_i$)
Output: $\hat{\theta}_i$ (estimated parameters)

Learn parameters $\bar{\theta}_i$ from data $y_i(1), \ldots, y_i(t)$
Transmit $\bar{\theta}_i$ to N_i
Receive $\{\bar{\theta}_j\}$ from neighbors ($j \in N_i$)
for all $j \in N_i$ **do**
  Test $\bar{\theta}_j$ on $y_i(1), \ldots, y_i(t)$
  Compute the fitness v_{ji} of $\bar{\theta}_j$ (e.g., likelihood)
  Transmit v_{ji} to N_i
end for
Receive $\{v_{jk}\}$ from N_i
for all $j \in \{m : v_{mk}$ is received$\}$ **do**
  $\mathcal{I} := \{k : v_{jk}$ is received$\}$
  $v_j \leftarrow \frac{1}{|\mathcal{I}|}\sum_{k\in\mathcal{I}} v_{jk}$
end for
$j_* \leftarrow \arg\max v_j$
$\hat{\theta}_i \leftarrow \bar{\theta}_{j_*}$

V. Simulation Results

In order to study the performance of our distributed CV learning algorithm, we consider the problem of estimating hyperparameters of the squared exponential kernel function (10) and the variance of the measurement noise in (2). Hence, our goal is to estimate σ_f^2, σ_l^2, and σ_w^2. The application of the algorithm to the model selection problem is shown in Section V-C.

We generated five scenarios from a Gaussian process (9) with the kernel function (10), where $\sigma_f^2 = 1.0$ and $\sigma_l^2 = 2.0$. An example is shown in Figure 2.

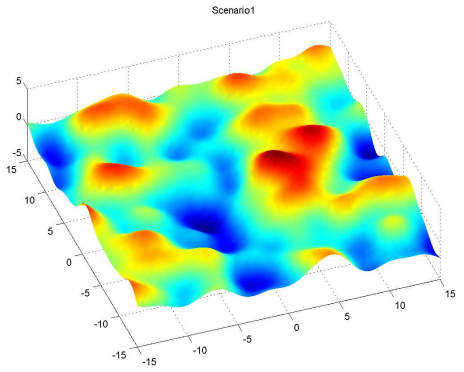


Fig. 2. An example of scenarios used in Section V-A and V-B. The figure shows the values of $z(s)$ for $s \in [-15, 15]^2$.

A. Comparison Against Independent Learning Agents

We first compared the performance of our distributed CV learning algorithm against the case in which agents learn independently. We assumed that all agents can communicate with each other (the next section discusses the effect of the communication range on learning) and moves randomly over the surveillance region, i.e., dynamic model (1) with random inputs.

For each scenario, we generated 10 independent runs (trajectories of agents). The standard deviation of the measurement noise was 0.2. As agents collect noisy measurements about the field, we performed two learning operations: (1) independent learning (each agent learns from its own data), and (2) distributed CV learning (agents learn using Algorithm 1). We used the optimization routine from the Gaussian process MATLAB toolbox [8] for estimating hyperparameters. At each simulation time, each agent uses all of its past measurements for this estimation. Then we computed the mean square errors of the estimated parameters against the true parameter values.

The simulation results are shown in Figure 3. The top row shows results from independent learning of a 10-agent system, the middle row shows results from distributed CV learning of a 10-agent system, and the bottom row shows results from distributed CV learning of a 20-agent system. While independent learning takes more time to correctly estimate parameters and its estimates are unstable, the distributed CV learning method shows faster convergence to the true values and its estimates are highly stable. It is important to note that, independent learning does not take an advantage of a multi-agent system; its performance does not improve as more mobile sensors are present. But we can see the benefit of adding more agents in distributed CV learning, i.e., the number of overshootings at the beginning of simulation is reduced.

