Sensor Network Localization Using Least Squares Kernel Regression

Anthony Kuh[1], Chaopin Zhu[1], and Danilo Mandic[2]

[1] University of Hawaii
[2] Imperial College London

Abstract. This paper considers the sensor network localization problem using signal strength. Unlike range-based methods signal strength information is stored in a kernel matrix. Least squares regression methods are then used to get an estimate of the location of unknown sensors. Locations are represented as complex numbers with the estimate function consisting of a linear weighted sum of kernel entries. The regression estimates have similar performance to previous localization methods using kernel classification methods, but at reduced complexity. Simulations are conducted to test the performance of the least squares kernel regression algorithm. Finally, the paper discusses on-line implementations of the algorithm, methods to improve the performance of the regression algorithm, and using kernels to extract other information from distributed sensor networks.

1 Introduction

Information gathering is relying more on distributed communication and distributed networking systems. Ad hoc sensor networks are being deployed in a variety of applications from environmental sensing to security and intrusion detection to medical monitoring [1]. These sensor networks are becoming increasingly more complex with sensors responsible for different tasks. We will consider a sensor network consisting of two different types of sensors: base sensors where the locations are known (the base sensors could have GPS) and simple sensors called motes where the locations of the motes are unknown. Assuming all sensors are capable of transmitting information via signal strength we use kernel least squares regression methods to estimate the location of the motes. The kernel regression method performs similarly to kernel classification methods [9] at much reduced complexity. We also discuss an on-line implementation of the algorithm to perform tracking of mobile sensors and ways of extracting other information from sensors using kernel regression methods.

When signal strength is available a common method to perform sensor localization is using ranging information as discussed in [2,6]. Range-based methods commonly use a two step approach when performing localization: first signal distance is estimated between pairs of devices from signal strength and then a localization algorithm is used based on these estimated distances. In wireless radio in ideal space, the signal attenuation s satisfies,

B. Gabrys, R.J. Howlett, and L.C. Jain (Eds.): KES 2006, Part III, LNAI 4253, pp. 1280–1287, 2006.

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

where d is the distance between transmitter and receiver $\eta > 2$ is a constant and P is the transmitting power [13]. Because of rich scattering in the real world, the received signal strength is quite noisy. This makes it more difficult for range-based methods such as [2,6] to get accurate readings of the distance between different devices. Statistical methods such as Maximum Likelihood Estimation (MLE) [10], EM algorithm [12], and Bayesian networks [3] were used to alleviate the scattering effect. These methods usually have high computing complexity.

We use an alternative approach first established in [9] where signal strength information is stored in a kernel matrix. In [9,16] a classification problem is solved to determine whether a sensor lies in a given region $\mathcal{A}$ or not. The classification problem uses training data from base sensors (whose location is known) to learn the parameters $\alpha(i)$ and threshold value b given by the following equation

$$f(x) = \operatorname{sgn}(\sum_{i=1}^{l} \alpha(i) y_i K(x, x_i) + b) \tag{2}$$

where $f(x)$ is the decision function to determine whether $x \in \mathcal{A}$ or not and x_i are input locations and $y_i = 1$ if $x_i \in \mathcal{A}$ otherwise $y_i = -1$. In [9] the classification problem is solved using the Support Vector Machine (SVM) (with hinge loss function) [8,15]. In [16] the classification problem is solved using the Least Squares SVM (LS-SVM) (with quadratic loss function with equality constraints) [14,7]. Fine localization is achieved by performing the classification problem several times with different overlapping regions. A mote's location is estimated from the average of the centroids of each region it belongs to.

This method gives reasonably accurate estimates of locations of motes, but is computationally expensive as the classification problem must be solved for each region considered. This paper estimate locations of motes using one complex least squares kernel regression problem. Sensors are located on a two-dimensional grid with their location represented by a complex number. The parameters $\alpha(i)$ are also complex and the kernels are real. The regression method saves computational costs while giving similar performance to the fine localization algorithm.

The paper is organized as follows. In Section 2 we discuss the ad hoc sensor network model. Section 3 gives a discussion of the least squares kernel regression algorithm, an on-line recursive version of the algorithm, and how signal strength is incorporated into the kernels. Section 4 simulates a sensor network model where sensors are randomly place on a two-dimensional grid. Finally, Section 5 discusses extensions of this work including and on-line implementation of the algorithm so mobile sensors can be tracked, improvements to the kernel regression algorithm, and references to how kernel regression methods are used to extract other sensor information. We also discuss learning in a distributed environment subject to communication and power constraints.

