LOCALIZATION PROBLEM IN SENSOR NETWORKS: THE MACHINE LEARNING APPROACH

Chapter in book “Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking” (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@gmail.com
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

LOCALIZATION PROBLEM IN SENSOR NETWORKS: THE MACHINE LEARNING APPROACH

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 imply about the sensor locations. In this chapter, we present a fundamentally different approach that is based on machine learning. Under this approach, we work directly on the natural (non-Euclidean) coordinate systems provided by the sensor devices. The known locations of a few sensors in the network and the sensor readings can be exploited to construct signal-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 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 sensors' 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 initial 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 beacons 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. For instance, pair-wise distance can be estimated based on 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)]. 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 converged to a "global rigid" state 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 imply about the sensor location. 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 (2007)]. The main insight of these methods is that the topology implicit in sets of sensor readings and locations can be exploited to in the construction of possibly non-Euclidean signal-based function spaces that are useful for the prediction of unknown sensor locations, as well as other extrinsic quantities of interest. Specifically, one can assume a set of sensors with known locations, which are called the 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 sensors that are of previously unknown positions.

Consider a sensor *S* whose true (unknown) position is *(x, y)* on a 2-D field. There are more than one way we can learn. For example, we can model the localization problem as a classification problem [Nguyen et al. (2005), Tran & Nguyen (2006), Tran & Nguyen (2007)]. Indeed, we can define a set of classes (e.g., *A*, *B*, and *C* as in Figure 1), which are geographic regions chosen appropriately in the sensor 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 classification procedure outputs that S is a member of class *A*, of *B*, and of *C*, then *S* must be in the intersection $A \cap B \cap C$.

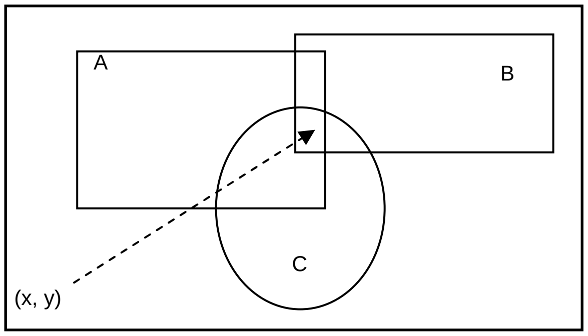

Figure 1 If we can define a set of classes which are geographic regions, a sensor's location can be estimated based on its memberships in these classes

We can also solve the localization problem as a regression problem [Pan et al. (2006)]. We can use a regression tool to learn about the 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 lengths 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 that *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 nor external assisting devices is 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 (2007)], which can be obtained easily at no 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, 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 output 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 the sensors, 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 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 length is denoted by $hc(S_i, S_j)$. 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 $hc(S_j, S_i)$.

We assume the existence of $k < N$ beacon nodes $\{S_1, S_2, ..., S_k\}$ that know their own location. When the network is small enough that any sensor can hear from these beacons, we denote by $ss(S_i, S)$ the signal strength a sensor S receives from each beacon S_i. Nevertheless, without specified otherwise, we assume that each node can communicate to a beacon not necessarily direct but via a multi-hop path. This assumption is applicable to more types of sensor networks.

We need to devise an algorithm each remaining node $\{S_{k+1}, S_{k+2}, ..., S_N\}$ can use to estimate its location. Before we present its details, 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-dimension hop-count vector

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

   or, for the case the network is small, the k-dimension signal-strength vector

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

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 broadcast to all sensors 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 $S_j \in \{S_{k+1}, S_{k+2}, ..., S_N\}$ computes the following k-dimension hop-count vector

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

   or, for the case the network is small, the k-dimension signal-strength vector

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

   The sensor then applies the prediction model it has obtained previously to this hop-count (or signal-strength) vector to estimate the sensor'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)]. The general steps are as follows:

- Class definition: Define a set of classes $\{C_1, C_2, ..., C_m\}$, each class C_i being a geographical region in the sensor field
- Training data: Because the beacon locations are known, the membership of each beacon node in each class C_i is known. The hop-count (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 which is used to output, for each given sensor S and class C_i, the membership of S in class C_i. As a result, we can determine the location of S. To solve the classification problem, an efficient tool is Support Vector Machines (SVM). A brief background on SVM is presented below and then how it is used for sensor localization.