B. Effects of the Communication Range

In the previous section, we assumed that all agents can communicate with each other. However, this is unrealistic under wireless communication. In this section, we assume the each agent can communicate with agents that are within the communication range of radius r. Then we followed the same steps as the previous section in our simulation. There are 10 sensor nodes and all perform distributed CV learning to estimate parameters. We compared two motion types at different communication ranges: (1) random motion; and (2) flocking-based motion described in Section II.

For the random motion, the dynamic model (1) is used but random inputs are given to the agents. However, we applied the boundary potential field $U_2(q)$ to make sure all agents stay within the surveillance region. For the flocking-based motion (or swarming motion), we used the dynamic model (1) with potential fields described in Section II-B to make sure that agents move in coordination, avoid obstacles, and maintain the communication distance with other agents. The input (8) is applied to all agents except agent 1 (leader), which moves based on the random motion. Hence, mobile sensors follow the leader in our setup. The following parameters are used for flocking: $\epsilon = 0.75$, $d = 1.50$, $d_0 = 2.43$, $\alpha = 0.1$, $k_1 = 0.1$, and $k_2 = 0.1 \times n$, where n is the number of agents.

Figure 4 shows the results. When $r = 0$, i.e., agents cannot communicate with each other, the performance of the flocking-based motion is similar to the random motion. But when agents moves based on flocking, the average number of neighbors increases with an increased communication range. Hence, we can take advantage of a multi-agent system and the performance of distributed CV learning improves. This example also demonstrates the benefits of flocking or swarming of mobile sensor networks in the presence of communication constraints.

C. Model Selection Using Distributed CV Learning

Next, we apply the distributed CV learning algorithm to the model selection problem and highlight the power of distributed CV learning. For an illustration purpose, we consider a simple model selection problem with two competing models.

Model 1: A Gaussian process with the squared exponential kernel function shown in (10).

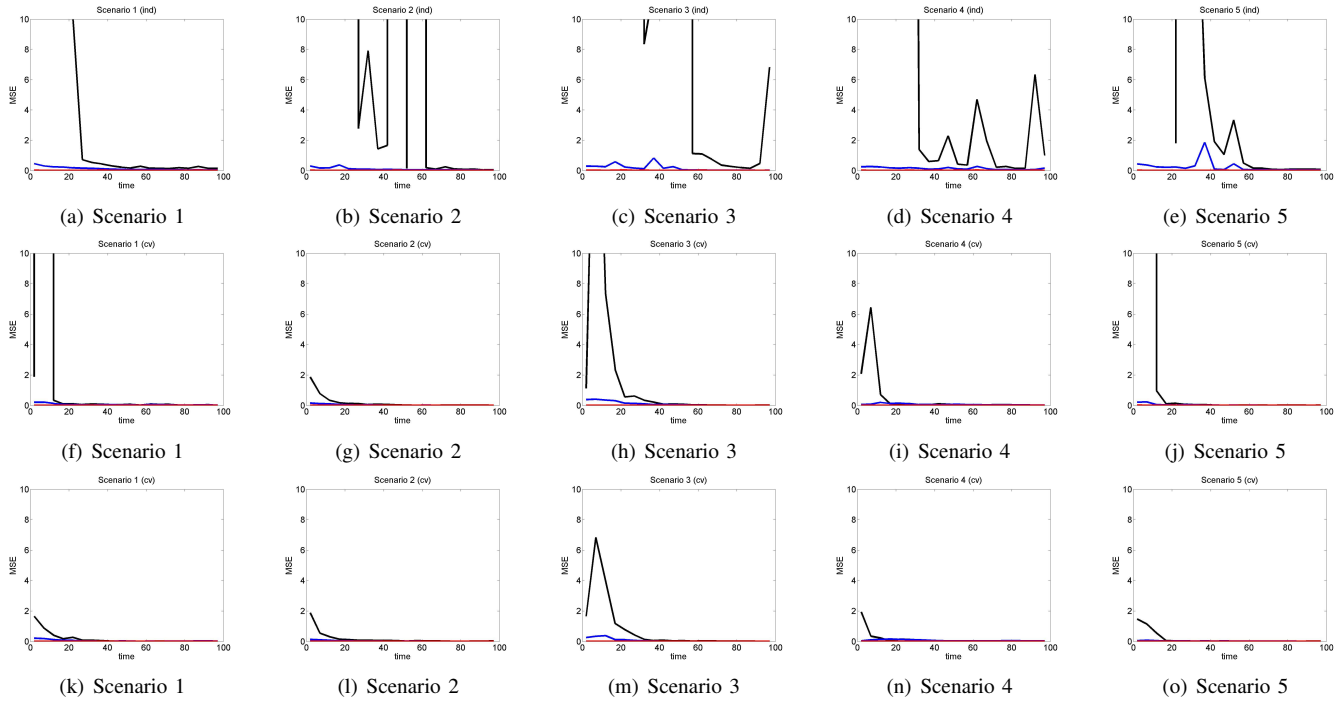
Model 2: A Gaussian process with a Matérn kernel function with parameter $\nu = 3/2$ [8]:

