Distributed Data Fusion Using Support Vector Machines

S. Challa
Department of Electrical and Electronic Engineering
The University of Melbourne
Parkville, Victoria 3101 Australia
schalla@ee.mu.oz.au

M. Palaniswami
Department of Electrical and Electronic Engineering
The University of Melbourne
Parkville, Victoria 3101 Australia
swami@ee.mu.oz.au

A. Shilton
Department of Electrical and Electronic Engineering
The University of Melbourne
Parkville, Victoria 3101 Australia
apsh@ee.mu.oz.au

Abstract - *The basic quantity to be estimated in the Bayesian approach to data fusion is the conditional probability density function (CPDF). In recent times, computationally efficient particle filtering approaches are gaining growing importance in estimating these CPDF. In this approach, i.i.d samples are used to represent the conditional probability densities. However, their application in data fusion is severely limited due to the fact that the information is stored in the form of a large set of samples. In all practical data fusion systems that have limited communication bandwidth, broadcasting this probabilistic information, available as a set of samples, to the fusion center is impractical. Support vector machines, through statistical learning theory, provide a way of compressing information by generating optimal kernal based representations. In this paper we use SVM to compress the probabilistic information available in the form of i.i.d samples and apply it to solve the Bayesian data fusion problem. We demonstrate this technique on a multi-sensor tracking example.*

Keywords: Bayesian Data Fusion, Density Estimation, Support Vector Machines, Particle Filters, Sequential Montecarlo Methods.

1 Introduction

The last few years have seen significant advances in the fields of sequential montecarlo methods [1,2], support vector machines and Bayesian data fusion. Sequential montecarlo methods and associated particle filters enable recursive Bayesian estimation in non-linear, non-Gaussian filtering and identification problems.

These methods represent the underlying probability densities with a set of *i.i.d* samples and provide estimates of target identity or target state from them. On the other hand, support vector machines are essentially techniques for function approximation based on statistical learning theory [5]. Recently, SVM's have been shown to perform well for density estimation problems where the PDF of the iid sample set can be learned and the entire sample set can be represented by a few support vectors and the associated kernal functions [4]. Thus while sequential montecarlo methods provide a means of estimating the sample set representing the underlying PDF's recursively, the SVM based density estimation provides methods of compressing the information available via the sample set into a small set of support vectors and the associated Kernal functions. These methods together provide a means of solving the distributed data fusion problem in the Bayesian framework in the Non-Gaussian context.

The paper is organized as follows. Following introduction, section 2 introduces the problem of distributed data fusion problem and highlights the key elements needed for its Bayesian solution. Section 3, introduces the density estimation problem and motivates the use of Support Vector Machines. Section 4 presents the SVM based solution to the density estimation problem while section 5 focuses on the choice of kernal functions and other practical issues. Section 6 provides simulation results for a two dimensional density estimation problem and it presents discussions and future research directions. Finally, conclusions are drawn in section 7.

2 Distributed Data Fusion

Distributed data fusion refers to the problem of fusing information available from remote sensors at the fusion node [3]. The problem is illustrated in figure 1. Bayesian solution to this problem requires the remote nodes to have information available in

$$p(x(k)\,|\,Y_1^k, Y_2^k) =$$

$$p(y_1(k)\,|\,x(k))\,p(y_2(k)\,|\,x(k))\frac{p(x(k)\,|\,Y_1^{k-1}, Y_2^{k-1})}{p(y_1(k), y_2(k)\,|\,Y_1^{k-1}, Y_2^{k-1})}$$

the form of the probability density function and can be expressed as follows:

$$p(x(k)|Y_1^k, Y_2^k) = \frac{p(x(k)|Y_1^k)}{p(x(k)|Y_1^{k-1})} \frac{p(x(k)|Y_2^k)}{p(x(k)|Y_2^{k-1})} \frac{p(x(k)|Y_1^{k-1}, Y_2^{k-1})}{p(y_1(k), y_2(k)|Y_1^{k-1}, Y_2^{k-1})}$$