SVM Classification

Consider the problem of classifying data in a data space X into either class G or not. Suppose that each data point x has a feature vector x' in some feature space $X' \subset R^n$. We are given k data points $x_1, x_2, ..., x_k$, called the training points, with labels $y_1, y_2, ..., y_k$, respectively (where $y_i = 1$ if $x_i \in G$ and -1 otherwise). We need to predict whether a new data point x is in G or not.

Support Vector Machines (SVM) [Cortes & Vapnik (1995)] is an efficient method to solve this problem. For the case of finite data space (e.g., location data of nodes in a sensor network), the steps typically taken in SVM are as follows:

- Define a kernel function $K: X \times X \to R$. This function must be symmetric and the $k \times k$ matrix $[K(x_i, x_j)]$ $(i, j \in [1, k])$ must be positive semi-definite (i.e., has non-negative Eigen values)
- Maximize the function

$$W(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{k} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

  subject to

$$\sum_{i=1}^{k} y_i \alpha_i = 0 \text{ and } 0 \leq \alpha_i \leq C \text{ for } i \in [1, k]$$

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

$$h_K(x) = \sum \alpha_i^* y_i K(x, x_i) + b^*$$

According to Mercer's theorem [cf., Scholkopf & Smola (2002)], there exists a feature space X' where the kernel K defined above is the inner product of X' (i.e., $K(x, z) = x' \circ z'$ for every $x, z \in X$). The function $h_K(.)$ represents the hyperplane in X' that maximally separates the training points in X (G-points in the positive side of the plane, $not(G)$-points in the negative side). Under standard assumptions in statistical learning, the SVM is known to yield bounded (and small) classification error when applied to test data.

The main property of the SVM is that it only needs 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, that uniquely and accurately identify its location. However, we can provide a similarity measure between two sensors 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 classes $\{C_1, C_2, ..., C_m\}$. For example, as illustrated in [Nguyen et al. (2005)], each class C_i can be an equi-size disk in the sensor area such that any point in the sensor field must be covered by at least three such disks. Thus, after the learning procedure, if a sensor S is found to be a member of three classes C_i, C_j, and C_k, the location of S is estimated as the centroid of the intersection $C_i \cap C_j \cap C_k$.

Using the disk partitioning method, or any method that defines classes as overlapping regions, the number of classes in the learning procedure could be high. Alternatively, [Tran & Nguyen (2006), Tran & Nguyen (2007)] propose the LSVM technique which partitions the sensor field using a fixed number of classes, thus bounding the learning cost. Hereafter, unless specifically mentioned, the technique we describe is LSVM. Let $M = 2^m$. LSVM defines $(2M-2)$ classes as follows (illustrated in Figure 2):

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

We need to solve $(2M-2)$ binary classification problems. Each problem, corresponding to a class cx_i (or cy_i), outputs a SVM prediction model that decides whether a sensor belongs to this class or not. If the SVM learning predicts that a node S is in class cx_i but not class cx_{i+1}, and in class cy_j but not class cy_{j+1}, 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 localization error for node S is at most $D/(M\sqrt{2})$. However, every SVM is subject to some classification error, and so we should maximize the probability that *S* is classified into its true cell, and, in case of misclassification, minimize the localization error.

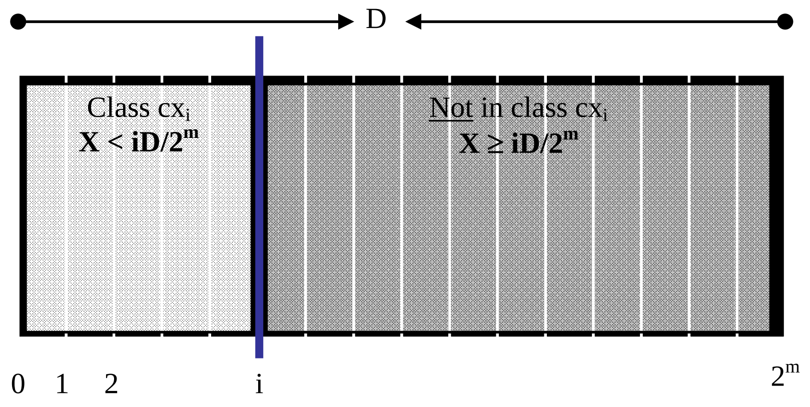

Figure 2 Definition of class cx_i

Kernel Function

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

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

where γ is a constant to be computed during the cross-validation phase of the training procedure, and h_i the hop-count vector of sensor S_i. More examples for the kernel function are discussed in [Nguyen et al. (2005)].