$$\mathcal{K}(x, x') = \sigma_{mf}^2 \left(1 + \sqrt{3}d(x, x')\right) \exp\left(-\sqrt{3}d(x, x')\right),$$

where $d(x, x') = \left(\frac{1}{\sigma_{ml}^2} \sum_{i=1}^{d} (x_i - x_i')^2\right)^{1/2}$.

The true hyperparameter values are set as the previous example for Model 1. For Model 2, we used $\sigma_{mf}^2 = 1.0$ and $\sigma_{ml}^2 = 1.0$. The configuration of the surveillance region is the same as the previous example. There are two agents moving around the surveillance region in coordination for 300 simulation times. We considered two cases: (Case 1) the true model is Model 1 and (Case 2) the true model is Model 2. For both cases, agent 1 learns Model 1 while agent 2 learns Model 2.

Figure 5 shows results from Case 1 and Figure 6 shows results from Case 2. From the learned parameters of both

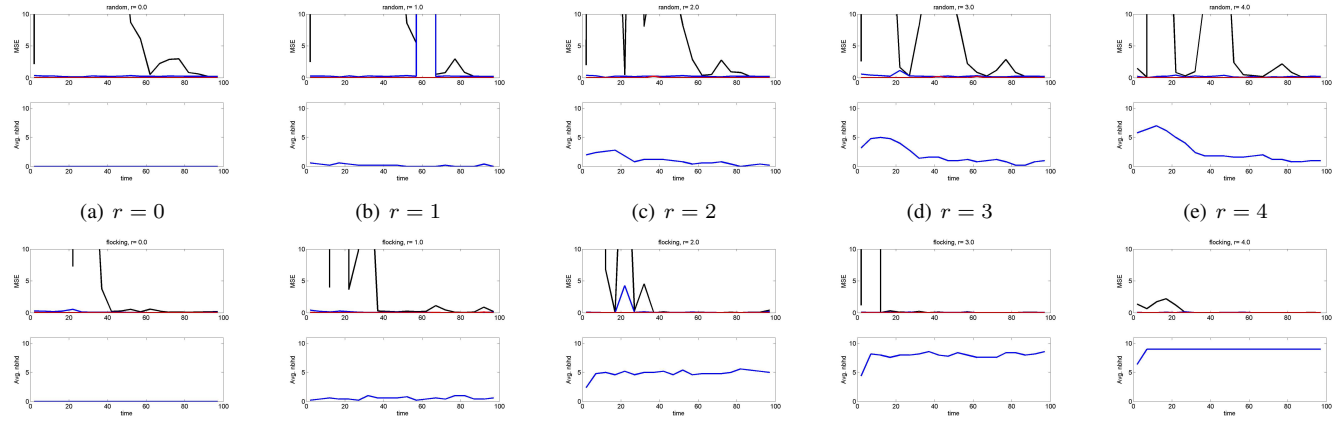


(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4 (e) Scenario 5

(f) Scenario 1 (g) Scenario 2 (h) Scenario 3 (i) Scenario 4 (j) Scenario 5

(k) Scenario 1 (l) Scenario 2 (m) Scenario 3 (n) Scenario 4 (o) Scenario 5

Fig. 3. Mean square errors (MSEs) of the estimated kernel parameters over simulation time for different scenarios. MSE represents the average mean square error over 10 independent runs for each scenario. (a-e) 10-agent system. Agents do not communicate and each agent learns independently. (f-j) 10-agent system. Agents communicate and learn using distributed CV learning. (k-o) 20-agent system. Agents communicate and learn using distributed CV learning.

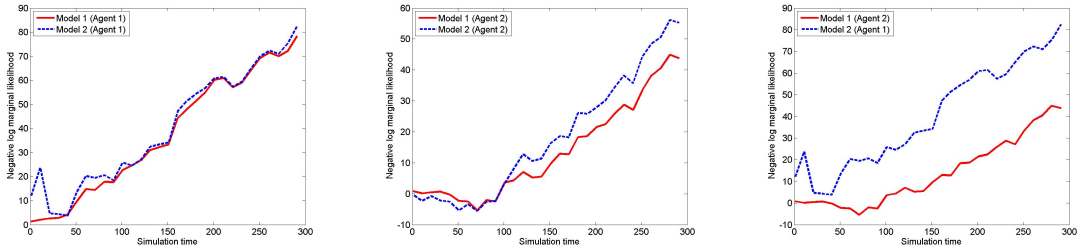


(a) $r = 0$ (b) $r = 1$ (c) $r = 2$ (d) $r = 3$ (e) $r = 4$

