MACHINE LEARNING BASED LOCALIZATION

Chapter in book "Localization Algorithms and Strategies for Wireless Sensor Networks" (Eds: Guoqiang Mao and Baris Fidan, IGI Global)

Authors:

Duc A. Tran, Ph.D. (corresponding author)
Department of Computer Science
University of Massachusetts, Boston, MA 02125
Email: duc@cs.umb.edu
Tel: (617) 287-6452
Fax: (617) 287-6433

XuanLong Nguyen, Ph.D.
Statistical and Applied Mathematical Sciences Institute
and Department of Statistical Science
Duke University, Durham, NC 27708
Email: xuanlong.nguyen@stat.duke.edu
Tel: (919) 685-9339

Thinh Nguyen, Ph. D.
School of Electrical Engineering and Computer Science
Oregon State University, Corvallis, OR 97331
Email: thinhq@eecs.oregonstate.edu
Tel: (541) 737-3470

MACHINE LEARNING BASED LOCALIZATION

Duc A. Tran, University of Massachusetts, Boston, MA 02125, USA
XuanLong Nguyen, Duke University, Durham, NC 27708, USA
Thinh Nguyen, Oregon State University, Corvallis, OR 97331, USA

Abstract – A vast majority of localization techniques proposed for sensor networks are based on triangulation methods in Euclidean geometry. They utilize the geometrical properties of the sensor network to infer the sensor locations. A fundamentally different approach is presented in this chapter. This approach is based on machine learning, in which we work directly on the natural (non-Euclidean) coordinate systems provided by the sensor devices. The known locations of a few nodes in the network and the sensor readings can be exploited to construct signal-strength or hop-count based function spaces that are useful for learning unknown sensor locations, as well as other extrinsic quantities of interest. We discuss the applicability of two learning methods: the classification method and the regression method. We show that these methods are especially suitable for target tracking applications.

Keywords – Sensor networks, localization, kernel-based learning methods, regression, classification, support vector machines, kernel canonical correlation analysis.

INTRODUCTION

A sensor node knows its location either via a built-in GPS-like device or a localization technique. A straightforward localization approach is to gather the information (e.g., connectivity, pair-wise distance measure) about the entire network into one place, where the collected information is processed centrally to estimate the nodes' locations using mathematical algorithms such as Semidefinite Programming [Doherty et al. (2001)] and Multidimensional Scaling [Shang et al. (2003)].

Many techniques attempt localization in a distributed manner. The relaxation-based techniques [Savarese et al. (2001), Priyantha et al. (2003)] start with all the nodes in initially random positions and keep refining their positions using algorithms such as local neighborhood multilateration and convex optimization. The coordinate-system stitching techniques [Capkun et al. (2001), Meertens & Fitzpatrick (2004), Moore et al. (2004)] divide the network into overlapping regions, nodes in each region being positioned relatively to the region's local coordinate system (a centralized algorithm may be used here). The local coordinate systems are then merged, or "stitched", together to form a global coordinate system. Localization accuracy can be improved by using a set of nodes with known locations, called the beacon nodes, and extrapolate unknown node locations from the beacon locations [Bulusu et al. (2002), Savvides et al. (2001), Savvides et al. (2002), Niculescu & Nath (2003a), Nagpal et al. (2003), He et al. (2003)].

Most current techniques assume that the distance between two neighbor nodes can be measured, typically via a ranging procedure. In this procedure, various information can be used to help estimate pair-wise distance, such as Received Signal Strength Indication (RSSI) [Whitehouse (2002)], Time Difference of Arrival (TDoA) [Priyantha (2005), Kwon et al. (2004)], or Angle of

Arrival (AoA) [Priyantha et al. (2001), Niculescu & Nath (2003a)]. Other range measurement methods can be found in [Priyantha (2001b), Savvides et al. (2001b), Priyantha (2005b), Lee & Scholtz (2002), Gezici et al. (2005)].

To avoid the cost of ranging, range-free techniques have been proposed [Bulusu et al. (2002), Meertens & Fitzpatrick (2004), He et al. (2003), Stoleru et al. (2005), Priyantha et al. (2005)]. APIT [He et al. (2003)] assumes that a node can hear from a large number of beacons. Spotlight [Stoleru et al. (2005)] requires an aerial vehicle to generate light onto the sensor field. [Priyantha et al. (2005)] uses a mobile node to assist pair-wise distance measurements until a "global rigid" state can be reached where the sensor locations can be uniquely determined. DV-Hop [Niculescu & Nath (2003b)] and Diffusion [Bulusu et al. (2002), Meertens & Fitzpatrick (2004)] are localization techniques requiring neither ranging nor external assisting devices.

All the aforementioned techniques use Euclidean geometrical properties to infer the sensor nodes' locations. Recently, a number of techniques that employ the concepts from machine learning have been proposed [Brunato & Battiti (2005), Nguyen et al. (2005), Pan et al. (2006), Tran & Nguyen (2006), Tran & Nguyen (2008), Tran & Nguyen (2008b)]. The main insight of these methods is that the topology implicit in sets of sensor readings and locations can be exploited in the construction of possibly non-Euclidean function spaces that are useful for the estimation of unknown sensor locations, as well as other extrinsic quantities of interest. Specifically, one can assume a set of beacon nodes and use them as the training data for a learning procedure. The result of this procedure is a prediction model that will be used to localize the sensor nodes of previously unknown positions.

Consider a sensor node S whose true (unknown) location is (x, y) on a 2-D field. There are more than one way we can learn the location of this node. For example, we can model the localization problem as a classification problem [Nguyen et al. (2005), Tran & Nguyen (2006), Tran & Nguyen (2008)]. Indeed, we can define a set of classes (e.g., A, B, and C as in Figure 1) that represent geographic regions chosen appropriately in the sensor network area. We then run a classification procedure to decide the membership of S in these classes. Based on these memberships, we can localize S. For example, in Figure 1, if the output of the classification procedure is that S is a member of class A, of B, and of C, then S must be in the intersectional area $A \cap B \cap C$.

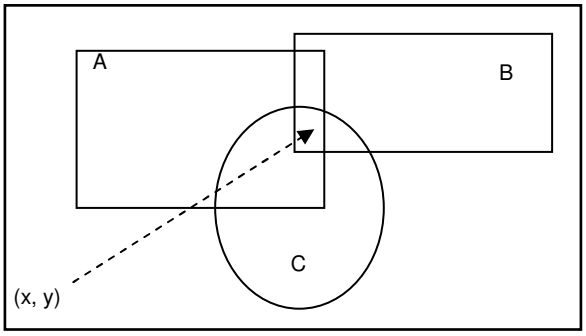

Figure 1 If we can define a set of classes that represent geographic regions, a sensor node's location can be estimated based on its memberships in these classes

We can also solve the localization problem as a regression problem [Pan et al. (2006), Tran & Nguyen (2008b)]. We can use a regression tool to infer the Euclidean distances between *S* and the beacon nodes based on the signal strengths that *S* receives from these nodes, or when *S* cannot hear directly from them, based on the hop-count distances between *S* and these nodes. After these distances are learned, trilateration can be used to estimate the location of *S*. Alternatively, we can apply a regression tool that maps the signal strengths *S* receives from the beacon nodes *directly* to a location. One such a tool was proposed by [Pan et al. (2006)], which is based on Kernel Canonical Correlation Analysis [Hardoon et al. (2004)].

