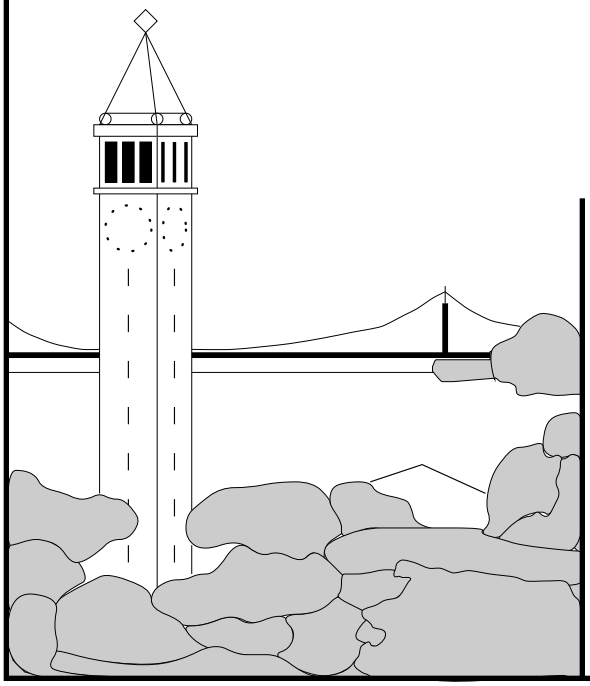
A kernel-based learning approach to ad hoc sensor network localization

XuanLong Nguyen†, Michael I. Jordan†‡ and Bruno Sinopoli†
{xuanlong,jordan,sinopoli}@eecs.berkeley.edu
†Department of Electrical Engineering and Computer Science
‡ Department of Statistics
University of California, Berkeley, CA 94720



Report No. UCB/CSD-04-1319

April 2004

Computer Science Division (EECS)
University of California
Berkeley, California 94720

A kernel-based learning approach to ad hoc sensor network localization

XuanLong Nguyen†, Michael I. Jordan†‡ and Bruno Sinopoli†
{xuanlong,jordan,sinopoli}@eecs.berkeley.edu
†Department of Electrical Engineering and Computer Science
‡ Department of Statistics
University of California, Berkeley, CA 94720

April 2004

Abstract

We show that the coarse-grained and fine-grained localization problems for ad hoc sensor networks can be posed and solved as a pattern recognition problem using kernel methods from statistical learning theory. This stems from an observation that the kernel function, which is a similarity measure critical to the effectiveness of a kernel-based learning algorithm, can be naturally defined in terms of the signal strength received by the sensors. Thus we work in the natural coordinate system provided by the physical devices. This not only allows us to sidestep the difficult ranging procedure required by many existing localization algorithms in the literature, but also enables us to derive a simple and effective localization algorithm. The algorithm is particularly suitable for networks with densely distributed sensors, most of whose locations are unknown. The computations are initially performed at the base sensors and the computation cost depends only on the number of base sensors. The localization step for each sensor of unknown location is then performed locally in linear time. We present an analysis of the localization error bounds, and provide an evaluation of our algorithm on both simulated and real sensor networks. [1]

1 Introduction

A sensor network can be viewed as a distributed pattern recognition device. In the pattern recognition approach, rather than transforming sensor locations and sensor readings into Euclidean, world-centric coordinates, we work directly with the (non-Euclidean) coordinate system given by the physical sensor readings themselves. These readings can be viewed as approximations to "kernel functions," and the function space that they induce can be viewed as a covariate space for the prediction of various extrinsic quantities of interest, using any of a variety of statistical algorithms for regression and classification. In the current paper we illustrate this approach in the setting of a localization problem (Hightower & Borriello, 2000; Bulusu et al., 2000; Savarese et al., 2002).

The localization problem that we study is that of determining the location of a (large) number of sensors of unknown location, based on the known location of a (small) number of base sensors. Our algorithm makes no assumptions regarding the topology of the network. Let $X_1, \ldots, X_m$ denote a set of m sensors, and let x_i denote the position in $\mathbb{R}^2$ of sensor X_i. Suppose that the locations of the first n sensors are known, i.e., $X_1 = x_1, \ldots, X_n = x_n$, where $n \ll m$. We want to recover

[1]This research is supported in part by the grant DAAD19-02-1-0383.

the positions of $X_{n+1}, \ldots, X_m$ solely on the basis of the receive/transmit signals $s(x_i, x_j)$ between pairs of sensors.

An important characteristic of radio or light signal strength is the relationship of the signal attenuation as a function of distance (Seidel & Rappaport, 1992). For instance, for radio signal in an idealized environment, given that the sending and receiving antennas are focused on the same radio frequency, we have:

$$s \propto Pd^{-\alpha}, \tag{1}$$

where $\alpha \in [0.1, 2]$, and P is the sending signal voltage. This relationship forms the basis for a large number of localization algorithms, which consist of two main steps: (1) a ranging procedure involves estimating the distance from a sensor to another sensor based on the signal strength of the signals transmitted/received between the two, and (2) a procedure that recovers the locations of the sensors based on their pairwise distance estimates either by triangulation or by least square error methods(Priyantha et al., 2000; Girod & Estrin, 2001; Savvides et al., 2001; Whitehouse, 2002). Unfortunately, this idealized model can be highly inaccurate due to variability caused by multipath effects and ambient noise interference as well as device-specific factors such as the frequencies of node radios, physical antenna orientation, and fluctuations in the power source (Bulusu et al., 2000; Priyantha et al., 2000).

In this paper we propose a method that bypasses the ranging step altogether. We show that it is possible to pose a coarse-grained localization problem as a discriminative classification problem that can be solved using tools from the statistical machine learning literature. Fine-grained localization is then achieved by a second application of the coarse-grained localization technique. Our localization algorithm thus involves two phases. First, there is a training phase that chooses discriminant functions for classifying positions using arbitrarily constructed boundaries. This phase is performed either on-line at the base stations, or taken off-line, and takes $O(n^3)$ computational time, where n is the number of base sensors. Hence, our assumption is that the base sensors have sufficient power and processing capability (indeed, these are also the nodes that might have GPS-capability to determine their own exact locations). Second, once the training phase is completed, other location-unknown low-power sensors can determine their own position locally, and the computation takes only $O(n)$ for each of these sensors.