If the measurements are available along with their likelihood functions, the first two terms can be easily evaluated. However, when the measurements are not readily available, then the following alternate expression can be used:

Thus the most important element of carrying out the fusion is the determination of the following fractions based on the information supplied by the remote sensors.

$$\frac{p(x(k)|Y_1^k)}{p(x(k)|Y_1^{k-1})} \text{ and } \frac{p(x(k)|Y_2^k)}{p(x(k)|Y_2^{k-1})}$$

Most of the existing methods of applying Bayesian data fusion to distributed sensor networks assume that the available probabilistic information, i.e., the numerator and denominator of the density function (from remote sensors) are Gaussian and hence can be represented by only two parameters (i.e., mean and covariance). However, in many real world problems, Gaussianity is far from reality and has given rise to a number of approaches to adapt to non-gaussian PDFs. Particle filters or

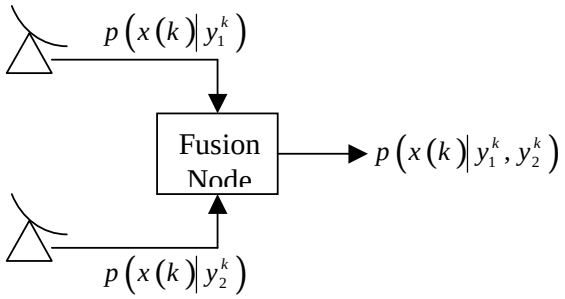

Figure 1.

sequential montecarlo methods are a set of techniques that address the problem of non-Gaussianity effectively. In such methods, the remote sensors use a particle filter that obtains an *iid* sample set that captures the non-gaussian elements of the PDF accurately. However, communicating this PDF to the fusion node is non-trivial, as it involves, broadcasting the complete particle or sample set to the fusion node. As the links connecting the remote sensors to the fusion node have band width constraints, communicating the sample set is impracticle. This creates a need for some of compression. Based in the results of vapnik et. al, [4], we chose support vectors machines as a means of achieving such a compression. It involves learning the PDF from the available sample set and communicating the support vectors in the place of the samples and reconstructing the density function at the fusion node. This problem is studied in the rest of the paper.

3 The Density Estimation Problem

Suppose we are given a set of training data:

$$\{\mathbf{x}_i | \mathbf{x}_i \in \Re^d, i = 1, 2, \ldots, n\} \quad (2.1)$$

generated by taking samples from some unknown probability distribution $P(\mathbf{x})$. We wish to estimate the density function $p(\mathbf{x})$ associated with this distribution. In the context of Bayesian data fusion the densities of interest are

$$p(x(k)|Y_1^k) \quad \text{and} \quad p(x(k)|Y_2^k)$$

where the training samples are obtained from the respective particulate representations. These densities are obtained and stored in the form of a set of samples at each sensor. Using the density estimation approach out-lined in this paper, we hope to encode the information contained in the sample set into a small set of parameters. In such a context, the density can be communicated to the fusion node via these parameters thus providing a method of solving the Bayesian data fusion problem adequately.

The density function is related to the cumulative density function by:

$$F(\mathbf{x}) = \Pr(\mathbf{X} \le \mathbf{x}) = \int_{-\infty}^{x^1} \int_{-\infty}^{x^2} \ldots \int_{-\infty}^{x^{d_L}} p(\mathbf{x}) dx^{d_L} \ldots dx^2 dx^1 \quad (2.2)$$

Using our sample set, we can construct an empirical cumulative distribution function and the related empirical density function as follows:

$$F_l(\mathbf{x}) = \frac{1}{l} \sum_{i=1}^{l} \theta(\mathbf{x} - \mathbf{x}_i)$$

$$\therefore p_l(\mathbf{x}) = \frac{1}{l} \sum_{i=1}^{l} \delta(\mathbf{x} - \mathbf{x}_i) \quad (2.3)$$

where: $\delta(\mathbf{x} - \mathbf{x}_i) =$ Dirac-delta function

Clearly, the empirical density function as defined here is unsatisfactory. The difficulty is that the problem of finding $p(\mathbf{x})$ from $F(\mathbf{x})$ using (2.2) is an

ill-posed problem. As we have just seen, small errors in the distribution function can lead to large errors in the resultant density function. One way to overcome this difficulty is to use regularization techniques to smooth our distribution function prior to finding our density function. The SV regression techniques allow us to do just this and are considered in the next section.