2 Sensor Network Model

Assume that an ad hoc network of size N is deployed in a two-dimensional geographical area $\mathcal{T}$. An integer from 1 to N represents each node (sensor) as its ID. Denote the set of all nodes in the network by $\mathcal{N} = (1, \cdots, N)$. The location of node $i \in \mathcal{N}$ is denoted by x_i. Assume that the first m nodes are *base sensors*. These nodes have more computational capabilities than other nodes and their location is known (i.e. x_i, $1 \leq i \leq M$ are known) as these nodes may be positioned or the nodes may have GPS. A goal is to determine the location of the other $N - M$ nodes called *motes* (i.e. estimate $x_{M+1}, \cdots, x_N$).

Each node is capable of transmitting signal strength to each of the other nodes. As mentioned in the previous section signal strength is often related to distance by equation (1). The signal strength is also contaminated by additive noise and may be affected by terrain conditions. For this paper we assume the same signal strength model as [9,16] given by

$$s(x_i, x_j) = \exp\{-\frac{\|x_i - x_j\|^2}{\Sigma} + V\} \tag{3}$$

where V is a zero mean Gaussian random variable with standard deviation τ.

The information about all signal strengths is stored in a kernel matrix K. Kernel entries are a function of signal strength and are constructed so that K is symmetric and positive semi-definite.

3 Least Squares Subspace Kernel Regression Algorithm

This algorithm is described in more detail in [7]. We describe the basic algorithm followed by a discussion of implementing a recursive on-line version of the algorithm. This section concludes by discussing the kernel regression localization algorithm.

3.1 Least Squares Kernel Subspace Algorithm

We are given m training examples or observations drawn from input $X \in \mathbf{R}^n$ and output $Y \in \mathcal{R}$. The observations at each time $(x_i, y_i), 1 \leq i \leq m$ are independent and can be represented compactly as $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} = [x_1 | \ldots | x_m]$ and $\mathbf{y} = [y_1, \ldots, y_m]^T$. The inputs are transformed from input space to feature space via kernel functions $\phi(x)$ that map inputs from $\mathcal{R}^n$ to feature space $\mathcal{R}^d$. Let $\Phi(\mathbf{x}) = [\phi(x_1) | \ldots | \phi(x_m)]$. The estimate is given by $\hat{Y}(x) = w^T \phi(x) + \mathbf{1}b$ where $w \in \mathcal{R}^d$ is the weight vector, $\mathbf{1}$ is an m vector of 1s, and b is the scalar threshold value.

The weight vector can be expressed as a linear combination of each of the feature vectors. For the standard LS-SVM, [14] the weight vector depends on all training feature vectors. This can be expressed as $w = \Phi(\mathbf{x})\alpha$ where α is an m vector. Each of the training examples x_i associated with a nonzero $\alpha(i)$ is called a support vector. For the standard SVM only a fraction of the training examples

are support vectors as an ϵ - insensitive cost function is used that removes training inputs as support vectors that are close to the zero error solution. For the LS-SVM an external procedure needs to be established to reduce the number of training examples. Methods to intelligently choose training examples are discussed briefly in the next subsection with more detail in [7]. Here we assume that the subset is chosen and we will denote it by $\mathbf{x}_S$ which is a matrix containing $l \leq m$ columns from $\mathbf{x}$. Denote the intelligent feature vectors by $\Phi(\mathbf{x}_S)$.

The optimization problem is a quadratic programming problem with equality constraints and is shown below:

$$\min J(w,b) = \min_{w,b} \frac{1}{2}||w||^2 + \frac{\gamma}{2}||e||^2 \tag{4}$$

subject to

$$e = \mathbf{y} - \Phi(\mathbf{x})w - \mathbf{1}b. \tag{5}$$

and

$$w = \Phi(\mathbf{x}_S)\alpha \tag{6}$$

where now α is an l vector describing the weighting of the training feature vectors. Defining $K_{SS} = \Phi^T(\mathbf{x}_S)\Phi(\mathbf{x}_S)$ and $K_S = \Phi^T(\mathbf{x}_S)\Phi(\mathbf{x})$ and substituting equation (5,6) into (4) we have that

$$\min Q(\alpha, b) = \min \frac{1}{2}\alpha^T K_{SS}\alpha + \frac{\gamma}{2}||Y - {K_S}^T\alpha - \mathbf{1}b||^2. \tag{7}$$

This problem is optimized by finding the solution to the following set of linear equations.

$$\begin{bmatrix} l & \underline{1}^T {K_S}^T \\ K_S\mathbf{1} & K_{SS}/\gamma + K_S{K_S}^T \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{1}^T Y \\ K_S Y \end{bmatrix}. \tag{8}$$