(f) $r = 0$ (g) $r = 1$ (h) $r = 2$ (i) $r = 3$ (j) $r = 4$

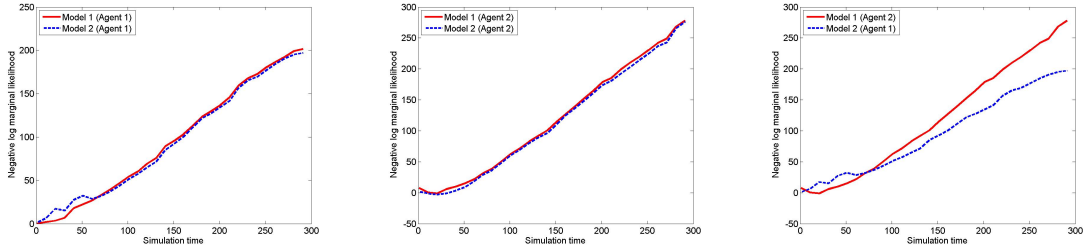
Fig. 4. 10 agent system. Mean square errors (MSEs) as a function of simulation time for different communication ranges when distributed CV learning is used. In each figure, the top plot shows MSE over time and the bottom plot shows the average number of neighboring agents. (a-e) random motion. (f-j) flocking-based motion.

models, each agent computed the negative log marginal likelihood (NLML). When the NLML is small, the model fits better based on the collected measurements. For Case 1, we expect the NLML for Model 1 to be small since it is the true model. However, it is a difficult task using one's own measurements to identify which model is the correct one as shown in Figure 5(a). But using cross validation, we can choose the correct model with higher confidence as shown in Figure 5(c). We can make the same conclusion for Case 2, where the situation is much more challenging. In this case, when each agent tests two models on its own data, there is virtually no difference between two models (Figure 6(a) and 6(b)). However, the test based on cross validation clearly indicates that Model 2 fits better (in other words, Model 2 has a smaller generalization error than Model 1). This example clearly demonstrates the power of cross validation for assessing generalizabilities of different models. Although it is not shown due to the space limitation, distributed CV learning also shows superior performance in terms of the mean square error (based on predicted values at 3,721 evenly spaced locations over the entire surveillance region).