Our approach makes use of kernel methods for statistical classification and regression (Scholkopf & Smola, 2002), an example of which is the "support vector machine (SVM)." Central to this method is the notion of *kernel function*, which provides a generalized measure of similarity for any pair of sensor locations. The SVM finds a *discriminant function*—a weighted sum of kernel functions, one for each sensor. In the simplest implementation, we simply equate the kernel function with the signal reading received by each of the sensors; this reading may not necessarily be a positive semidefinite function as required to define a kernel function, but in some cases it may be a reasonable approximation. We also consider other implementations that build kernel functions on top of the signal readings.

Our focus is on the discriminative classification problem of locating sensors in an ad hoc sensor network setting. It is worth noting that similar methods have been explored recently in the context of tracking one or more objects (e.g., mobile robots) that move through a wireless sensor field. [2] Systems of this type include Active Badge (Want et al., 1992), (Ward et al., 1997), RADAR (Bahl & Padmanabhan, 2000), Cricket (Priyantha et al., 2000), and UW-CSP (cf. (Li et al., 2002)).

[2] The alternative to discriminative classification is classification using *generative* probabilistic models. This is a well-explored area that dates back to contributors such as Wiener and Kalman. Recent work in this vein focuses on the distributed and power-constrained setting of wireless sensor networks (e.g. (Sheng & Hu, 2003; D'Costa & Sayeed, 2003)).

In (Bahl & Padmanabhan, 2000), the authors describe a simple nearest neighbor classification algorithm to obtain coarse localization of objects. Most closely related to our approach is the work of (Li et al., 2002) in which a number of classification algorithms are used for tracking moving vehicles, including k-nearest neighbor and support vector machines. We elaborate more on the connections between this work and ours in the description of our algorithm.

The paper is organized as follows. We begin with a brief background of classification using the kernel method, and motivate our application of kernel methods to the localization problem based on sensor signal strength. Next, the localization algorithm and its error analysis are described. We then present details of the implementation of the algorithm and its computational cost, followed by an evaluation of our algorithm with simulated and real sensor networks. Finally, we present our conclusions in the final section.

2 Classification using kernel methods

In a classification algorithm, we are given as our training data n samples $(x_i, y_i)_{i=1}^n$ in $\mathcal{X} \times \{\pm 1\}$, where $\mathcal{X}$ denotes the input space. Intuitively, each y_i specifies whether the data point $x_n \in \mathcal{X}$ lies in a class $C \subseteq \mathcal{X}$ (if $y_i = 1$) or not. A classification algorithm involves finding a discriminant function $y = \text{sign(f(x))}$ that minimizes the classification error $P(Y \neq \text{sign(f(X))})$.

Central to a kernel-based classification algorithm (e.g. the SVM) is the notion of a kernel function $K(x, x')$ that provides a measure of similarity between two data point x and x' in $\mathcal{X}$. Technically, K is required to be a symmetric positive semidefinite kernel.[3] For such a function, Mercer's theorem implies that there must exist a feature space $\{\Phi(x)|x \in \mathcal{X}\}$ in which K acts as an inner product, i.e., $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$. The SVM and related kernel-based algorithms then choose a linear function in this feature space $f(x) = \langle \text{w}, \Phi(x) \rangle$ for some w such that $||\text{w}|| \leq B$ for some constant B and minimizes the training error

$$\sum_{i=1}^{n} \phi(y_i f(x_i)).$$

Here ϕ denotes an appropriate loss function that is convex and is an upper bound for the 0-1 loss $\mathbb{I}(y \neq \text{sign(f(x))})$.[4] For instance, the SVM algorithm is based on the hinge loss $\phi(yf(x)) = (1 - yf(x))_+$.[5] In also turns out that f can be expressed directly in terms of the kernel function K:

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x). \tag{2}$$

There are a large number of kernel functions that satisfy the positive semidefinite property required by the SVM algorithm. Examples include the Gaussian kernel:

$$K(x, x') = \exp -(||x - x'||^2/\sigma)$$

as well as the polynomial kernel:

$$K(x, x') = (\gamma + ||x - x'||)^{-\sigma},$$

[3] For a translation-invariant kernel, i.e., $K(x, x') = h(x - x')$ for some function h, K is a positive semidefinite kernel if the Fourier transform of h is non-negative.

[4] The indicator function is defined as $\mathbb{I}(A) = 1$ if A is true, and 0 otherwise.

[5] The subscript + notation means that $x_+ = \max(x, 0)$.

for some exponent parameter σ and constant γ. Both of these kernel functions decay with respect to the increase in the distance $||x - x'||$, a key property that is shared by most signal strength models. In particular, the radio signal model (1) has a similar form to that of a polynomial kernel. In (Sheng & Hu, 2003), the authors justified the use of an acoustic energy model for localization that has the form of the Gaussian kernel above. This motivates the idea of using the signal strength directly to construct an appropriate positive semidefinite kernel function, which is then used to find a discriminant function for localization purposes. Finally, since we are only interested in the value of $K(x, x')$ at the training and test data points x_i for $i = 1, \ldots, m$, we only need to ensure that the Gram matrix $K = (K(x_i, x_j)_{ij}$ is a positive semidefinite matrix.

The use of a kernel method also makes it straightforward to incorporate multi-modal signals. Indeed, suppose that we ahve D types of sensory signals, each of which can be used to define a kernel function $K_d(x, x')$ for $d = 1, \ldots, D$. Then any conic combination of K_d yields a new positive semidefinite kernel:

$$K(x, x') = \sum_{d=1}^{D} \beta_d K_d(x, x').$$

There are methods for choosing the parameters $\beta_d > 0$ based on empirical data (Lanckriet et al., 2004).