Compared to geometric-based localization techniques, the requirements for the learning-based techniques to work are modest. Neither ranging measurements nor external assisting devices are needed. The only assumption is the existence of a set of beacon nodes at known locations. The information serving as input to the learning can be signal strengths [Nguyen et al. (2005), Pan et al. (2006)] or hop-count information [Tran & Nguyen (2006), Tran & Nguyen (2008)], which can be obtained easily at little cost.

The correlation between the signal-strength (and/or hop-count) space and the physical location space is generally non-linear. It is also usually not possible to know *a priori*, given a sensor node, the exact features that uniquely identify its location. A versatile and productive approach for learning correlations of this kind is based on the kernel methods for statistical classification and regression [Scholkopf & Smola (2002)]. Central to this methodology is the notion of a *kernel function*, which provides a generalized measure of similarity for any pair of entities (e.g., sensor locations, sensor signals, hop-counts). The functions that are produced by the kernel methods (such as support vector machines and kernel canonical correlation analysis) are sums of kernel functions, with the number of terms in the sum equal to the number of data points. Kernel methods are examples of nonparametric statistical procedures – procedures that aim to capture large, open-ended classes of functions.

Given that the raw signal readings in a sensor network implicitly capture topological relations among sensor nodes, kernel methods would seem to be particularly natural in the sensor network setting. In the simplest case, the signal strength/hop-count would itself be a kernel function. More generally, and more realistically, derived kernels can be defined based on the signal strength/hop-count matrix. In particular, inner products between vectors of received signal strengths/hop-counts can be used in kernel methods. Alternatively, generalized inner products of these vectors can be computed – this simply involves the use of higher-level kernels whose arguments are transformations induced by lower-level kernels. In general, hierarchies of kernels can be defined to convert the initial topology provided by the raw sensor readings into a topology more appropriate for the classification or regression task at hand. This can be done with little or no knowledge of the physical sensor model.

In this chapter, we describe localization techniques that build on kernel-based learning methods for classification and regression/correlation analysis.

NOTATIONS AND ASSUMPTIONS

We consider a wireless sensor network of N nodes $\{S_1, S_2, \ldots, S_N\}$ deployed in a 2-D geographic area $[0, D]^2$ ($D > 0$). (Here, we assume two dimensions for simplicity, though the techniques to

be presented can work with any dimensionality.) We assume the existence of k beacon nodes $\{S_1, S_2, \ldots, S_k\}$ with known location ($k < N$). We will devise learning-based algorithms where an estimate can be made for the location of each remaining node $\{S_{k+1}, S_{k+2}, \ldots, S_N\}$.

We assume that the network is connected and an underlying routing protocol exists to provide a path $path(S_i, S_j)$ to navigate from any sensor node S_i to any other S_j, whose hop-count distance (or distance, *in short*) is denoted by $hc(S_i, S_j)$. If the routing protocol defines this path to be the shortest path in hop-count from S_i to S_j, the distance $hc(S_i, S_j)$ is the least number of hops between them. The sensor coverage is not necessarily uniform; hence, $path(S_i, S_j)$ may not equal $path(S_j, S_i)$ and $hc(S_i, S_j)$ may not equal $hc(S_j, S_i)$. Also, we denote by $ss(S_i, S)$ the signal strength a sensor node S receives from each beacon S_i.

If the network is small enough that any sensor node can hear directly from a majority of the beacons, we can use signal-strength information to estimate the locations for sensor nodes. In practice, however, there is a large class of sensor networks where a node may hear directly from just a few beacons and there may be nodes that do not hear directly from any beacon node. For this type of networks, we learn to estimate the locations based on hop-count information rather than signal-strength information.

Before we present the details in the next sections, the localization procedure is summarized as follows:

1. The beacon nodes communicate with each other so that for each beacon node S_i we can obtain the following k-dimensional distance vector

   $$h_i = \left(hc(S_1, S_i) \quad hc(S_2, S_i) \quad \ldots \quad hc(S_k, S_i) \right)$$

   or, for the case of a small network, the k-dimensional signal-strength vector

   $$s_i = \left(ss(S_1, S_i) \quad ss(S_2, S_i) \quad \ldots \quad ss(S_k, S_i) \right)$$

2. One beacon node is chosen, called the head beacon, to collect all these vectors from the beacon nodes and run a learning procedure (regression or classification). After the learning procedure, the prediction model is broadcasted to all the nodes in the network. Furthermore, each beacon node broadcasts a HELLO message to the network also.

3. As a result of receiving the HELLO message from each beacon, each sensor node $S_j \in \{S_{k+1}, S_{k+2}, \ldots, S_N\}$ computes the following k-dimensional distance vector

   $$h_j = \left(hc(S_1, S_j) \quad hc(S_2, S_j) \quad \ldots \quad hc(S_k, S_j) \right)$$

   or, for the case of a small network, the k-dimensional signal-strength vector

   $$s_j = \left(ss(S_1, S_j) \quad ss(S_2, S_j) \quad \ldots \quad ss(S_k, S_j) \right)$$

The sensor node then applies the prediction model it has obtained previously to this distance (or signal-strength) vector to estimate the node's location.

LOCALIZATION BASED ON CLASSIFICATION

As we mentioned in the Introduction section, the localization problem can be modeled as a classification problem. The idea was initiated in [Nguyen et al. (2005)]. Generally, the first two steps are as follows:

- Class definition: Define a set of classes $\{C_1, C_2, \ldots\}$, with each class C_i being a geographical region in the sensor network area
- Training data: Because the beacon locations are known, the membership of each beacon node in each class C_i is known. The distance (or signal-strength) vector of each beacon node serves as its feature vector. The feature vector and membership information serves as the training data for the classification procedure on class C_i

We then run the classification procedure to obtain a prediction model. This model is used to estimate for each given sensor node S and class C_i the membership of S in class C_i. As a result, we can determine the area in which S is located. To solve the classification problem, it is proposed in [Nguyen et al. (2005), Tran & Nguyen (2006), Tran & Nguyen (2008)] that we use Support Vector Machines (SVM), a popular and efficient machine learning method [Cortes & Vapnik (1995)]. Specifically, these techniques use binary SVM classification methods – the traditional form of SVM. A brief background on binary SVM classification is presented below and then how it is used for sensor localization.

Binary SVM Classification

Consider the problem of classifying data in a data space U into a class G or not in class G. Suppose that k data points $u_1, u_2, \ldots, u_k$, are given, called the *training* points, for which the corresponding memberships in class G are known. We need to predict whether a new data point u is in G or not. This problem is called a binary classification problem.

Support Vector Machines (SVM) [Boser et al. (1992), Cortes & Vapnik (1995)] is an efficient method to solve this problem. Central to this method is the notion of a kernel function K: $U \times U \rightarrow R$ that provides a measure of similarity between two data points in U. For the case of finite data space (e.g., location data of nodes in a sensor network), this function must be symmetric and the $k \times k$ matrix $[K(u_i, u_j)]$ $(i, j \in \{1, 2, \ldots, k\})$ must be positive semi-definite (i.e., has non-negative eigenvalues).