(a) (b) (c)

Fig. 5. Evolution of the negative log marginal likelihood (NLML) over simulation time for Case 1 (the true model is Model 1). The model fits measurements better when the negative log likelihood is smaller. (a) NLMLs of two models computed by agent 1 using its measurements. (b) NLMLs of two models computed by agent 2 using its measurements. (c) Results from cross validation: agent 1 computes the NLML of Model 2 using agent 1's measurements and agent 2 computes the NLML of Model 1 using agent 2's measurements.



(a) (b) (c)

Fig. 6. Evolution of the negative log marginal likelihood over simulation time for Case 2 (the true model is Model 2). (a) NLMLs of two models computed by agent 1 using its measurements. (b) NLMLs of two models computed by agent 2 using its measurements. (c) Results from cross validation: agent 1 computes the NLML of Model 2 using agent 1's measurements and agent 2 computes the NLML of Model 1 using agent 2's measurements.

VI. Conclusions

In this paper, we have proposed a simple, yet powerful, distributed learning method for mobile sensor networks using cross validation. We have shown that cross validation can be easily embedded in a multi-agent system for rapid learning and model selection. In stead of sharing all measurements among agents, the algorithm reduces the communication load by only sharing model parameters. While learning and testing with the same set of measurements is prone to make a wrong selection, the distributed CV learning algorithm takes the advantage of a multi-agent system and is capable of making a more confident selection of a model. Since agents learn models in parallel in distributed CV learning, we have observed the speed up of the overall learning process. We have also shown that the performance of the distributed CV learning algorithm improves as more agents participate. We expect the proposed method can be applied to a wide range of estimation and model selection problems in multi-agent systems due to its simplicity and high performance.

References

[1] A. Singh, M. Batalin, M. Stealey, V. Chen, Y. Lam, M. Hansen, T. Harmon, G. S. Sukhatme, and W. Kaiser, "Mobile robot sensing for environmental applications," in *Proceedings of the International Conference on Field and Service Robotics*, 2007.

[2] N. Leonard, D. Paley, F. Lekien, R. Sepulchre, D. Fratantoni, and R. Davis, "Collectivemotion, sensor networks, and ocean sampling," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 48–74, 2007.

[3] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," *IEEE Pervasive computing*, vol. 3, no. 4, pp. 24–33, Oct. 2004.

[4] N. Cressie, "Kriging nonstationary data," *Journal of the American Statistical Association*, vol. 81, no. 395, pp. 625–634, September 1986.

[5] B. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," in *Proc. of the 20th International Joint Conference on Artificial Intelligence*, 2007.

[6] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, pp. 235–284, Feb. 2008.

[7] H. Durrant-Whyte, F. Ramos, E. Nettleton, A. Blair, and S. Vasudevan, "Gaussian process modeling of large scale terrain," in *Proc. of the 2009 IEEE International Conference on Robotics and Automation*, 2009.

[8] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, London, England, 2006.

[9] S. Geisser, *Predictive Inference*. New York: Chapman and Hall, 1993.

[10] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1995, pp. 1137–1143.

[11] A. Blum, A. Kalai, and J. Langford, "Beating the hold-out: Bounds for k-fold and progressive cross-validation," in *Proc. of the Conference on Computational Learning Theory*, Santa Cruz, CA, July 1999.

[12] Y. Yang, "Consistency of cross-validation for comparing regression procedures," *The Annals of Statistics*, vol. 35, no. 6, pp. 2450–2473, 2007.

[13] Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithm and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, March 2006.

[14] J. Choi, S. Oh, and R. Horowitz, "Distributed learning and cooperative control for multi-agent systems," *Automatica*, vol. 45, no. 12, pp. 2802–2814, Dec. 2009.

[15] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stability of flocking motion," University of Pennsylvania," Technical Report, 2003.

[16] J. Choi, J. Lee, and S. Oh, "Swarm intelligence for achieving the global maximum using spatio-temporal Gaussian processes," in *Proc. of the American Control Conference*, Seattle, WA, June 2008.