3 Localization in ad hoc sensor network

3.1 Algorithm description

We assume that there is a large number of sensors densely deployed in a geographical area. The input to our algorithm is a set of m sensors, denoted by $X_1, \ldots, X_m$. For each i we denote by x_i the position in $\mathbb{R}^2$ of sensor X_i. Suppose that the first n sensor locations are known, i.e., $X_1 = x_1, \ldots, X_n = x_n$, where $n \ll m$. For every pair of sensors X_i and X_j, we are given the signal $s(x_i, x_j)$ that sensor X_j receives from X_i. We want to recover the positions of $X_{n+1}, \ldots, X_m$.

We first aim to obtain a coarse location estimate for $X_{n+1}, \ldots, X_m$. Given an arbitrarily constructed region $C \subseteq \mathbb{R}^2$, we ask whether $X_i \in C$ or not, for $i = n+1, \ldots, m$. This can be easily formulated as a classification problem. Indeed, since the location of the base sensors $X_1, \ldots, X_n$ are known, we know whether or not each of these base sensors are in C. Hence we have as our training data n pairs of $(x_i, y_i = \text{sign}(\text{x}_\text{i} \in \text{C}))_{\text{i=1}}^{\text{n}}$. For any sensor X_j, $j = n+1, \ldots, m$, we can predict whether $X_j \in C$ or not based on the sign of the discriminant function $f(x_j)$ of the form:

$$f(x_j) = \sum_{i=1}^{n} \alpha_i K(x_i, x_j). \tag{3}$$

We emphasize that the value of $f(x_j)$ is known because the values of the kernels, $K(x_i, x_j)$, are known, despite the fact that we do not know the position x_j per se.

Next, we turn to the definition of the kernel matrix $K = (K(x_i, x_j))_{1 \le i,j \le m}$. There are many possibilities. Here we list a few simple alternatives:

1. Simply define $K(x_i, x_j) = s(x_i, x_j)$. We call this a *first-tier* kernel. This implicitly assumes that the matrix $S = (s(x_i, x_j))_{1 \le i,j \le m}$ is a Gram matrix (i.e., symmetric positive semidefinite). In practice, this might not be true for the real signal matrix, but it may be a reasonable approximation.

2. Alternatively, define $K = S^T S$, to be refered to as a *second-tier* linear kernel. K is always symmetric positive semidefinite. Another interpretation is that K is an inner product of the Euclidean feature space $\{\Phi(x)\}$, which is defined at the data point x_i as:

$$\Phi(x_i) = (s(x_i, x_1), s(x_i, x_2), \ldots, s(x_i, x_m)).$$

3. Finally, it is also possible to evaluate any positive semidefinite kernel (e.g., Gaussian) on the Euclidean feature space $\{\Phi(x)\}$. This yields a symmetric positive semidefinite matrix, to be refered to as a *second-tier* (Gaussian) kernel.

Our classification formulation has several unique characteristics. The training points correspond to the base sensors, and thus may be limited in number, making the learning problem nominally a difficult one. However, because we are free to choose the region C to learn, the problem can in fact be made easy. This ability to design the geometry of the boundary to fit the geometry of the classifier distinguishes this problem from a traditional pattern recognition problem.

Now we turn to the fine-grained estimate of localization. We use the coarse-grained solution presented above as a sub-routine for a localization algorithm for sensor $X_j (j = n+1, \ldots, m)$. The idea is quite simple: We fix a number of overlapping regions $C_\beta (\beta = 1, \ldots, U)$ in the sensor network. For each β, we formulate a corresponding classification problem with respect to class C_β and predict whether or not $X_j \in C_\beta$. Hence, X_j has to be in the intersection of regions that contain it. We might, for example, assign its location x_j to be the center of such an intersection. Given an appropriate choice of granularity and shapes for the regions C_β, if most of the classification labels are correct we expect to be able to obtain a good estimate of x_j.

As will be shown by our experiments on both simulated data (using a Gaussian or polynomial kernel) and real sensor data (using kernels that are constructed directly from the signal matrix), given a sufficient number of base sensors (i.e., training data points), the SVM algorithm can fit regions of arbitrary shape and size with high accuracy. When the number of base sensors is limited, it is found that the SVM algorithm can still fit elliptic shapes very well. This can be turned to our advantage for fine-grained localization: By picking appropriate regions C_β such as ellipses that are easy to classify, we do not need many base sensors to achieve good localization for the entire network. In the sequel, we will show that this intuition can be quantified to give an upper bound on the expected (fine-grained) localization error with respect to the number of base sensors.

3.2 Localization error analysis

Suppose that the sensor network of size $L \times L$ is covered uniformly by k^2 discs with radius R. Then any given point in the sensor network is covered by approximately $\pi(Rk/L)^2$ discs. Each of these discs are used to define the region for our region classification problem. To obtain a fine-grained location estimate for all remaining sensors X_j for $j = n+1, \ldots, m$, we need to solve k^2 region classification problems. Let e_β be the training error for each of these problems, for $\beta = 1, \ldots, k^2$. That is,

$$e_\beta = \sum_{i=1}^{n} \phi(\mathrm{sign}(\mathrm{x_i} \in \mathrm{C}_\beta)\mathrm{f}(\mathrm{x_i})).$$

Since the size and shape of the regions are ours to decide, it is reasonable to assume that the training error for these classification problems are very small. For instance, the circle/elliptic shape is particularly suited for Gaussian or polynomial kernels. Define ϵ to be the upper bound for all training errors:

$$\epsilon = \max_{1 \le \beta \le k^2} e_\beta.$$

From statistical learning theory (Vapnik, 1998), for each $\beta = 1, \ldots, k^2$, the probability of misclassification for each new sensor X_j and region C_β is $e_\beta + O(1/\sqrt{n})$, where n is the number of training points (i.e., number of base sensors). Since each location the probability of misclassification for at least one these covering discs is, by the union bound, less than $\frac{\pi R^2 k^2}{L^2}(\epsilon + O(1/\sqrt{n}))$, in which case the localization error is $O(L)$. If a given sensor is correctly classified with respect to all of its covering discs, then we assign the sensor's location to be the center of the intersection of all these discs, in which case localization error is bounded by $O(L/k)$.