Given such a kernel function K, according to Mercer's theorem [cf., Scholkopf & Smola (2002)], there must exist a feature space in which the kernel K acts as the inner product, i.e., $K(u,u') = \langle \Phi(u), \Phi(u') \rangle$ for some mapping $\Phi(u)$. Suppose that we associate with each training data point u_i a label l_i to represent that $l_i = 1$ if $u_i \in G$ and -1 otherwise. The idea is to find a hyperplane in the feature space, that maximally separates the training points in class G from those not in G. For this purpose, the SVM and related kernel-based algorithms find a linear function $h_K(u) = \langle w, \Phi(u) \rangle - b$ in the feature space, where the vector w and parameter b are

chosen to maximize the margin, or distance between the parallel hyperplanes that are as far apart as possible while still separating the training data points. Thus, if the training data points are linearly separable in the feature space, we need to minimize $\|w\|$ subject to $1 - l_i h_K(u_i) \leq 0$ for all $1 \leq i \leq k$.

Solving the above minimization problem requires the knowledge about the feature mapping $\Phi(u)$. Fortunately, by the Representer Theorem [cf., Scholkopf & Smola (2002)], the function h_K can be expressed in terms of the kernel function K only

$$h_K(u) = \sum_{i=1}^{k} \alpha_i l_i K(u, u_i) + b$$

for an optimizing choice of coefficients α_i. Using this dual form, to find the function $h_K(u)$, we solve the following maximization problem:

$$\text{Maximize} \qquad W(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{k} l_i l_j \alpha_i \alpha_j K(u_i, u_j)$$

$$\text{subject to} \qquad \sum_{i=1}^{k} l_i \alpha_i = 0 \text{ and } 0 \leq \alpha_i \text{ for } i \in \{1, 2, \ldots, k\}$$

Suppose that $\{\alpha_1^*, \alpha_2^*, \ldots, \alpha_k^*\}$ is the solution to this optimization problem. We choose $b = b^*$ such that $l_i h_K(u_i) = 1$ for all i with $0 < \alpha_i^*$. The training points corresponding to such (i, α_i^*)'s are called the *support vectors*. The decision rule to classify a data point u is: $u \in G$ iff $sign(h_K(u)) = 1$, where $h_K(u) = \sum_{i=1}^{k} \alpha_i^* l_i K(u, u_i) + b^*$.

Under standard assumptions in statistical learning, SVM is known to yield bounded (and small) classification error when applied to the test data. The SVM method presented above is the 1-norm soft margin version of SVM. There are several extensions to this method, whose details can be found in [Boser et al. (1992), Cortes & Vapnik (1995)].

The main property of the SVM is that it only needs the definition for a kernel function K that represents a similarity measure between two data points. This is a nice property because other classifier tools usually require a known feature vector for every data point, which may not be available or derivable in many applications. In our particular case of a sensor network, it is impossible to find the features for each sensor node that uniquely and accurately identify its location. However, we can provide a similarity measure between two sensor nodes based on their relationships with the beacon nodes. Thus, SVM is highly suitable for the sensor localization problem.

Class Definition

There are more than one way to define the classes $\{C_1, C_2, \ldots\}$. For example, as illustrated in [Nguyen et al. (2005)], each class C_i can be an equi-size disk in the sensor network area such that any point in the sensor field must be covered by at least three such disks. Thus, after the learning

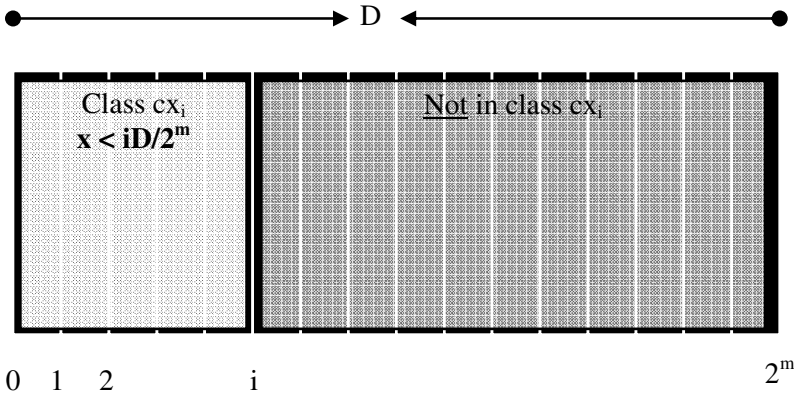

Figure 2 Definition of class cx_i (i = 1, 2, …, 2^m-1)

procedure, if a sensor node S is found to be a member of three classes C_i, C_j, and C_k, the location of S is approximated as the centroid of the intersectional area $C_i \cap C_j \cap C_k$.

Using the above disk partitioning method, or any method requiring that any point in the sensor network field be covered by two or more regions represented by classes, the number of classes in the learning procedure is dependant on the field dimension and could be very high. Alternatively, [Tran & Nguyen (2006), Tran & Nguyen (2008)] propose the LSVM technique which partitions the sensor network field using a fixed number of classes, thus independent of the network field dimension (LSVM is the abbreviation for **L**ocalization based on **SVM**). Hereafter, unless otherwise mentioned, the technique we describe is LSVM. As illustrated in Figure 2, LSVM defines (2M-2) classes as follows, where $M = 2^m$ for some m determined later:

- M-1 classes for the X-dimension $\{cx_1, cx_2, ..., cx_{M-1}\}$, each class cx_i containing nodes with the x-coordinate $x < iD/M$
- M-1 classes for the Y-dimension $\{cy_1, cy_2, ..., cy_{M-1}\}$, each class cy_i containing nodes with the y-coordinate $y < iD/M$

We need to solve (2M-2) binary classification problems. Each solution, corresponding to a class cx_i (or cy_i), results in a SVM prediction model that decides whether a sensor node belongs to this class or not. If the SVM learning predicts that a node S is in class cx_{i+1} but not class cx_i, and in class cy_{j+1} but not class cy_j, we conclude that S is inside the square cell $[iD/M, (i+1)D/M] \times [jD/M, (j+1)D/M]$. We then simply use the cell's center point as the estimated position of node S (see Figure 3). If the above prediction is indeed correct, the location error (i.e., Euclidean distance between true location and estimated location) for node S is at most $\frac{D}{M\sqrt{2}}$. However, every SVM is subject to some classification error, and so a challenge is to maximize the probability that S is classified into its *true* cell, and, to minimize the location error in the case that S is classified into a wrong cell [Tran & Nguyen (2008)].