Assuming $A = K_{SS}/\gamma + K_S{K_S}^T$ is invertible, then by elimination we get that

$$b = \frac{\mathbf{1}^T Y - \mathbf{1}^T {K_S}^T A^{-1} K_S Y}{l - \mathbf{1}^T {K_S}^T A^{-1} K_S \mathbf{1}} \tag{9}$$

and

$$\alpha = A^{-1}K_S(Y - \mathbf{1}b) \tag{10}$$

3.2 Recursive Kernel Subspace Least Squares Algorithm

Here we discuss a recursive on-line procedure for updating the algorithm presented in the last subsection. At each update we update the estimate using a windowed recursive least squares algorithm. The basic algorithm can be described as follows:

1. Train parameters on initial set of data using batch or online methods.
2. Get new training data and add to information set.

3. If new data satisfies specified criteria add as a support vector.
4. Selectively prune support vectors.
5. Selectively prune information data.
6. If there is more data go to 2.

Here we consider a fixed window size where If we decide to add a support vector from training data we must also delete a support vector. A simple criteria is used to add and delete support vectors. To add a support vector, the new feature data is tested to see if it can reduce the training error data. The vector is added as a support vector when the training error can be reduced by a prescribed amount. This can be implemented as follows: evaluate the kernel vector between the newest data point and all other l_W data points. Compare to the training error vector e. We normalize each of the vectors to having magnitude one and compute the inner product between the two vector which is the same as computing the cosine of the angle between the two vectors. If the magnitude of the inner products is above a specified threshold value, then the new training data is added as a support vector. This criteria is also used in [4]. During the deletion process we delete the support vector that makes the least contribution to the weight vector.

3.3 Kernel Localization Algorithm

In order to implement the algorithms described in the previous subsections we need to form a kernel matrix from signal strength information. Signal strength information is not symmetric due to additive noise and other impairments. In [9] several different kernel functions are formed. Here we use the following kernel function described by

$$K(x_i, x_j) = \exp\{-\frac{\sum_{t=1}^{M}(s(x_i, x_t) - s(x_j, x_t))^2}{2\sigma^2}\} \tag{11}$$

Here the targets Y are complex numbers representing the locations of the base sensors. We pick the base sensor locations $i_1, \ldots i_l$ that we want as support vectors and then use equation (8) to solve for α and b which are complex valued. We then can estimate the location of mote j by using the following estimate

$$f(x_j) = \sum_{n=1}^{l} \alpha(i_n) K(x_j, x_{i_n}) + b, \quad M+1 \leq j \leq N. \tag{12}$$

There is considerable savings by using the complex kernel regression algorithm as only one regression algorithm is performed to get estimates of the mote locations as opposed to the many kernel classification algorithms necessary to get fine localization in [9,16].

4 Simulations

Several simulations were conducted to test the performance of the kernel regression localization algorithm. Sensors were placed on a 10 by 10 grid. There were

$M = k^2$ base sensors placed on a grid and then the locations were perturbed by additive Gaussian noise. There were 400 motes placed randomly on the grid. Signal strength had additive noise with deviation $\tau = 0.2$. The subspace algorithm was used with $l = 3k$ support vectors. The algorithm worked well with the regularization parameter set at $\gamma = 60$ and the width of the Gaussian kernels was set at $\sigma = 2.7$. Fig. 1 shows how the mean localization error varies as the number of base sensors increases and Σ is varied. Simulations were conducted 100 times for each setting with average curves and standard deviations shown.

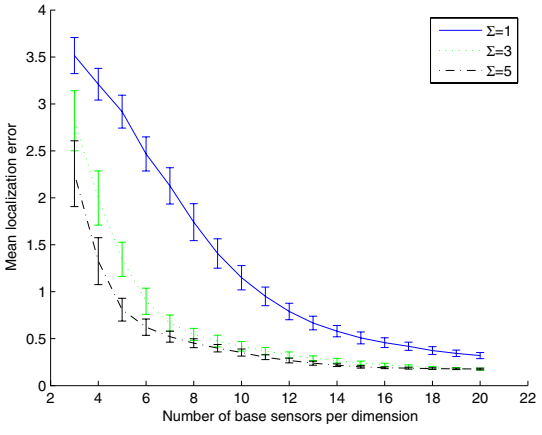

Fig. 1. Mean localization error versus number of base sensors

The plots show that the localization estimation error goes down as the number of base sensors increases. The results are similar to the results for fine localization shown in [9,16]. When $\Sigma = 3, 5$ signal strength decreases rapidly as distance and the error rate remains roughly constant at a little below .5 when more than 100 base sensors are used.