4 Support Vector Machines for Density Estimation

The general non-linear regression problem may be stated as follows. Given a set of training pairs:

$$\left\{(\mathbf{x}_i, z_i) \middle| \mathbf{x}_i \in \Re^d, z_i \in \Re, i = 1, 2, \ldots, n\right\} \quad (2.4)$$

where $\mathbf{x}_i$ is sampled from some unknown probability distribution $P(\mathbf{x})$ and z_i is generated by some unknown function:

$$\hat{f} : \Re^d \to \Re \quad (2.5)$$

and possibly corrupted by noise, and a class of functions:

$$F = \left\{ f \middle| f : \Re^d \to \Re \right\} \quad (2.6)$$

we want to find the function $f \in F$ that minimises the risk functional:

$$R[f] = \int l\left(\hat{f}(\mathbf{x}) - f(\mathbf{x})\right) dP(\mathbf{x}) \quad (2.7)$$

where l is the loss function. Unfortunately, we do not know $P(\mathbf{x})$, so we are unable to calculate the actual risk. We can only calculate the empirical risk:

$$R_{emp}(f) = \tfrac{1}{l} \sum_{i=1}^{l} l\left(z_i - f(\mathbf{x}_i)\right) \quad (2.8)$$

It would be unwise to attempt to minimise the empirical risk directly, as this would likely lead to overfitting and bad generalisation properties. In order to avoid this, it is usual to add a capacity control or *regularisation* term, $\Omega[f]$, which leads us to the regularised risk functional:

$$R_{reg}[f] = R_{emp}[f] + C\Omega[f] \quad (2.9)$$

The constant C is called the regularisation constant. It controls the trade-off between capacity minimisation and empirical risk minimisation. It is usually selected using some form of error cross-validation.

The loss function l controls how training errors are penalised. The de-facto standard loss function for SVM work is Vapnik's ε-insensitive loss function:

$$l\left(z_i - f(\mathbf{x}_i)\right) = \max\left(0, \left|z_i - f(\mathbf{x}_i)\right| - \varepsilon_i\right) \quad (2.10)$$

The motivation behind this choice is the expected existence of noise in our measurement of z_i. As can be seen from figure 2, the parameter ε_i may be used to make the empirical risk insensitive to small errors due to noise. Thus the regularised risk functional has a degree of noise insensitivity.

The regularisation term $\Omega[f]$ is usually chosen to maximise the margin of separation. Unfortunately, in this case this leads to unacceptable constraints being imposed on our choice of function class F.

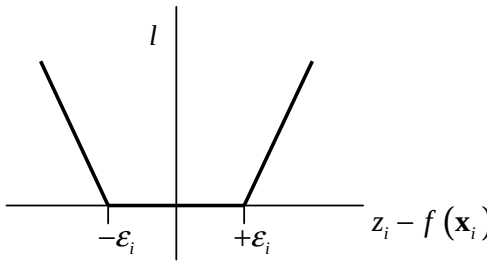

Figure 2: The ε-insensitive loss function.

Motivated by the usual dual form the F in the max-margin case, we will consider the function set:

$$f(\mathbf{x}) = \sum_{i=1}^{l} \sum_{j=1}^{k} \left(\alpha_{ij} - \alpha'_{ij}\right) K_j(\mathbf{x}, \mathbf{x}_i) + b \quad (2.11)$$
$$\alpha_{ij} \geq 0, \ \alpha'_{ij} \geq 0$$

The functions $K_j(\mathbf{x}, \mathbf{x}_i)$ are called the kernel functions. Following Vapnik's paper, we will be using the following regularisation term:

$$\Omega[f] = \sum_{i=1}^{l} \sum_{j=1}^{k} w_j \left(\alpha_{ij} + \alpha'_{ij}\right) \quad (2.12)$$