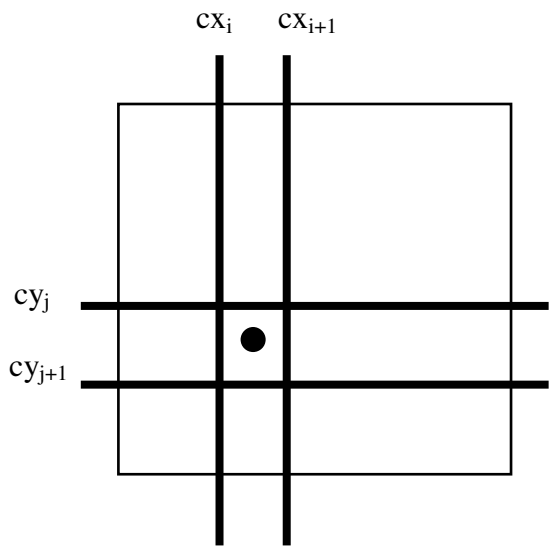

Figure 3 Localization of a node based on its memberships in regions cx_i and cy_j

Kernel Function

The kernel function $K(S_i, S_j)$ provides a measure for similarity between two sensor nodes S_i and S_j. We define the kernel function as a Radial Basis Function because of its empirical effectiveness [Chang & Lin (2008)]:

$$K(S_i, S_j) = \exp(-\gamma \left\| h_i - h_j \right\|^2)$$

where γ is a constant to be computed during the cross-validation phase of the training procedure, and h_i the k-dimensional distance vector of sensor node S_i with the j-th entry of the vector representing the hop-count distance from S_i to beacon node S_j. More examples for the kernel function are discussed in [Nguyen et al. (2005)].

Training Data

For each binary classification problem (for a class $c \in \{cx_1, cx_2, ..., cx_{M-1}, cy_1, cy_2, ..., cy_{M-1}\}$), the training data is the set of beacon nodes with corresponding labels $\{l_1, l_2, ..., l_k\}$, where $l_i = 1$ if beacon node S_i belongs to class c and -1 otherwise.

Now that the training data and kernel function have been defined for each class c, we can solve the SVM optimization problem aforementioned to obtain $\{\alpha_1^*, \alpha_2^*, ..., \alpha_k^*\}$ and b^*. We then use the decision function $h_K(.)$ to decide whether a given node S is in class c:

$$h_K(S) = \sum_{i=1}^{k} \alpha_i^* l_i K(S, S_i) + b^*$$

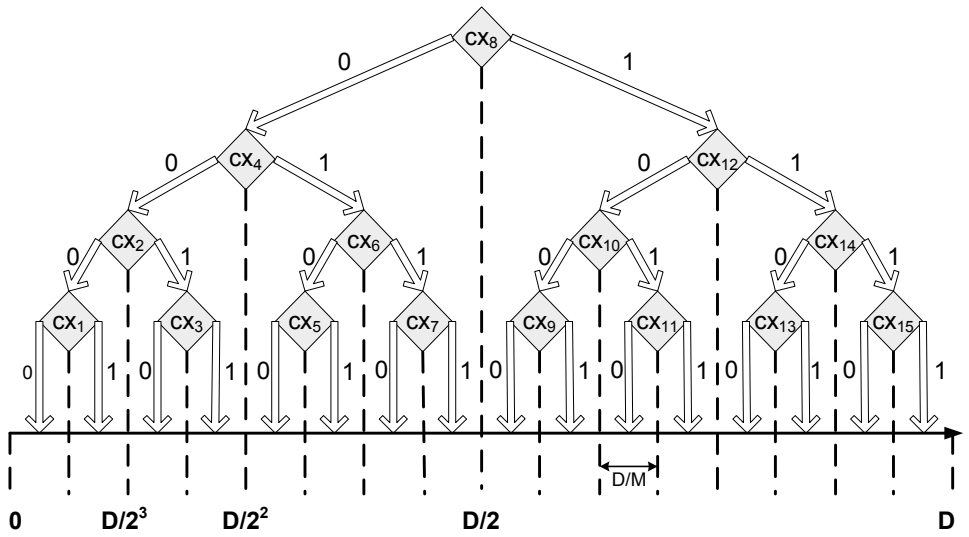

Figure 4 Decision tree: $m = 4$

The training procedure is implemented as follows. The head beacon obtains the hop-count vector and location of each beacon. Then, it runs the SVM training procedure (e.g., using a SVM software tool like *libsvm* [Chang & Lin (2008)] on all (2M-2) classes $cx_1, cx_2, ..., cx_{M-1}, cy_1, cy_2, ..., cy_{M-1}$ and, for each class, computes the corresponding b^* and the information $(i, l_i\alpha_i^*)$. This information is called the SVM model information. This model information is used to predict the location of any sensor given its distance vector.

Location Estimation

Let us focus on the classification along the X-dimension. LSVM organizes the cx-classes into a binary decision tree, illustrated in Figure 4. Each tree node is a cx-class and the two outgoing links represent the outcomes (0: "not belong", 1: "belong") of classification on this class. The classes are assigned to the tree nodes such that if the tree is traversed in the *in-order* order {*left-subtree*→ *parent*→ *right-subtree*}, the result is the ordered list $cx_1 \rightarrow cx_2 \rightarrow ... \rightarrow cx_{M-1}$. Given this decision tree, each sensor node S can estimate its x-coordinate using the following algorithm:

Algorithm: X-dimension localization
Estimate the x-coordinate of sensor node S:

1. *Initially, $i = M/2$ (start at root of the tree $cx_{M/2}$)*
2. *IF (SVM predicts S not in class cx_i)*
   - *IF (cx_i is a leaf node – i.e., having no child decision node)*
     RETURN $x'(S) = (i - 1/2)D/M$
   - *ELSE Move to left-child cx_j and set $i = j$*
3. *ELSE*
   - *IF (cx_i is a leaf node) RETURN $x'(S) = (i + 1/2)D/M$*
   - *ELSE Move to right-child cx_t and set $i = t$*
4. *GOTO Step 2*
5. *END*

Similarly, a decision tree is built for the Y-dimension classes and each sensor node S estimates its y-coordinate $y'(S)$ based on the *Y-dimension localization algorithm* (like the *X-dimension localization algorithm*). The estimated location for node S, consequently, is $(x'(S), y'(S))$. Using these algorithms, the localization of a node requires visiting $log_2 M$ nodes of each decision tree, after each visit the geographic range that contains node S downsizing by a half. The parameter M (or m) controls the precision of the localization.

SVM is subject to error and so is LSVM. Let ε be the worst-case SVM classification error when SVM is applied to solve the *(2M-2)* binary classification problems, each regarding one of the classes $\{cx_1, cx_2, \ldots, cx_{M-1}, cy_1, cy_2, \ldots, cy_{M-1}\}$. For each class c, a misclassification occurs when SVM predicts that a sensor node is in c but in fact the node is not, or when SVM predicts that the node is not in c but the node actually is. The SVM classification error corresponding to class c is the ratio between the number of sensor nodes for which SVM predicts correctly to the total number of all sensor nodes. In [Tran & Nguyen (2008)], it is shown that for a uniformly distributed sensor network field, the location error expected for any node is bounded by

$$E^u = \sqrt{2}D\left(\frac{1}{2^m} + \frac{7}{8} - \frac{(1-\varepsilon)^m}{2^{m+1}} - \frac{(2-\varepsilon)^m}{2^m} + \frac{(4-3\varepsilon)^m}{2^{2m+3}}\right)$$

The location error expectation E^u decreases as the SVM error ε gets smaller. Figure 5 plots the error expectation E^u for various values of ε. There exists a choice for m (no larger than 8) that minimizes the error expectation. In a real-world implementation, it is recommended that we use this optimal m. A nice property of SVM is that ε is typically upper-bounded and under certain assumptions on the choice of the kernel function, the bound