Hence, the expectation of the localization error is bounded by

$$O\left(\frac{L}{k}\right) + \frac{\pi R^2 k^2}{L}\left(\epsilon + O(1/\sqrt{n})\right).$$

This asymptotic bound is minimized by letting $k \propto L^{2/3} R^{-2/3} (\epsilon + O(1/\sqrt{n}))^{-1/3}$. The bound then becomes $O(L^{1/3} R^{2/3} (\epsilon + O(1/\sqrt{n}))^{1/3})$. Thus, we have proved:

Proposition 3.1. *Assume that all sensor locations are independently and identically distributed according to an (unknown) distribution. For any sensor location x, let $\hat{x}$ be the location estimate given by our algorithm, then:*

$$\mathrm{E}||\mathrm{x} - \hat{\mathrm{x}}|| \leq \mathrm{O}(\mathrm{L}^{1/3}\mathrm{R}^{2/3}(\epsilon + \mathrm{O}(1/\sqrt{\mathrm{n}}))^{1/3}).$$

If the base sensor locations are linearly separable in the feature space $\{\Phi(x)\}$ for all k^2 discs, i.e. $\epsilon = 0$, the following corollary is then immediate.

Corollary 3.2. *Assume that the base sensor locations are linearly separable in the feature space $\{\Phi(x)\}$ for all k^2 discs, then the localization error is $O(L^{1/3} R^{2/3} n^{-1/6})$.*

This gives a rate of decrease of the localization error as a function of the number of base sensors. Our analysis is rather loose (due to the use of the union bound), and we expect that the rate is much better than $n^{-1/6}$. In addition, for certain kernel-based function classes it is possible to obtain estimation rates better than $O(1/\sqrt{n})$, in which case the localization error rate can also be tightened further.

4 Algorithm details and computational cost

During the training phase associated with each coarse localization sub-routine, i.e., classification with respect to a fixed region C_β, we construct the training data set based on the locations of the base sensors as described in the previous section. At the system level, this is achieved by having all base stations send the signal matrix entries $s(x_i, x_j)$, and their known locations to a central station. This involves receiving and storing $n^2 + n$ numbers at the central station, which then invokes the SVM algorithm. This requires solving the following optimization problem:

$$\min_{\mathrm{w}} ||\mathrm{w}||^2 + \frac{c}{n}\sum_{i=1}^{n} \phi(y_i f(x_i)).$$

where $f(x) = \langle \mathrm{w}, \Phi(x) \rangle$, $\phi(yf(x)) = (1 - yf(x))_+$, and c is a fixed parameter. This is a convex optimization problem, which has the following dual form (Scholkopf & Smola, 2002):

$$\max_{0 \leq \alpha \leq c} 2\sum_{i=1}^{n} \alpha_i - \sum_{1 \leq i,j \leq n} \alpha_i \alpha_j y_i y_j K(x_i, x_j).$$

It is known that the solution to this optimization problem can be found in the worst case in $O(n^3)$ computational time. Thus if there are k^2 regions to classify, the total training time is $O(n^3k^2)$ in the worst case. However, this estimate is overly conservative in our case. Indeed, an estimate based on the number of support vectors n_s returned by each classification algorithm (those X_i such that $\alpha_i \neq 0$) reveals that the computational complexity is $O(n_s^3 + n_s^2 n)$ instead of $O(n^3)$. Usually $n_s \ll n$. Our simulation experience (to be presented in the next section) shows that when discs with radius R are used, the support vectors reside mostly along the boundary of the discs, hence $n_s \approx O(\min(n\pi R^2/L^2, 2\pi R))$, in which case the overall training phase takes only $O(R^2nk^2)$ time. Note also that this training phase is the most expensive part of our algorithm and is performed at a central station.

Once the training phase completes, each base sensor is required to store the n parameters $(\alpha_1, \ldots, \alpha_n)$ for the classification purpose of the new (location-unknown) sensors. Now, if the first-tier kernel is used, a new sensor X_j for $j = n+1, \ldots, m$ receives the signal $s(x_i, x_j)$ from the n_s base sensors $i \in \{1, \ldots, n\}$, along with the non-zero values α_i, resulting in a $O(n_s)$ cost in time and storage. If a second-tier linear kernel or second-tier Gaussian kernel is used, a new sensor X_j receives n-element signal vectors $(s(x_i, x_1), \ldots, s(x_i, x_n))$ from the n_s base stations $i \in \{1, \ldots, n\}$, resulting in a $O(n_s n)$ cost in time and storage. $K(x_i, x_j)$ is then readily computable from the received signals $s(x_i, x_j)$ in $O(1), O(n), O(n^2)$ time for the first-tier, second-tier linear, and second-tier Gaussian kernel, respectively. Then a simple computation (Equation 3) determines for sensor X_j whether it resides in the region C or not. The attractive feature of the localizing step is that it is done locally (in a distributed fashion), taking only linear (for the first-tier and second-tier linear kernel) or quadratic (for the second-tier Gaussian kernel) time and storage space (in terms of n). Since the localization is done on an as-need basis, its time and storage cost does not depend on the total number of sensors m in the network.

Now we turn to fine-grained localization. At both algorithmic and system levels, this involves invoking the coarse localization sub-routine k^2 times with respect to regions $C_1, \ldots, C_{k^2}$. Therefore, for each region $\beta = 1, \ldots, k^2$ we have a set of parameter $(\alpha_i)_{i=1}^n$. Each sensor X_j can then determine its location by setting x_j to be the center of the intersection of all regions C_β that it finds itself residing in. In the case C_β are all discs with centers c_β, a simple method is to set:

$$x_j := \frac{\sum_{\beta=1}^{k^2} c_\beta \mathbb{I}(X_j \in C_\beta)}{\sum_{\beta=1}^{k^2} \mathbb{I}(X_j \in C_\beta)}.$$

Other methods for determining the shapes of C_β and for estimating x_j are certainly worthy of further investigation.

5 Experiment Results

We evaluate our algorithm on simulated sensor networks in the first two subsections, and then on a real network using Berkeley sensor motes.

5.1 Coarse localization

We consider a network of size 10×10 square units. The base sensors are distributed uniformly in a grid-like structure. There are a total $n = n_0^2$ number of such sensors. We are concerned with recognizing whether a sensor position x, given by the signal reading $s(x_i, x)$ with $i = 1, \ldots, n$, lies in a to-be-defined region C or not.