Training Data

For each binary classification problem (for a class $c \in \{cx1, cx2, ..., cx_{M-1}, cy_1, cy_2, ..., cy_{M-1}\}$), the training data is the set of values $\{y_1, y_2, ..., y_k\}$, where $y_i = 1$ 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 sensor *S* belongs to class *c*:

$$h_K(S) = \sum \alpha_i^* y_i K(S, S_i) + b^*$$

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 $(i, y_i\alpha_i^*)$. This information is called the SVM model information. This model information is used to predict the location of any sensor given its hop-count vector.

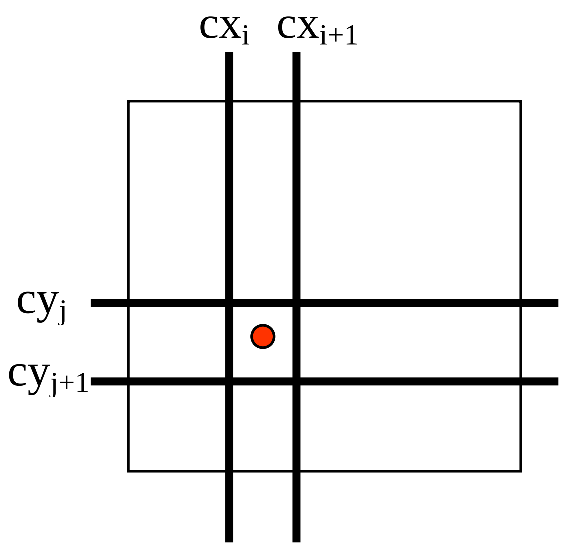

Figure 3 Localization of a sensor based on its membership in regions cx_i and cy_j

Location Estimation

Let us focus on the classification along the X-dimension. LSVM organizes the x-classes into a binary decision tree, illustrated in Figure 4. Each tree node is an x-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 we traverse the tree in the order {*left-child*$\rightarrow$ *parent*$\rightarrow$ *right-child*}, the result is the ordered list $cx_1 \rightarrow cx_2 \rightarrow ... \rightarrow cx_{M-1}$. Given this decision tree, each sensor *S* can estimate its x-coordinate using the following algorithm:

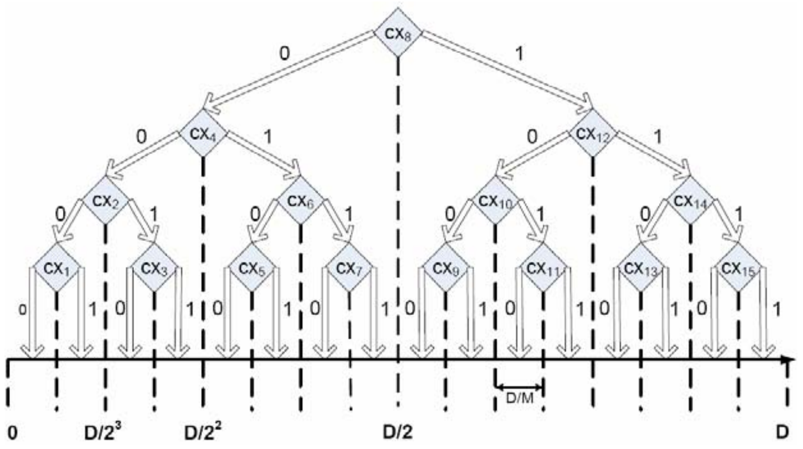

Figure 4 Decision tree: m = 4

Algorithm: X-dimension Localization
Estimate the x-coordinate of sensor 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) RETURN $x'(S) = (i - 1/)D/M$
   - 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
```

Similarly, a decision tree is built for the Y-dimension classes and each sensor *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, localization of a node requires visiting log_2M 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 how close we want to localize a sensor.

SVM is subject to error and so is LSVM. A misclassification with respect to a class *G* occurs when SVM predicts that a sensor is in *G* but in fact it is not or predicts that the sensor is not in *G* but it actually is. In [Tran & Nguyen (2007)], it is shown that for a uniformly distributed sensor field, the location error expected for any node is bounded by

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

where ε is the worst-case SVM classification error. This error expectation is decreased with smaller ε. Figure 5 plots the location error expectation E^u according to various SVM error ε. There exists a choice for *m* (no larger than 8) that minimizes the error expectation. In real implementation, it is recommended that we use this optimal *m*. A nice property of SVM is that ε is typically bounded and under certain assumptions on the choice of the kernel function, the bound diminishes if the training size is sufficiently large. In the evaluation study of [Tran & Nguyen (2007)], 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 shows that the SVM approach can offer good accuracy.

LOCALIZATION BASED ON REGRESSION

Trilateration is a geometrical technique that can locate an object based on its distances from three or more other objects. In our case, to locate a sensor we do not know its true distances from the *k* beacon nodes. We can use a regression tool (e.g., *libsvm* [Chang & Lin (2008)]) to learn about these distances based on 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 $i, j \in [1, k]$

Once this regressor *f* is computed, it is broadcast to all the sensors. Since each sensor receives a HELLO message from each beacon, the sensor knows its hop-count distances from all the beacons and can apply the regressor *f* to compute its location. A similar approach, but applied on signal strength data, was considered by Kuh et al (2006).

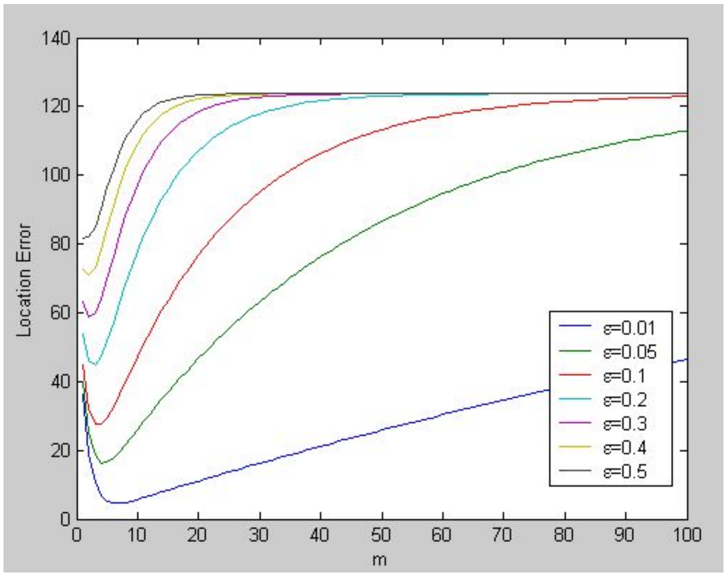

Figure 5 Upper bound on the expectation of worst-case location error under various values of SVM classification error. A lower SVM error corresponds to a lower-appearing curve

When the network is sufficiently small, each sensor can hear from all the beacon nodes. It is observed that if two sensors S_i and S_j receive similar signal strengths from the beacons, they 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) [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) [Hardoon et al. (2004)]. Suppose that there are two sets of multidimensional variables, $s = (s_1, s_2, \ldots, s_k)$ and $l = (l_1, l_2, \ldots, l_k)$. CCA finds two canonical vectors, w_s and w_l, one for each set such that the correlation between these two sets under the projections, $(w_s \circ s_1, w_s \circ s_2, \ldots, w_s \circ s_1)$ and $(w_l \circ l_1, w_l \circ l_2, \ldots, w_l \circ l_k)$, is maximized. While CCA only exploits linear relationship between *s* and *l*, its extension using kernels KCCA can work with non-linear relationships. KCCA defines two kernels, $K_s(.)$ for the *s* space and $K_l(.)$ for the *l* space. Each kernel K_s (or K_l) represents implicitly a feature vector space Φ_s (or Φ_l) for the corresponding variable *s* (or *l*). Then, a mapping that maximizes the correlation between *s* and *l* in the feature space is found using the kernel functions only (requiring no knowledge about Φ_s and Φ_l).

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

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

and a Matern kernel $K_l(.)$ for the location space

$$K_l(l_i, l_j) = \frac{2(\sqrt{v}w\|l_1 - l_2\|)^v}{\Gamma(v)} K_v(2\sqrt{v}w\|l_i - l_j\|)$$

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 $(s_1, l_1), (s_2, l_2), \ldots, (s_k, l_k)$, where (s_i, l_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 T canonical vectors $P_1, P_2, \ldots, P_T$ are found. A sensor $S \in \{S_{k+1}, S_{k+2}, \ldots, S_N\}$ is localized as follows:

- Computer the signal-strength vector s of sensor S: $s = (ss(S_1, S), ss(S_2, S), \ldots, ss(S_k, S))$
- Compute the projection of $P(s) = (P_1(s), P_2(s), \ldots, P_T(s))$