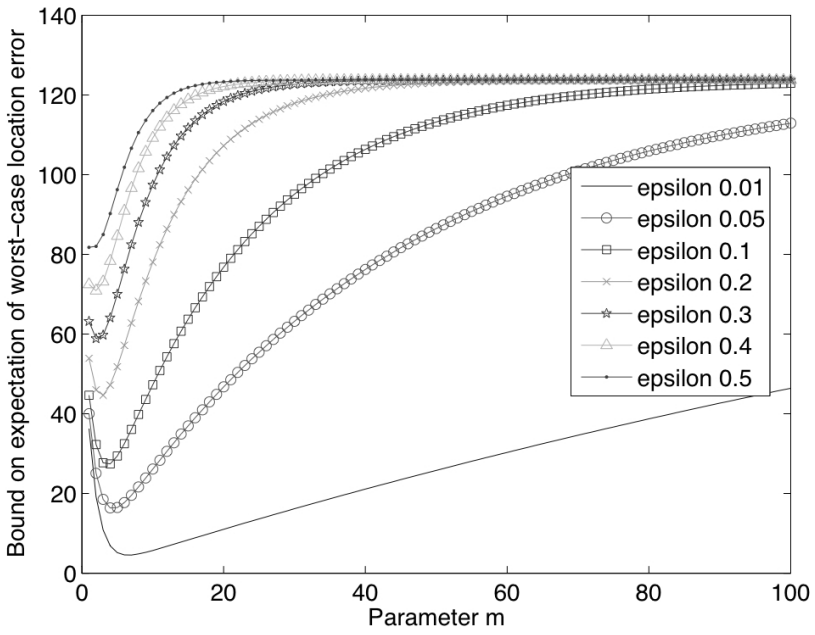

Figure 5 Upper bound on the expectation of worst-case location error under various values of SVM classification error (epsilon ε). A lower SVM error corresponds to a lower-appearing curve.

diminishes if the training size gets sufficiently large. In the evaluation study of [Tran & Nguyen (2008)], when simulated on a network of 1000 sensors with non-uniform coverage, of which 5% serves as beacon nodes, the error ε is no more than 0.1. This is one example showing that SVM offers a high accuracy when used to classify the sensor nodes into their correct classes. Later in this chapter more evaluation results are presented to demonstrate the localization accuracy of LSVM.

LOCALIZATION BASED ON REGRESSION

Trilateration is a geometrical technique that can locate an object based on its Euclidean distances from three or more other objects. In our case, to locate a sensor node we do not know its true Euclidean distances from the k beacon nodes. We can use a regression tool (e.g., *libsvm* [Chang & Lin (2008)]) to learn about these distances using hop-count information. The beacon leader constructs a linear regression function f: $N \rightarrow R$ with the following training data

$f(hc(S_i, S_j)) = d(S_i, S_j)$ for all $i, j \in \{1, 2, \ldots, k\}$

where $d(S_i, S_j)$ is the Euclidean distance between S_i and S_j. Once this regressor f is computed, it is broadcast to all the sensor nodes. Since each node receives a HELLO message from each beacon, the former can compute its distance vector and apply the regressor f to compute its location [Tran & Nguyen (2008b)]. A similar approach, but applied on signal-strength data, was considered by [Kuh & Zhu (2006), Zhu & Kuh (2007), Kuh & Zhu (2008)]. Kuh & Zhu uses least squares SVM regression to solve the localization problem with beacon locations as training data. This involves solving a system of linear equations. To achieve sparseness, a procedure is used to choose the support vectors based on training data error.

If the network is sufficiently small, each sensor node can hear from all the beacon nodes. It is observed that if two nodes S_i and S_j receive similar signal strengths from the beacon nodes, and, if the number of beacons is large enough (at least 3), these nodes should be near each other in the physical space. Thus, one could be able to exploit directly the high correlation statistics between the similarity of signal strengths and that of sensor locations. This insight was observed by [Pan et al. (2006)], who proposed to use Kernel Canonical Correlation Analysis (KCCA) [Akaho (2001), Hardoon et al. (2004)] for the regression that maps a vector in the signal-strength space to a location in the physical space. We briefly present KCCA below and then how it is used for the localization problem.

Kernel Canonical Correlation Analysis (KCCA)

KCCA is an efficient non-linear extension of Canonical Correlation Analysis (CCA) [Hotelling (1936), Hardoon et al. (2004)]. Suppose that there are two sets of multidimensional variables, $s = (s_1, s_2, \ldots, s_k)$ and $t = (t_1, t_2, \ldots, t_k)$. CCA finds two canonical vectors, w_s and w_t, one for each set such that the correlation between these two sets under the projections, $a = \left(\langle w_s, s_1 \rangle, \langle w_s, s_2 \rangle, \ldots, \langle w_s, s_k \rangle\right)$ and $b = \left(\langle w_t, t_1 \rangle, \langle w_t, t_2 \rangle, \ldots, \langle w_t, t_k \rangle\right)$ is maximized (the correlation is defined as $cor(a,b) = \dfrac{\langle a,b \rangle}{\|a\|\|b\|}$).

While CCA only exploits linear relationship between s and t, its extension using kernels KCCA can work with non-linear relationships. KCCA defines two kernels, K_s for the s space and K_t for the t space. Each kernel K_s (or K_t) represents implicitly a feature vector space Φ_s (or Φ_t) for the corresponding variable s (or t). Then, a mapping that maximizes the correlation between s and t in the feature space is found using the kernel functions only (requiring no knowledge about Φ_s and Φ_t).

KCCA for Localization

[Pan et al. (2006)] applies KCCA to find a correlation-maximizing mapping from the signal-strength space to the physical location space (because the relationship is non-linear, KCCA is more suitable than CCA). Firstly, two kernel functions are defined, a Gaussian kernel K_s for the signal space

$$K_s(s_i, s_j) = \exp(-\gamma \left\| s_i - s_j \right\|^2)$$

and a Matern kernel K_t for the location space

$$K_t(t_i, t_j) = \frac{2(\sqrt{v} w \left\| t_i - t_j \right\|)^v}{\Gamma(v)} K_v(2\sqrt{v} w \left\| t_i - t_j \right\|)$$

where v is a smoothness parameter, $\Gamma(v)$ the gamma function, and $K_v(.)$ the modified Bessel function of the second kind. The signal strengths between the beacon nodes and their location form the training data. In other words, the k instances (s_1, t_1), (s_2, t_2), …, (s_k, t_k), where (s_i, t_i) represents the signal-strength vector and the location of beacon node S_i, serve as the training data.

After the training is completed, suppose that q pairs of canonical vectors (ws_1, wt_1), (ws_2, wt_2), …, (ws_q, wt_q) are found. The choice for q is flexible. Technically, more than one pair of canonical vectors can be found recursively in such a way that a newly found pair must be orthogonal to the previous pair and maximally correlate the canonical variates resulted from the previous pair. Thus, q can be chosen as large as we can find a new pair of canonical vectors that improves the correlation according to the previous pair by a *significant* margin (which can be defined by some threshold).