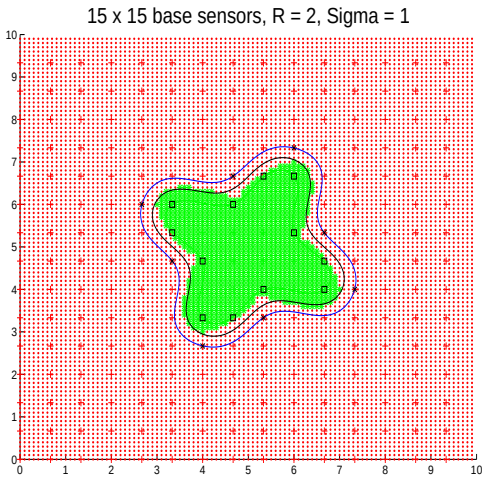


Figure 1: Illustration of a simulated sensor network with 15×15 base sensors, and the recognized boundary (with $R = 2$) using a Gaussian kernel with $\sigma = 1$. See text for a detailed description.

Every pair of sensors with locations x_1 and x_2 are assumed to receive the same signal value from one another given by the following Gaussian kernel: $K(x_1, x_2) = \exp -(||x_1 - x_2||^2/\sigma)$, or polynomial kernel: $K(x_1, x_2) = (1 + ||x_1 - x_2||)^{-\sigma}$. We call σ the decay parameter, and we experiment with different values of σ. The results for the polynomial kernels are similar to the Gaussian kernels, and are not presented in the paper due to space constraints.

We take the signal matrix $(s(x_i, x_j))_{ij}$ as the input to our algorithm, *without* regard to its specific functional relationship to the distances. Different values of exponent parameter σ in either the Gaussian or polynomial kernel reduces to considering different exponent parameters σ_0 for the kernel $K(x, x') = s(x, x')^{\sigma_0}$.

The area to be recognized consists of all locations x that satisfy the following equation:

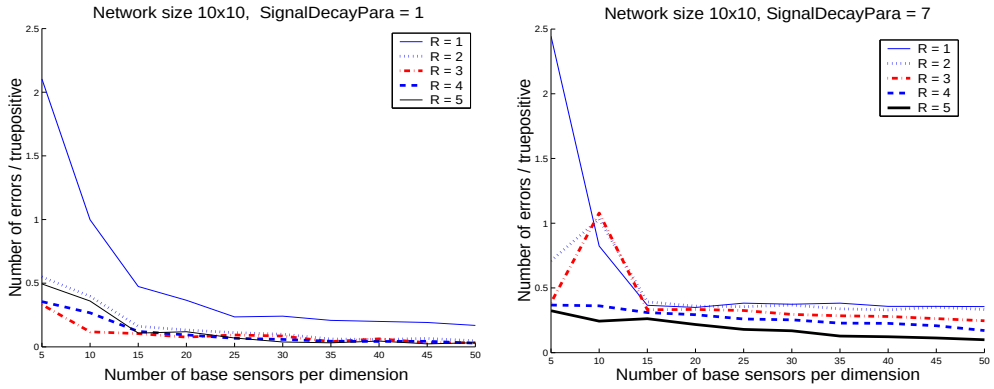
$$(x - v)^T H_1 (x - v) \leq R \text{ and } (x - v)^T H_2 (x - v) \leq R,$$

where $v = [5\ 5]^T$, $H_1 = [2\ \ -1; -1\ \ 1]$ and $H_2 = [2\ 1; 1\ \ 1]$. We experiment with different values of the radius R. Let $C(R)$ denote the area to be recognized.

For each network configuration (n, R, σ), we learn a discriminant function f using the training data given by the base sensor positions. Once f is learned, we test the classification at 100×100 sensor locations distributed uniformly in the network. Figure 1 illustrates the boundaries of the regions learned by our algorithm using Gaussian kernels. These boundaries lie close to the true boundary (the shaded region) of $C(R)$. Clearly, most misclassified points are near the true boundary of $C(R)$.

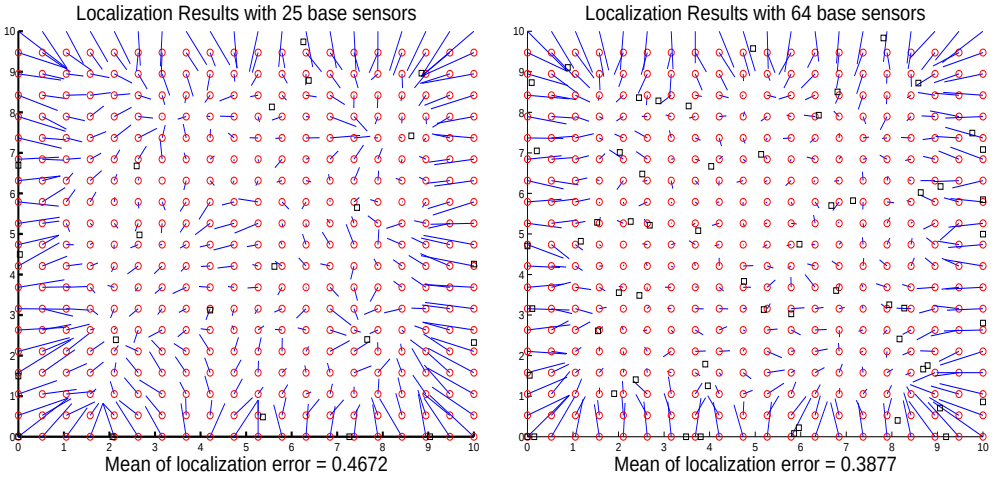
The plots in Figure 2 report the test error with respect to the number of base sensors deployed in the network. The test error is defined to be the ratio between the number of misrecognized points and the number of points located within the area $C(R)$ (out of 100×100 locations distributed uniformly in the grid).