The terms w_i allow us to favor some kernels over others. A set of kernel functions:

$$\left\{K_j(\mathbf{x}, \mathbf{x}_i) \middle| K_j(\bullet, \mathbf{x}_i) : \Re^d \to \Re, j = 1, 2, \ldots, k\right\} \quad (2.13)$$

is called a kernel dictionary. Under these assumptions, the risk minimisation problem (2.9) becomes a linear programming problem of the form (2.14). A training point $\mathbf{x}_i$ is called a *support vector* if any one element of the set $\{\alpha_{ij}\} \cup \{\alpha'_{ij}\}$ is non-zero. Thus (2.11) may be expressed solely in terms of support vectors. Typically, these support vectors make up only a small fraction of all training set. Hence SV regression may be thought of as a form of compression.

$$\text{minimise:} \quad \sum_{i=1}^{l}\sum_{j=1}^{k} w_j\left(\alpha_{ij}+\alpha'_{ij}\right)+C\sum_{i=1}^{l}\xi_i+C\sum_{i=1}^{l}\xi'_i$$

$$\text{such that:} \quad \sum_{m=1}^{l}\sum_{j=1}^{k}\left(\alpha_{mj}-\alpha'_{mj}\right)K_j\left(\mathbf{x}_i,\mathbf{x}_m\right)+b\le y_i+\varepsilon_i+\xi'_i$$

$$\sum_{m=1}^{l}\sum_{j=1}^{k}\left(\alpha_{mj}-\alpha'_{mj}\right)K_j\left(\mathbf{x}_i,\mathbf{x}_m\right)+b\ge y_i-\varepsilon_i-\xi_i$$

$$\alpha_{ij}\ge 0,\ \ \alpha'_{ij}\ge 0,\ \ \xi_i\ge 0,\ \ \xi'_i\ge 0$$

$$i=1,2,\ldots,l,\quad j=1,2,\ldots,l \qquad (2.14)$$

Two characteristics of this technique make it ideal are:

1. The approximation given by the SV regressor may be expressed solely in terms of a small number of *support vectors*. Thus the training set is compressed in that we may discard all non-support vectors without losing any information.
2. The density function can be expressed using only the parameters of the regression function.

Essentially, the SV density estimation approach allows us to approximate a sample set using a small subset of that sample set that none-the-less gives the characteristics of the density function from which the samples were drawn.

Following Vapnik's method [4], we train our SV regressor using the triples:

$$\left(\mathbf{x}_1,F_l\left(\mathbf{x}_1\right),\varepsilon_1\right),\left(\mathbf{x}_2,F_l\left(\mathbf{x}_2\right),\varepsilon_2\right),\ldots,\left(\mathbf{x}_l,F_l\left(\mathbf{x}_l\right),\varepsilon_l\right) \qquad (2.15)$$

$$\text{where: } \varepsilon_i=\left(1+\delta\right)\sqrt{\tfrac{1}{l}F_l\left(\mathbf{x}_i\right)\left(1-F_l\left(\mathbf{x}_i\right)\right)}$$

The resulting distribution function will have the form of (2.11). It follows that:

$$p\left(\mathbf{x}\right)=\sum_{i=1}^{l}\sum_{j=1}^{k}\left(\alpha_{ij}-\alpha'_{ij}\right)\kappa_j\left(\mathbf{x},\mathbf{x}_i\right) \qquad (2.16)$$

$$\text{where } \kappa_j\left(\mathbf{z},\mathbf{x}_i\right)=\frac{\partial^{d_L}}{\partial z^1\partial z^2\ldots\partial z^{d_L}}K_j\left(\mathbf{z},\mathbf{x}_i\right)$$

The set of functions:

$$\left\{\kappa_j\left(\mathbf{x},\mathbf{x}_i\right)\middle|\,\kappa_j\left(\bullet,\mathbf{x}_i\right):\Re^d\to\Re,\ j=1,2,\ldots,k\right\} \qquad (2.17)$$

are called the cross-kernel dictionary associated with the kernel dictionary (2.13). However, due to the following constraints:

1. Because $p\left(\mathbf{x}\right)\ge 0$, we must set $\alpha'_{ij}=0$ and insist that our kernel dictionary (2.13) consists only of monotonically increasing functions.
2. Assuming that all training data is drawn from $\mathbf{x}\in[\mathbf{a},\mathbf{b}]$, we add the constraints $F\left(\mathbf{a}\right)=0$ and $F\left(\mathbf{b}\right)=1$ for obvious reasons.

we rewrite (2.14) as follows:

$$\text{minimise:} \quad \sum_{i=1}^{l}\sum_{j=1}^{k} w_j\alpha_{ij}+C\sum_{i=1}^{l}\xi_i+C\sum_{i=1}^{l}\xi'_i$$

$$\text{such that:} \quad \sum_{m=1}^{l}\sum_{j=1}^{k}\alpha_{mj}K_j\left(\mathbf{x}_i,\mathbf{x}_m\right)+b\le y_i+\varepsilon_i+\xi'_i$$

$$\sum_{m=1}^{l}\sum_{j=1}^{k}\alpha_{mj}K_j\left(\mathbf{x}_i,\mathbf{x}_m\right)+b\ge y_i-\varepsilon_i-\xi_i \qquad (2.18)$$

$$\sum_{m=1}^{l}\sum_{j=1}^{k}\alpha_{mj}K_j\left(\mathbf{a},\mathbf{x}_m\right)+b=0$$

$$\sum_{m=1}^{l}\sum_{j=1}^{k}\alpha_{mj}K_j\left(\mathbf{b},\mathbf{x}_m\right)+b=1$$

$$\alpha_{ij}\ge 0,\ \ \xi_i\ge 0,\ \ \xi'_i\ge 0$$

$$i=1,2,\ldots,l,\quad j=1,2,\ldots,l$$

In the density estimation problem, the parameters w_j are often used to favour wider distributions over narrower distributions in order to penalise overfitting without leading to underfitting.

The resulting approximation of the density function will have the form:

$$p\left(\mathbf{x}\right)=\sum_{i=1}^{l}\sum_{j=1}^{k}\alpha_{ij}\kappa_j\left(\mathbf{x},\mathbf{x}_i\right) \qquad (2.19)$$

5 Choice of Kernel Dictionary and other Practical Issues

While the SV method automatically selects the height and center-point of each kernel distribution, the widths of our distributions must be selected a-priori when choosing our kernel dictionary. Here, we must trade-off between providing an adequate range of widths to properly describe the distribution and the computational time required to solve (2.18). The trade-off should be done on a problem-by-problem basis.

For all simulations in this paper we have used a gaussian-like kernel functions of the form:

$$K\left(\mathbf{x},\mathbf{y}\right)=\prod_{i=1}^{d_L}\frac{1}{1+e^{-\gamma\left(x^i-y^i\right)}}$$

$$\kappa\left(\mathbf{x},\mathbf{y}\right)=\prod_{i=1}^{d_L}\frac{\gamma}{2+e^{\gamma\left(x^i-y^i\right)}+e^{-\gamma\left(x^i-y^i\right)}} \qquad (2.20)$$

The density estimator was implemented in C++ Simulations were run in DOS under Windows 2000 on a 1GHz Pentium III Coppermine computer. Linear programming was done using the Higher-Order Primal-Dual Method, HOPDM, softv package with permission.

6 Results

We tested the SV density estimator using sam drawn from the distribution:

$$p(x)=\frac{1}{4\pi}e^{-\frac{1}{2}\begin{bmatrix}x & y\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}}+\frac{1}{\pi}e^{-\frac{1}{2}\begin{bmatrix}x & y\end{bmatrix}\begin{bmatrix}4 & 0\\ 0 & 4\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}}$$ (

The kernel dictionary consisted of 4 sigm kernels of type (2.20) $\gamma = 1.33,\ 2.66,\ 2.99$ and 6.65 ($w_i = i$), with bounds $\mathbf{a} = [-7.98 \quad -7.98]^T$, $\mathbf{b} = [5.32 \quad 5.32]^T$.