A sensor node $S \in \{S_{k+1}, S_{k+2}, \ldots, S_N\}$ is localized as follows:

- Compute the signal-strength vector s of sensor node S: $s = (ss(S_1, S), ss(S_2, S), \ldots, ss(S_k, S)$
- Compute the projection of $P(s) = \left(\langle ws_1, s\rangle, \langle ws_2, s\rangle, \ldots, \langle ws_q, s\rangle\right)$
- Choose from the set of beacon nodes m nodes $\{S_i\}$ whose projections $\{P(s_i)\}$ are nearest to $P(s)$. The distance metric used is a weighted Euclidean distance where the weights are obtained from the KCCA training procedure and the canonical vectors $wt_1, wt_2, \ldots, wt_q$
- Compute the location for S as the centroid position of these m neighbors

EVALUATION RESULTS

This section presents some evaluation results that demonstrate the effectiveness of the learning-based approach to the sensor localization problem. The main overhead for this approach is the training procedure. It involves communication among the beacon nodes to obtain their distance (or signal-strength) vectors. Then, the head beacon collects this information to run the SVM, resulting in a prediction model which is then broadcast to all the nodes in the network. The location estimation procedure at each node consists of only a small number of comparisons and simple computations. Thus, the approach is fast and simple.

In the following, we show the location error results for LSVM – the classification based technique that uses the hop-count information to learn the sensor nodes' locations. These results are extracted from the evaluation study presented in [Tran & Nguyen (2008)]. The evaluation results for the other learning-based techniques can be found in [Tran & Nguyen (2008b)] (regression-based localization using hop-count information), [Nguyen et al. (2005)] (classification-based localization using signal strength) and [Pan et al. (2006)] (regression-based localization using signal strength).

In [Tran & Nguyen (2008)], LSVM is compared to Diffusion [Bulusu et al. (2002), Meertens & Fitzpatrick (2004)]. Diffusion is an existing technique that does not require ranging measurements and also uses beacon nodes with known locations. Unlike LSVM, Diffusion is not based on machine learning. In Diffusion, each sensor node's location is initially estimated as a random location in the sensor network area. Each node, a sensor node or a beacon node, then repeatedly exchanges its location estimate with its neighbors and uses the centroid of the neighbors' locations as the new location estimate. This procedure after a number of iterations will converge to a state where each node does not improve its location estimate significantly.

Consider a network of 1000 sensor nodes located in a 100m by 100m 2-D area. The selection of the beacon nodes among the sensor nodes is based on uniform distribution. The communication radius for each node is 10m. Five different beacon populations are considered: 5% of the network size (k = 50 beacons), 10% (k = 100 beacons), 15% (k = 150 beacons), 20% (k = 200 beacons), and 25% (k = 250 beacons). The algorithms in the *libsvm* software kit [Chang & Lin (2008)] are used for SVM classification. The parameter m is set to 7 (i.e., M = 128).

Figure 6 shows that LSVM is more accurate than Diffusion. In this study, node locations are uniformly distributed in the network area. Diffusion converges after 100 iterations (Diff-100). It does not improve when more iterations are run, 1000 iterations (Diff-1000) or 10,000 iterations (Diff-10000). The difference between the two techniques is the most noticeable when the number of beacons is small (k = 50) and decreases as more beacon nodes are used. In any case, even when k = 50 (only 5% of the network serve as beacon nodes), the location error for an average node using LSVM is always less than 6m.

Another nice property of LSVM is that it distributes the error fairly across all the nodes. As an example, Figure 7 shows the localization results for the case k = 50. In this figure, a line connects the true location and the estimated location for each node. It can be observed that Diffusion suffers severely from the border problem: nodes near the border of the network area are poorly localized. LSVM does not incur this problem.

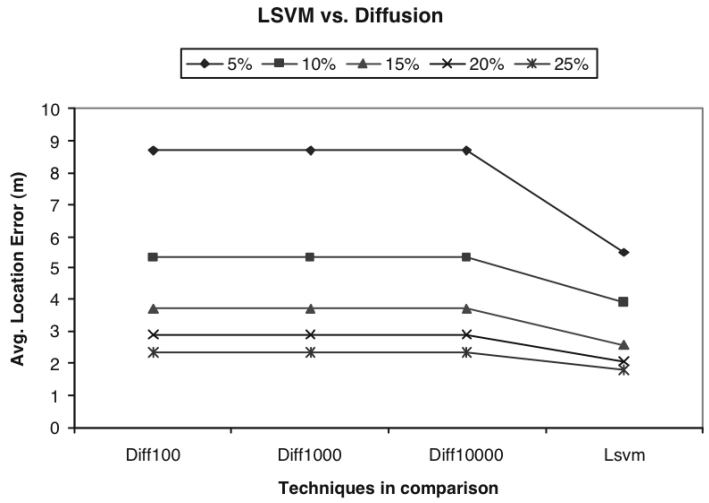

Figure 6 LSVM vs. Diffusion: Average location error with various choices for the number of beacons.

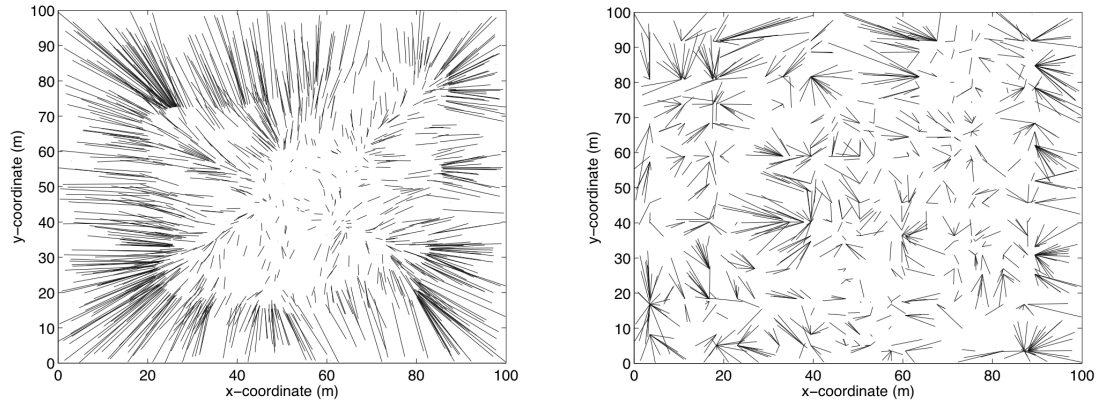

Figure 7 Diffusion (1000 iterations, left) vs. LSVM (right): A line connects the true location and the estimated location of each sensor node (total 1000 nodes, 50 beacon nodes)

Many networking protocols such as routing and localization suffer from the existence of coverage holes or obstacles in the sensor network area. [Tran & Nguyen (2008)] also shows that even so, LSVM remains much better than Diffusion. For example, with the sensor network placement shown in Figure 8, where there is a big hole of radius 25m centered at position (50, 50). Table 1 shows that LSVM improves the location error over Diffusion by at least 20% in all measures (average, worst, standard deviation) under every beacon population size.