A few observations are in order. First, as we increase the number of base sensors, the test error unsurprisingly goes down. Given a fixed area $C(R)$ parameterized by R, and a fixed signal decay parameter σ, there appears to exist a threshold for the number of base sensors beyond which the



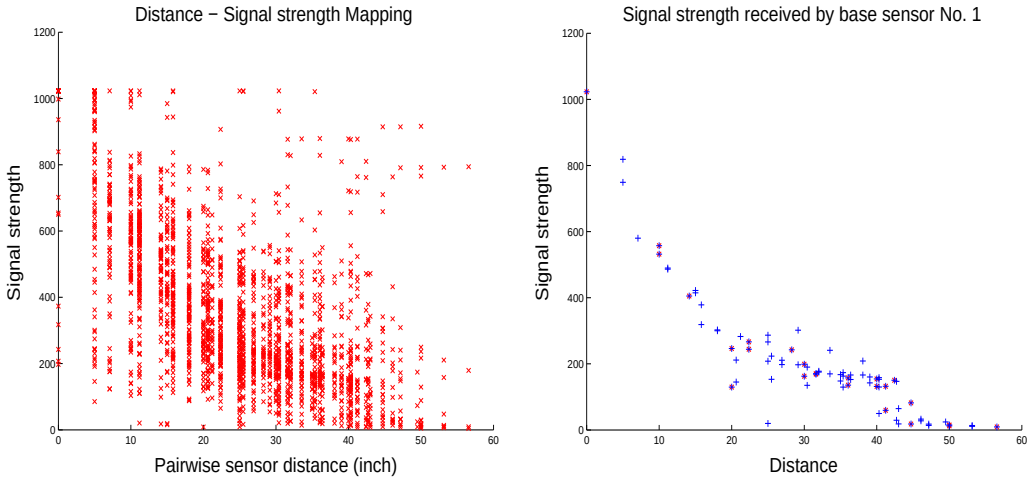
(a) (b)

Figure 2: Simulation results with Gaussian kernels. The x-axis shows the number of sensors employed along each dimension of the network. The y-axis shows the ratio between number of misrecognized points and the number of points inside the area to be recognized. The signal decay parameters are, from left to right, $\sigma = 1$ and $\sigma = 7$ for the Gaussian kernel.

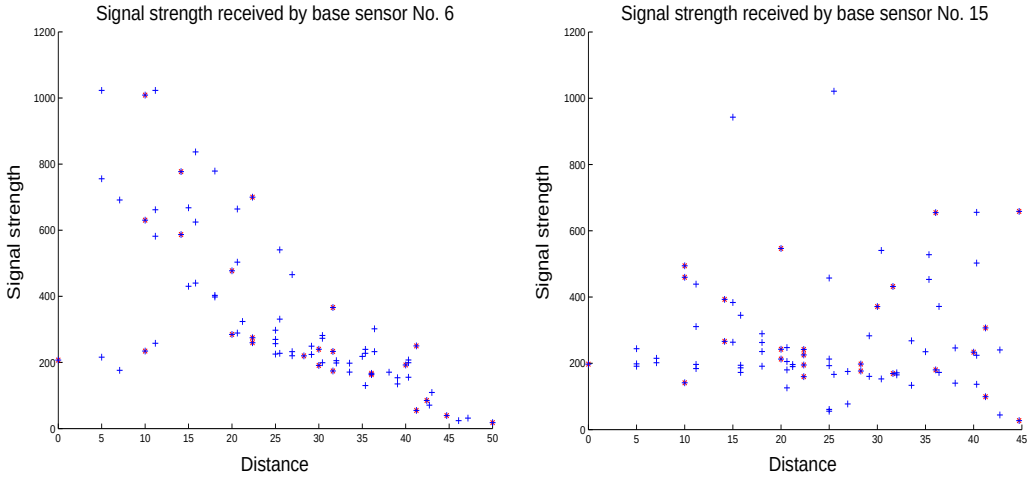


(a) (b)

Figure 3: Localization result on a simulated sensor network of size 10×10 square units with 25 base sensors (left figure) and 64 base sensors (right figure).The base sensors are the black squares. Each blue line connects a true sensor position (in circle) and its estimate. The signal reading is assumed to be a Gaussian kernel function. The mean of localization error is 0.4672 in the left figure and 0.3877 in the right figure.



(a) (b)



(c) (d)

Figure 4: Figure (a) shows the noisy relationship between signal strength received by sensors and the distances. A few sensors such as sensor No. 1 in Figure(b) exhibit a clear signal strength-distance functional pattern, while most other sensors are like No. 6 and No. 15 as shown in Figures (c) and (d). Note that only data points marked with x in red are available for regression training.

n	$R = 2.5, \sigma = 5, k = 15^2$	$R = 5, \sigma = 1, k = 10^2$	$R = 5, \sigma = 5, k = 10^2$
3	3.8483 ± 0.0187	3.8418 ± 0.0368	3.5474 ± 0.0954
4	3.5787 ± 0.0599	2.9052 ± 0.3528	2.2925 ± 0.3060
5	3.0292 ± 0.1311	0.9003 ± 0.2876	0.6813 ± 0.2612
6	2.0536 ± 0.3541	0.4789 ± 0.0165	0.4765 ± 0.0767
7	0.9112 ± 0.3018	0.4380 ± 0.0162	0.4309 ± 0.0179
8	0.4228 ± 0.1642	0.4153 ± 0.0127	0.4045 ± 0.0163
9	0.3107 ± 0.0635	0.4057 ± 0.0107	0.3868 ± 0.0106
10	0.2826 ± 0.0065	0.3992 ± 0.0093	0.3748 ± 0.0083

Table 1: Localization error (mean and standard deviation) for 400 sensors with unknown locations deployed in a sensor network covering an $L \times L$ square unit area with $L = 10$. There are $N = n^2$ base sensors distributed randomly (approximately uniformly) in the $L \times L$ square. We use k^2 discs distributed uniformly in the grid points, with the same radius $R = 2.5$ and $R = 5$. The above results correspond to the use of Gaussian kernel with different settings of σ, k, R. For each setting, we repeat the experiment 20 times to obtain the means and variances of localization error.

test error is not improved further. This is because the number of support vectors (i.e. those X_i with $\alpha_i \neq 0$) does not increase any further. Our ability to fit the area $C(R)$ depends not only on the number of base sensors, but also on the sensitivity of the signals received from these sensors (determined by the parameter σ). In addition, if we need to recognize a particular area, we only need to plant base sensors in the area around the boundary, because these are the likely locations of support vectors. Of course, in our context coarse-grained localization is only a sub-routine for fine-grained localization, and it is our interest to have base sensors spread throughout the whole geographical area.