- Choose from the set of beacon nodes m nodes S_i whose projections $P(s_i)$ are nearest $P(s)$. The distance metric used is a weighted Euclidean distance where the weights are obtained from the KCCA training procedure.
- Compute the location for S as the mean position of these m neighbors

APPLICATION TO TARGET TRACKING

An appealing feature of the learning approach is that the localization of a sensor can be done independently from that of another sensor. The training procedure involves the beacon nodes only, whose result is a prediction model any sensor can use to localize itself without knowledge about other sensors. This feature is suitable for target tracking in a sensor network where to save cost not every sensor needs to run the localization algorithm; only the target needs to be localized. For example, consider a target tracking system with k beacon nodes at known locations deployed. When a target T occurs in an area, and is detected by a sensor S_T, the detecting sensor reports the event to the k beacon nodes. The hop-count 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 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 the sensor (or target) under localization may change depending on the location of the sensor (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 sensor reports the event to a few, possibly random, beacon nodes. The 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 learning 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)] addresses 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 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 approach is favored for its simplicity and modest requirements. 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

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

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

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

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

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

[Chang & Lin (2008)] 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 et al. (2001)] Doherty, L., Ghaoui, L. E., & Pister, K. S. J. (2001). Convex position estimation in wireless sensor networks. In *IEEE Infocom*.

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

[He et al. (2003)] He, T., Huang, C., Blum, B., Stankovic, J., & Abdelzaher, T. (2003). Rangefree localization schemes in large scale sensor networks. In *ACM Conference on Mobile Computing and Networking*.

[Kwon et al. (2004)] 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.

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

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

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

[Meertens & Fitzpatrick (2004)] 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.

[Moore et al. (2004)] Moore, D., Leonard, J., Rus, D., & Teller, S. (2004). Robust distributed network localization with noisy range measurements. In *ACM Sensys*. Baltimore, MA.

[Nagpal et al. (2003)] 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*.

[Nguyen et al. (2005)] 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.

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

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

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

[Pan et al. (2007)] 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)*

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

[Priyantha et al. (2001)] 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)*.

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

[Priyantha et al. (2003)] Priyantha, N. B., Balakrishnan, H., Demaine, E., & Teller, S. (2003). Anchor-free distributed localization in sensor networks. In *ACM Sensys*.

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

[Savarese et al. (2001)] 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*. Salt Lake city, UT.

[Savvides et al. (2001)] Savvides, A., Han, C.-C., & Strivastava, M. B. (2001). 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.

[Savvides et al. (2002)] 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)*. Atlanta, GA.

[Shang et al. (2003)] Shang, Juml, Zhang, & Fromherz (2003). Localization from mere connectivity. In *ACM Mobihoc*.

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

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

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

[Tran & Nguyen (2007)] Tran, D. A., & Nguyen, T. (2007). Localization in Wireless Sensor Networks based on Support Vector Machines. *IEEE Transactions on Parallel and Distributed Systems*.

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