Figure 2 shows the estimate attained using samples, of which the regressor selected 12 support vectors. The more accurate estimate shc in figure 3 shows the result for 200 training samp In this case, only 8 support vectors were found 4% of the total number of training vectors.

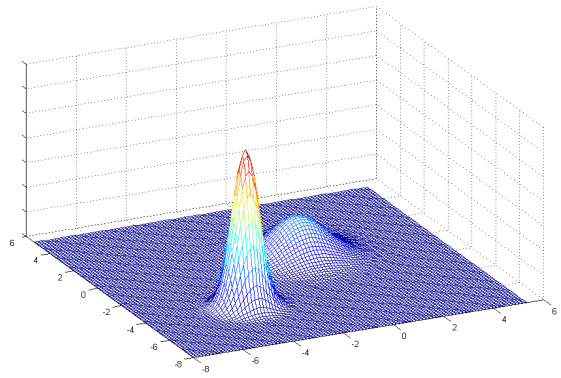

Figure 2: Test distribution function.

7 Conclusions

We consider the problem of density estimation in the context of Distributed Bayesian Data Fusion. The key issue in such a problem is the accurate transmission of the probability density from sensor nodes to the fusion node. We propose a method based on support vector machines for this approach and demonstrate its effectiveness in a two dimensional problem. We show that, in the example shown, the number of support vectors needed to adequately estimate the density function is extremely small at the cost of increased computational load. Incremental methods for SVMs provide a method of reducing this load and future work is focused in this area.

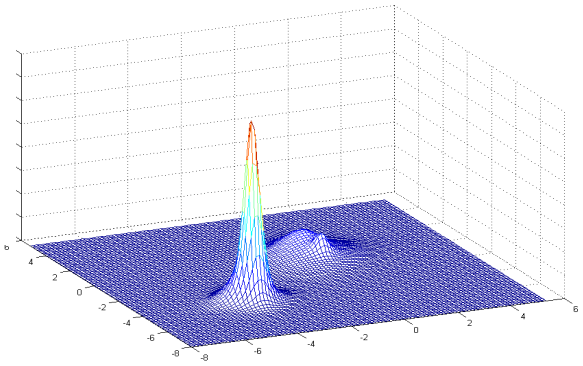

Figure 3: SV estimate learned from 100 particles

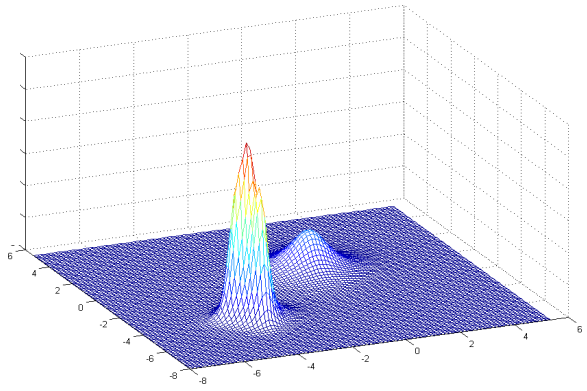

Figure 4: SV estimate learned from 200 particles

References

[1] N. Gordon, D. Salmond and A. Smith, "Novel Approach to nonlinear/non-Gaussian Bayesian state estimation", IEEE Proceedings-F, Vol. 140, pp. 107-113, April 1993.

[2] A. Doucet, N. Gordon, and V. Krishnamurthy, "Particle Filters for state Estimation of Jump Markov Linear Systems", IEEE Transactions on Signal Processing, Vol. 49, pp. 613-624, March 2001.

[3] Y. Bar-shalom and X. R. Li, "Multitarget-Multisensor Tracking: Principles and Techniques, ISSN 0895-99110, YBS Publishing, 1995.

[4] V. Vapnik and S. Mukherjee, "Support Vector Method for Multivariate Density Estimation", Advances in Neural Information Processing Systems, Vol 12, MIT Press 2000

[5] V. Vapnik, "Nature of Statistical Learning Theory", Second Edition, Springer, NY, 1999