The second observation relates to the radius R. As R gets larger, the test error gets smaller. This is because for small R, the number of base sensors residing in the region is small, which results in a disproportionate representation of the training data. The third observation concerns the effect of the signal decay parameter σ. If σ is small for the Gaussian kernel (and large for the polynomial kernel), i.e., the signal dies down very quickly for distant sensors, the SVM algorithm can fit areas of almost arbitrary shapes. As σ gets large (or respectively small for the polynomial kernel), the discriminant function f fits a mostly elliptic shape. We observe that as σ gets large (i.e., the sensors can transmit very far), the optimal threshold for the number of base sensors increases. In addition, the optimal test error becomes less sensitive to the size of area $C(R)$. The observations above are helpful in determining the best size and shape of the recognized regions to be used in fine-grained localization.

5.2 Fine-grained localization

Table 1 summarizes the localization results for various settings of a simulated sensor network of size $L \times L$, where $L = 10$ units. In this network, there are n^2 base sensors, which are distributed approximately uniformly at random in the whole area. By this we mean that each base sensors is initially planted at a grid point in the $L \times L$ square, and then perturbed by Gaussian noise $N(0, L/(2n))$. There are 400 other sensors whose locations are to be determined using our algorithm. These 400 sensors are simply deployed uniformly in the grid. Again, we assume the signal strength follows a Gaussian kernel.

Our error analysis and the simulation results with coarse localization in the previous subsection suggests that given each network setting, it is possible to optimize the setting of the regions C_β for

accurate localization. However, we only wish to demonstrate that a robust setting is possible that still gives good localization results, and we simply use k^2 discs uniformly distributed on the grid. Thus, the only parameters to set are k and the radius R of the disc. Since we expect to have a small number of base sensors, it is important to have R sufficiently large so that the training data are well represented in each classification problem for each region. We chose $R = L/2 = 5$ and $R = L/4 = 2.5$ in our experiments. In this case, k is sufficiently large to cover the whole network. The experiment results show that increasing k tends to give a better localization estimate, at the cost of computational time (see Table 1).

5.3 Localization with Berkeley sensor motes

We evaluated our algorithm on a real sensor network using Berkeley tiny sensor motes (Mica motes) as the base stations, each of which is equipped with a light sensor.

Our algorithm was applied to estimate the positions of light sources given the light signal strength received by the 25 base sensors placed 10 inches apart on a 5×5 grid. Only the position of light sources placed at the base sensors are given as training data, while other positions are to be estimated. Three different kernels are used in our algorithm. The first one is a first-tier symmetric positive semi-definite approximation of the signal matrix, while the second and third kernel are second-tier linear and Gaussian kernels defined on feature vectors constructed from the signal matrix. For fine-grained localization, coarse localization is repeatedly applied for circles of radius $R = L/2 = 20$ inches that cover part of the network area. These circle centers are five inches apart in both dimensions.

We compared our algorithm to a state-of-the-art algorithm that epitomizes a majority of localization algorithms in the literature. This algorithm was described in (Whitehouse, 2002), and consists of two main steps: (1) a ranging procedure aimed at establishing a mapping between the signal strength received by a base sensor and the distance to the light source, and (2) a localization procedure giving the distance estimates using least squares methods.

Figure 4 illustrates the difficulty of the ranging problem due to the very noisy functional relationship between the distances between light source positions and sensors, and signal strengths received by the sensors. Much of this noise is device-specific. As shown in Figure 4, a few sensors exhibit a clear distance-to-signal-strength pattern, while most others exhibit a very noisy pattern. As shown in (Whitehouse, 2002), improvement in the ranging step can be achieved by accounting for specific base sensors. This is done by introducing regression coefficients for each of these base sensors. Once the ranging step is completed, we have the distance estimates between the base sensors and the light source's positions. The initial position estimates are obtained using the Bounding-Box algorithm, and then iteratively updated using a least squares method (see (Whitehouse, 2002; Savvides et al., 2001)).

Figure 5 shows that the localization error achieved by our algorithm using three different kernels are significantly lower than that of the two-step algorithm. Among the three choices of signal kernels, the second-tier kernels are much better than the simple first-tier kernel. The mean, median and standard deviation of localization error measured in inches (for all 81 unknown positions) for the 2-step algorithm and ours with 3 different kernels are, respectively, (6.99, 5.28, 5.79), (6.67, 4.60, 7.38), (3.65, 2.51, 4.29), (3.53, 2.63, 3.50).

6 Discussion

We have presented an algorithm for coarse-grained and fine-grained localization for ad hoc wireless sensor networks. Our approach treats the signal strength as measured by sensor motes as a natural

coordinate system in which to deploy statistical classification and regression methods. For the localization problem, this approach avoids the ranging computation, a computation which requires accurate signal models that are difficult to calibrate. Instead, we use signal strength either directly to define basis functions for kernel-based classification algorithms, or indirectly via derived kernels that operate on top of the signal strength measurements. We show how a kernel-based classification algorithm can be invoked multiple times to achieve accurate localization results, and we present an error analysis for the accuracy which can be achieved as a function of base sensor density. Our algorithm is particularly suitable for densely distributed sensor networks, and is appealing for its computational scaling in such networks: The preprocessing computations are performed at the base sensors, which are assumed to have sufficient processing and power capability, while the localizing step that is done at each location-unknown sensor can be achieved in linear time.

We have argued for a simple approach to localization that dispenses with ranging computations and sensor modeling. We do not necessarily believe, however, that our statistical approach is always to be preferred. In particular, the level of accuracy that we appear to be able to obtain with our approach is on the order of half the distance between the motes. While this accuracy is sufficient for many potential applications of sensor networks, in some applications higher accuracy may be required. In this case, ranging-based approaches offer an alternative, but only in the setting in which highly accurate models of the relationship between sensor signals and distances are available.