APPLICATION TO TARGET TRACKING

An appealing feature of the presented learning-based approach is that the localization of a sensor node can be done independently from that of another sensor node. The training procedure involves the beacon nodes only, whose result is a prediction model any sensor node can use to localize itself without knowledge about other nodes. This feature is suitable for target tracking in a sensor network where, to save cost, not every sensor node needs to run the localization algorithm; only the target needs to be localized. For example, consider a target tracking system

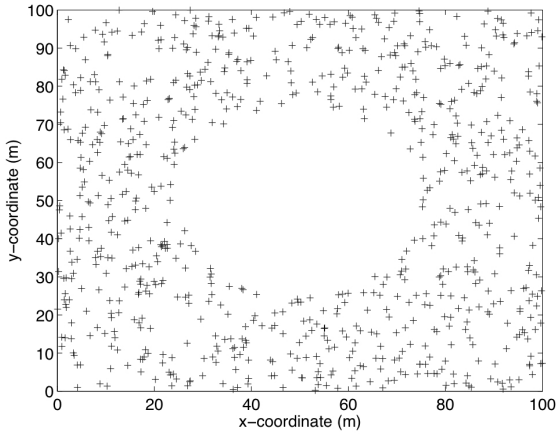

Figure 8 A big coverage hole at the middle of the network area

	k = 50	100	150	200	250
Average-case	30.97%	31.30%	34.88%	33.91%	26.50%
Worst-case	27.35%	21.40%	34.35%	33.42%	23.83%
Std. deviation	35.74%	36.29%	37.66%	37.40%	28.34%

Table 1 Location-error improvement of LSVM over Diffusion for the network with a coverage hole shown in Figure 8.

with k beacon nodes deployed at known locations. When a target T occurs in an area and is detected by a sensor node S_T, the detecting node reports the event to the k beacon nodes. The distance vector $[hc(S_T, S_i)]$ $(i = 1, 2, \ldots, k)$ is forwarded to the sink station who will use the prediction model learned in the training procedure to estimate the location of target T.

An important issue in the learning-based approach is that its accuracy depends on the size of the training data; in our case, the number of beacon nodes. However, in many situations, the beacon nodes are deployed incrementally, starting with a few beacon nodes and gradually with more. In other cases, the set of beacon nodes can also be dynamic. The beacon nodes that are made available to a sensor node (or target) under localization may change depending on the location of this node (or target). We need a solution that learns based on not only the current measurements but also the past. For example, reconsider the target tracking system mentioned above. When a target is detected, sending the event to all the beacon nodes can be very costly. Instead, the detecting node reports the event to a few, possibly random, beacon nodes. Learning based on the current measurements (signal strengths or hop-counts) may be inaccurate because of the sparse training data, but as the target moves, by combining the past learned information with the current, we can better localize the target. Sequential prediction techniques [Cesa-Bianchi & Lugosi (2006)] can be helpful for this purpose.

[Letchner et al. (2005)] propose a localization technique aimed at such dynamism of the beacon nodes. The technique is based on a hierarchical Bayesian model which learns from signal strengths to estimate the target's location. It is able to incorporate new beacon nodes as they appear over time. Alternatively, [Oh et al. (2005)] consider a challenging problem of multiple-target tracking by Markov chain Monte Carlo inference in a hierarchical Bayesian model.

Recently, [Pan et al. (2007)] address the problem of not only locating the mobile target but also dynamically located beacon locations. The solution proposed in [Pan et al. (2007)] is based on online and incremental manifold-learning techniques [Law & Jain (2006)] which can utilize both labeled and unlabeled data that come sequentially.

Both [Letchner et al. (2005)] and [Pan et al. (2007)] learn from signal strength information, thus suitable for small networks where measurements of direct signals from the beacons are possible. The ideas could be applicable to a large network where hop-count information is used in the learning procedure rather than signal strengths. The effectiveness, however, has not been evaluated. Investigation in this direction would be an interesting problem for future research.

SUMMARY

This chapter provides a nonconventional perspective to the sensor localization problem. In this perspective, sensor localization can be seen as a classification problem or a regression problem, two popular subjects of Machine Learning. In particular, the presented localization techniques borrow the ideas from kernel methods.

The learning-based approach is favored for its simplicity and modest requirements. The localization of a node is independent from that of others. Also, past information is useful in the learning procedure and, therefore, this approach is highly suitable for target tracking applications where the information about the target at each time instant is partial or sparse, insufficient for geometry-based techniques to work effectively.

Although the localization accuracy can improve as more training data is available, collecting large training data or having many beacon nodes results in significant processing and communication overhead. A challenge for future research is to reduce this overhead. Also, it would be interesting to make one or more beacon nodes mobile and study how learning can be helpful in such an environment.

REFERENCES

Akaho, S. (2001). A kernel method for canonical correlation analysis. In *Proceedings of the International Meeting of the Psychometric Society (IMPS 2001).*

Boser, B. E., Guyon, I. M., & Vapnik V. N. (1992). A training algorithm for optimal margin classifiers. In *5th Annual ACM Workshop on COLT*, 144-152. ACM Press.

Brunato, M. & Battiti, R. (2005). Statistical learning theory for location fingerprinting in wireless LANs. *Computer Networks,* 47(6): 825-845, 2005.

Bulusu, N., Bychkovskiy, V., Estrin, D., & Heidemann, J. (2002). Scalable ad hoc deployable rf-based localization. In *2002 Grace Hopper Celebration of Women in Computing Conference*. Vancouver, Canada.

Capkun, S., Hamdi, M., & Hubauz, J.-P. (2001). Gps-free positioning in mobile ad hoc networks. In *2001 Hawai International Conference on System Sciences,* pp. 9008-.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, Learning, and Games*. Cambridge University Press. ISBN-10 0-521-84108-9, 2006

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273-297, 1995

Chang, C.-C., & Lin, C.-J. (2008). *LIBSVM – A library for Support Vector Machines*. National Taiwan University. URL http://www.csie.ntu.edu.tw/ cjlin/libsvm

Doherty, L., Ghaoui, L. E., & Pister, K. S. J. (2001). Convex position estimation in wireless sensor networks. In *IEEE INFOCOM,* 2001.

Gezici, S., Giannakis, G., Kobayashi, H., Molisch, A., Poor, H., & Sahinoglu, Z. (2005). Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE Signal Processing Magazine*, 22(4): 70-84, 2005.

Hardoon, D. R., Szedmak, S., & Shawe-Taylor, J. (2004). Canonical correlation analysis; an overview with application to learning methods. *Neural Computation*, 16:2639–2664, 2004.

He, T., Huang, C., Blum, B., Stankovic, J., & Abdelzaher, T. (2003). Range-free localization schemes in large scale sensor networks. In *ACM Conference on Mobile Computing and Networking*, pp. 81-95, 2003.

Hotelling, H. (1936). Relations between two sets of variants. *Biometrika*, 28: 321-377, 1936.

Kwon, Y., Mechitov, K., Sundresh, S., Kim, W., & Agha, G. (2004). Resilient localization for sensor networks in outdoor environments. Tech. rep., University of Illinois at Urbana-Champaign, 2004.

Kuh, A., Zhu, C., & Mandic, D. P. (2006). Sensor network localization using least squares kernel regression. In *Knowledge-Based Intelligent Information and Engineering Systems*, pp. 1280-1287, 2006

Kuh, A., & Zhu, C. (2008). Sensor network localization using least squares kernel regression. *Signal Processing Techniques for Knowledge Extraction and Information Fusion.* Mandic D. et al., Editors, 77-96, Springer, April 2008.