5 Discussion and Extensions

This paper shows that sensor localization can be successfully performed using a least squares kernel subspace regression algorithm. This algorithm has good performance and has computational savings over kernel classification algorithms. There are several further directions to this work.

The sensor localization error is dependent on the number of base sensors. If there are too few base sensors the localization error will be high. In real sensor applications the number of base sensor may be limited. Additional information needs to be considered for the sensor localization algorithm. Note that signal strength information is not used between base sensors and motes in determining the regression estimates. If this information could be incorporated into the kernels this could improve estimation error. In practical applications the exact

location of motes may not be known, but the location may be modelled by a random distribution. This knowledge could be combined with signal strength information to get more accurate estimates of location using kernel methods.

Another consideration are power and communication constraints placed on sensors. If there are constraints on transmission of signal strength information how does this affect localization estimation. If signal strength is weak only nearby base sensors may receive the signal strength. Can good estimation procedures be established when sensors have power and communication constraints.

In Section 3.2 we discussed a recursive on-line least squares kernel subspace algorithm. The kernel localization algorithm can be modified to perform tracking capabilities when sensors are mobile. This is discussed in more detail in [17].

Finally, other information can be gathered from sensors networks using kernel regression algorithms. In [5,11] distributed kernel regression algorithms are used to approximate sensor information such as temperature information. Distributed kernel learning algorithms are proposed to learn this information. Distributed learning could also be applied to develop computationally efficient learning algorithms for sensor localization.

References

1. I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, *IEEE Communications Magazine* 102-114, Aug. 2002.
2. N. Bulusu, J. Heidemann, D. Estrin. "GPS-less low cost outdoor localization for very small devices", Tech. Rep. 00-0729, Computer Science Dept., Univ. of Southern California, 2000.
3. P. Castro, P. Chiu, T. Kremenek, and R. Muntz, "A probabilistic room location service for wireless networked environments", ACM Ubicomp 2001, Atlanta, GA., 2001.
4. B. de Kruif. *Function approximation for learning control, a key sample based approach*, Ph.D. thesis, University of Twente, Netherland, 2004.
5. C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, S. Madden. "Distributed regression; an efficient framework for modeling sensor network data", *Information Processing in Sensor Networks 2004*, Berkeley, CA, Apr. 2004.
6. J. Hightower, G. Borriello, "Real-time error in location modeling for ubiquitous computing, in Location, Modeling for Ubiquitous Computing, 21-27, Ubicomp 2001 Workshop Proceedings, 2001.
7. A. Kuh, "Intelligent recursive kernel subspace estimation algorithms", the 39th Annual Conference of Information Sciences and Systems (CISS 2005), 216-221, Baltimore, MD., 2005.
8. K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An Introduction to Kernel-Based Learning Algorithms. *IEEE Trans. on Neural Networks*, Vol. 12, #2, 181-202, 2001.
9. X. Nguyen, M. Jordan, and B. Sinopoli. A kernel-based learning approach to ad hoc sensor network localization, *ACM Transactions on Sensor Networks*, Vol 1(1), pages 134–152, 2005.
10. N. Patwari, A. Hero, M. Perkins, N. Correat, R. O'Dea. "relative location estimation in wireless sensor networks", *IEEE Transaction on Signal Processing*, vol. 51, no. 8, pp. 2137-2148, Aug. 2003.

11. J. B. Predd, S.R. Kulkarni, H. V. Poor. Distributed Regression in Sensor Networks: Training Distributively with Alternating Projections, *Proceedings of the SPIE Conference and Advanced Signal Processing Algorithms, Architectures, and Implementations XV*, San Diego, CA, Aug., 2005.
12. T. Roos, P. Myllymaki, H. Tirri, "A statistical modeling approach to location estimation", *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, pp, 59-69, Jan.-Mar. 2002.
13. S. Seidel, T. Rappaport (1992), "914MHz path loss prediction models for indoor wireless communications in multifloored buildings", *IEEE Transactions on Antennas and Propagation*, vol. 40, no. 2, pp. 207-217, Feb. 1992.
14. J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least squares support vector machines*, World Scientific Publishing Co., Singapore, 2002.
15. V. Vapnik. *Statistical learning theory.* John Wiley, New York City, NY, 1998.
16. C. Zhu and A. Kuh. "Sensor network localization using pattern recognition and least squares kernel methods", in *Proceedings of 2005 Hawaii, IEICE and SITA Joint Conference on Information Theory, (HISC 2005)*, Honolulu, HI, May 25-27, 2005.
17. C. Zhu, A. Kuh. "Dynamic Ad Hoc network localization using online least squares kernel subspace methods", submitted to *2006 IEEE International Symposium on Information Theory*, Seattle WA., July 2006.