References

Bahl, P., & Padmanabhan, V. N. (2000). RADAR: An in-building RF-based user location and tracking system. *INFOCOM (2)* (pp. 775–784).

Bulusu, N., Heidemann, J., & Estrin, D. (2000). *GPS-less low cost outdoor localization for very small devices* (Technical Report 00-729). CS Department, University of Southern California.

D'Costa, A., & Sayeed, A. (2003). Collaborative signal processing for distributed classification in sensor networks. *2nd Intl. Workshop on Info. Proc. in Sensor Networks (IPSN).*

Girod, L., & Estrin, D. (2001). Robust range estimation using acoustic and multimodal sensing. *IEEE/RSI Int. Conf. on Intelligent Robots and Systems (IROS).*

Hightower, J., & Borriello, G. (2000). Real-time error in location modeling for ubiquitous computing. *Proceedings of Ubicomp Workshop in Location Modeling for Ubiquitous Computing.*

Lanckriet, G., Cristianini, N., Ghaoui, L. E., Bartlett, P., & Jordan, M. (2004). Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research.*

Li, D., Wong, K., Hu, Y., & Sayeed, A. (2002). Detection, classification, and tracking of target. *IEEE Sig. Proc. Magazine.*

Priyantha, N., Chakraborty, A., & Balakrishnan, H. (2000). The Cricket location-support system. *ACM International Conference on Mobile Computing and Networking.* Boston, MA.

Savarese, C., Rabaey, J., & Langendoen, K. (2002). Robust positioning algorithms for distributed ad-hoc wireless sensor networks. *USENIX Technical Annual Conference, Monterey, CA.*

Savvides, A., Han, C., & Srivastava, M. (2001). Dynamic fine grained localization in ad-hoc sensor networks. *Proceedings of Mobicom.*

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

Seidel, A., & Rappaport, T. (1992). 914MHz path loss prediction models for indoor wireless communications in multi-floored buildings. *IEEE Trans. on Antennas and Prop., 40*.

Sheng, X., & Hu, Y. (2003). Energy based acoustic source localization. *2nd Intl. Workshop on Info. Proc. in Sensor Networks (IPSN)*.

Vapnik, V. (1998). *Statistical learning theory*. New York: John Wiley and Sons Inc.

Want, R., Hopper, A., Falcao, V., & Gibbons, J. (1992). The active badge location system. *ACM Trans. on Info. Sys., 40*.

Ward, A., Jones, A., & Hopper, A. (1997). A new location technique for the active office. *IEEE Personnel Communications*.

Whitehouse, C. (2002). The design of Calamari: an ad-hoc localization system for sensor networks. Master's thesis, EECS Department, University of California, Berkeley.

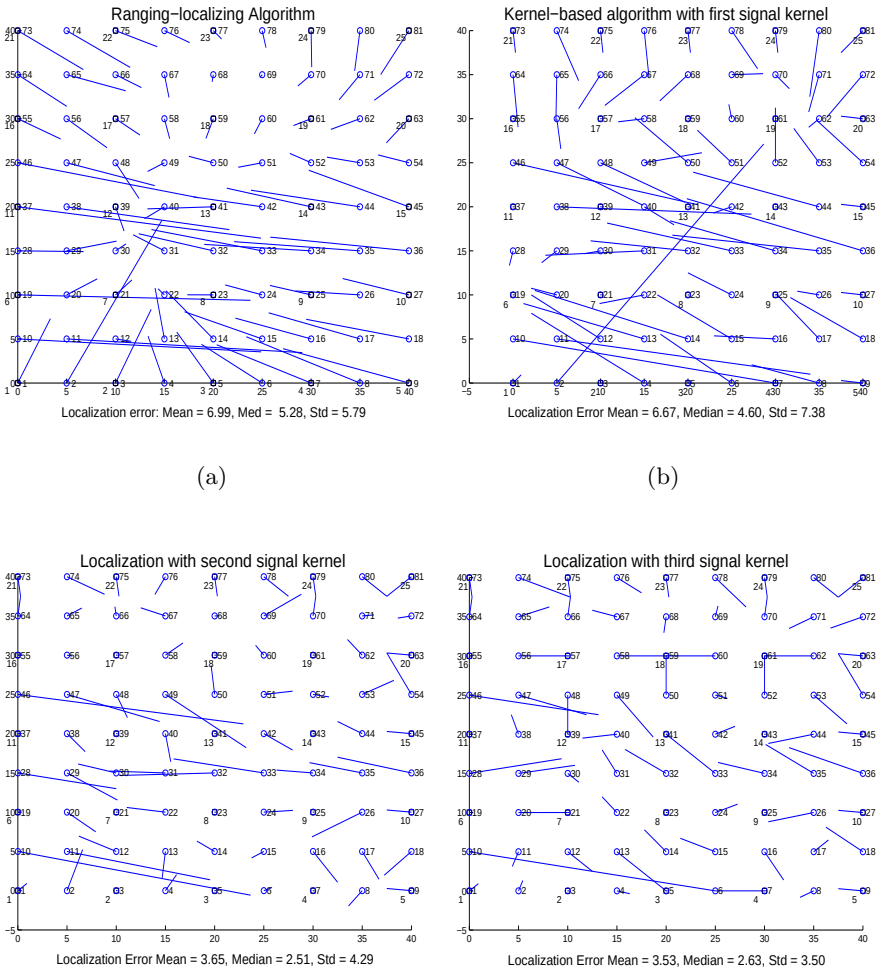


Figure 5: Localization result for a real sensor networks covering a 40×40 inch-square area. There are 25 base sensors (Berkeley motes) spaced in a 5×5 grid. Each line connects a true position (in circle) and its estimate. Figure (a) shows the results given by a traditional 2-step localization algorithm, while Figures (b,c,d) show the localization results obtained by our algorithm using three different kernels (the first-tier, second-tier linear and second-tier Gaussian kernel, respectively constructed on top of the signal matrix).