Law, M. H. C., & Jain, A. K. (2006). Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 28(3):377–391, 2006

Lee, J. –Y., & Scholtz, R. Ranging in a dense multipath environment using an UWB radio link. *IEEE Journal on Selected Areas in Communications*, 20(9): 1677-1683, 2002.

Letchner, J., Fox, D., & LaMarca, A (2005). Large-Scale Localization from Wireless Signal Strength. In *Proc. of the National Conference on Artificial Intelligence (AAAI),* pp. 15-20, 2005.

Meertens, L., & Fitzpatrick, S. (2004). The distributed construction of a global coordinate system in a network of static computational nodes from inter-node didstances. Tech. rep., Kestrel Institute, 2004.

Moore, D., Leonard, J., Rus, D., & Teller, S. (2004). Robust distributed network localization with noisy range measurements. In *ACM Sensys*, pp. 50-61. Baltimore, MA, 2004.

Nagpal, R., Shrobe, H., & Bachrach, J. (2003). Organizing a global coordinate system from local information on an ad hoc sensor network. In *International Symposium on Information Processing in Sensor Networks*, pp. 333-348, 2003.

Nguyen, X., Jordan, M. I., & Sinopoli, B. (2005). A kernel-based learning approach to ad hoc sensor network localization. *ACM Transactions on Sensor Networks*, 1: 134-152, 2005.

Niculescu, D., & Nath, B. (2003a). Ad hoc positioning system (aps) using aoa. In *IEEE INFOCOM,* 2003.

Niculescu, D., & Nath, B. (2003b). Dv based positioning in ad hoc networks. *Telecommunication Systems*, *22*(1-4), 267–280, 2003.

Oh, S., Sastry, S., & Schenato, L. (2005). A Hierarchical Multiple-Target Tracking Algorithm for Sensor Networks. In Proc. *International Conference on Robotics and Automation*, 2005.

Pan, J. J., Kwok, J. T., & Chen, Y. (2006). Multidimensional Vector Regression for Accurate and Low-Cost Location Estimation in Pervasive Computing. *IEEE Transactions on Knowledge and Data Engineering*, 18(9): 1181-1193, 2006.

Pan, J. J., Yang, Q., & Pan, J. (2007). Online Co-Localization in Indoor Wireless Networks by Dimension Reduction. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-07),* pp. 1102-1107, 2007.

Priyantha, N., Chakraborty, A., & Balakrishnan, H. (2000). The cricket location-support system. In *ACM International Conference on Mobile Computing and Networking (MOBICOM),* pp. 32-43.

Priyantha, N., Miu, A., Balakrishnan, H., & Teller, S. (2001). The cricket compass for context-aware mobile applications. In *ACM conference on mobile computing and networking (MOBICOM),* pp. 1-14, 2001.

Priyantha, N. B., Balakrishnan, H., Demaine, E., & Teller, S. (2003). Anchor-free distributed localization in sensor networks. In *ACM Sensys,* pp. 340-341, 2003.

Priyantha, N. B., Balakrishnan, H., Demaine, E., & Teller, S. (2005). Mobile-Assisted Localization in Wireless Sensor Networks. In *IEEE INFOCOM*. Miami, FL.

Priyantha, N. B. (2005b). *The Cricket Indoor Location System*. Ph.D. thesis, Massachussette Institute of Technology, 2005.

Savarese, C., Rabaey, J., & Beutel, J. (2001). Locationing in distributed ad-hoc wireless sensor networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing,* pp. 2037-2040. Salt Lake city, UT, 2001.

Savvides, A., Han, C.-C., & Strivastava, M. B. (2001b). Dynamic fine-grained localization in ad hoc networks of sensors. In *ACM International Conference on Mobile Computing and Networking (Mobicom)*, pp. 166–179. Rome, Italy, 2001.

Savvides, A., Park, H., & Srivastava, M. (2002). The bits and flops of the n-hop multilateration primitive for node localization problems. In *Workshop on Wireless Networks and Applications (in conjunction with Mobicom 2002)*, pp. 112-121. Atlanta, GA, 2002.

Shang, Juml, Zhang, & Fromherz (2003). Localization from mere connectivity. In *ACM Mobihoc,* pp. 201-212, 2003.

Scholkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT Press, Cambridge, MA, 2002.

Stoleru, R., Stankovic, J. A., & Luebke, D. (2005). A high-accuracy, low-cost localization system for wireless sensor networks. In *ACM Sensys,* pp. 13-26. San Diego, CA, 2005.

Tran, D. A., & Nguyen, T. (2006). Support vector classification strategies for localization in sensor networks. In *IEEE Int'l Conference on Communications and Electronics,* 2006.

Tran, D. A., & Nguyen, T. (2008). Localization in Wireless Sensor Networks based on Support Vector Machines. *IEEE Transactions on Parallel and Distributed Systems*, 19(7): 981-994, July 2008.

Tran, D. A., & Nguyen, T. (2008b). Hop-count based learning techniques for passive target tracking in sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics*, submitted, 2008.

Whitehouse, C. (2002). *The design of calamari: an ad hoc localization system for sensor networks*. Master's thesis, University of California at Berkeley, 2002.

Zhu, C., & Kuh, A. (2007). Ad hoc sensor network localization using distributed kernel regression algorithms. In *Int'l Conference on Acoustics, Speech, and Signal Processing*, 2: 497-500, 2007.

Duc A. Tran is an Assistant Professor in the Department of Computer Science at the University of Massachusetts at Boston, where he leads the Network Information Systems Laboratory (NISLab). He received a PhD degree in Computer Science from the University of Central Florida (Orlando, Florida) in 2003. Dr. Tran's interests are in the areas of computer networks and distributed systems, particularly in support of information systems that can scale with both network size and data size. The results of his work have led to research grants from the US National Science Foundation, a Best Paper Award at ICCCN 2008, and a Best Paper Recognition at DaWak 1999. Dr. Tran has engaged in many professional activities. He has been a Guest-Editor for two international journals, a Workshop Chair, a Program Vice-Chair for AINA 2007, a PC member for 20+ international conferences, and a referee and session chair for numerous journals/conferences.

XuanLong Nguyen is a postdoctoral researcher at Duke University's Department of Statistical Science. He received his Master's degree in statistics and PhD degree in computer science from the University of California, Berkeley in 2007. Dr. Nguyen is interested in learning with large-scale spatial and nonparametric models with applications to distributed and adaptive systems in computer science, and modeling in the environmental sciences. He is a recipient of the 2007 Leon O. Chua Award from UC Berkeley for his PhD research, the 2007 IEEE Signal Processing Society's Young Author best paper award, an outstanding paper award from the ICML-2004 conference.

Thinh Nguyen is an Assistant Professor at the School of Electrical Engineering and Computer Science of the Oregon State University. He received his Ph.D from the University of California, Berkeley in 2003 and his B.S. degree from the University of Washington in 1995. He has many years of experience working as an engineer for a variety of high tech companies. He has served in many technical program committees. He is an associate editor of the IEEE Transactions on Circuits and Systems for Video Technology, the IEEE Transactions on Multimedia, the Peer-to-Peer Networking and Applications. His research interests include Network Coding, Multimedia Networking and Processing, Wireless and Sensor Networks.