LIDS Technical Report # 2601

Using Sample-based Representations Under Communications Constraints

ALEXANDER T. IHLER
JOHN W. FISHER III
ALAN S. WILLSKY
Massachusetts Institute of Technology

Created June, 2004; Revised December, 2004

Abstract

In many applications, particularly power-constrained sensor networks, it is important to conserve the amount of data exchanged while maximizing the utility of that data for some inference task. Broadly, this tradeoff has two major cost components—the representation's size (in distributed networks, the *communications* cost) and the error incurred by its use (the *inference* cost).

We analyze this tradeoff for a particular problem: communicating a particle-based representation (and more generally, a Gaussian mixture or kernel density estimate). We begin by characterizing the exact communication cost of these representations, noting that it is *less* than might be suggested by traditional communications theory due to the invariance of the representation to reordering. We describe the optimal, lossless encoder when the generating distribution is known, and pose a sub-optimal encoder which still benefits from reordering invariance.

However, lossless encoding may not be sufficient. We describe one reasonable measure of error for distribution-based messages and its consequences for inference in an acyclic network, and propose a novel density approximation method based on KD-tree multiscale representations which enables the communications cost and a bound on error to be balanced efficiently. We show several empirical examples demonstrating the method's utility in collaborative, distributed signal processing under bandwidth or power constraints.

1 Introduction

Wireless sensor networks are increasingly utilized in a large number of applications, including tracking, surveillance, and environmental monitoring applications [1, 2]. They require significantly less physical infrastructure than their wired counterparts and can be deployed at substantially lower cost. Efficient utilization of finite energy resources is one of the most important requirements for wireless sensor networks. The high energy cost of communications, relative to sensing or computation, makes limiting inter-sensor communications while maintaining effectiveness a desirable goal.

Unfortunately, reducing communications comes at the expense of the primary goal of a sensor network: to accumulate and fuse information from distributed sensors. Power may be conserved through intelligent routing of messages or data selection [3, 4]; however, it is also possible to trade off the *fidelity* of the information with its communications cost. This is particularly true for redundant representations (e.g., fine-grain discretizations or large collections of samples). The latter compromise falls into the category of lossy source coding; approximating the data representation

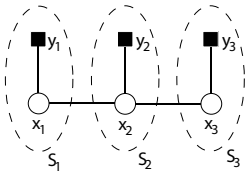


Figure 1: A simple (yet sufficiently general) graphical model description for the transmission problem. A sensor S_1 wishes to send its information y_1 to S_2, who will use it to perform inference on x_2 (or pass it on to S_3).

to fit within some communications budget [5]. Lossy data compression is generally examined from the perspective of minimizing reconstruction error of the data; in contrast, analysis with respect to *inferential* utility is relatively unexplored [6, 7].

In this paper, we explore the tradeoff between the cost of communication and the fidelity of the data representation when the transmitted message is a *distribution* (or likelihood function). Specifically, we evaluate the impact of message approximation on subsequent inference rather than distortion measures on the original data. This has many similarities to standard density approximation tasks such as vector quantization for source coding [5] or "reduced-set" density estimation [8]; however, when communication resources are dear, a more careful examination of both error measures and communication cost is warranted.

We outline the problem statement in Section 2. Section 3 examines the cost of optimal, lossless encoding of sample based representations, discussing some necessary features of any such encoder. We then turn to lossy encoding—Section 4 describes one appropriate measure of error while Section 5 presents a multi-resolution description for which one may efficiently balance communications cost with the potential for error in subsequent inference. Section 6 describes a few additional considerations and directions, and we conclude with examples and applications in Section 7.

2 Problem overview

We begin with a canonical inference problem in which to frame our analysis. Sensors $S_1 \dots S_3$ each observe a local random variable $y_1 \dots y_3$ (respectively); each sensor S_i then uses the observations as information about a local latent variable x_i. Our goal is for S_1 to encode and transmit to S_2 information about y_1 so as to compute the posterior marginal distribution $p(x_2|y_1, y_2, y_3)$. An additional goal is to propagate this information from S_2 to S_3 so as to compute $p(x_3|y_1, y_2, y_3)$ as well.

A graphical model [9] depicting this scenario is shown in Fig. 1. Graphical models are a popular means of encoding statistical (in)dependency relationships; specifically, two random variables are conditionally independent if they are separated in the graph by the conditioning variables. For example, denoting $\mathbf{y} = [y_1, y_2, y_3]$, the graphical model in Fig. 1 factors as

$$p(x_1, x_2, x_3|\mathbf{y}) = p(x_1|y_1)p(x_2|y_2, x_1)p(x_3|y_3, x_2) \tag{1}$$

which respects the graph's separation properties, e.g., that x_1 and x_3 are independent given x_2. Graphical models are often used to construct *global* inference algorithms via *local* information or message passing. Local sensing, distributed in-network processing, and limited bandwidth combined with finite energy resources necessitate a compromise between communication costs and information content.

We focus on calculating the posterior marginal distributions $p(x_i|\mathbf{y})$ on tree-structured graphs, a task which may be accomplished using the *belief propagation* (or *sum-product*) algorithm [10]. In regard to Figure 1, and without loss of generality, we analyze the computation of posterior distribution of x_2:

$$p(x_2|\mathbf{y}) \propto p(x_2)p(y_1|x_2)p(y_2|x_2)p(y_3|x_2). \tag{2}$$

while minimizing communications from S_1, momentarily ignoring the inference tasks of the other sensors. This situation arises, for example, if S_2 were responsible for fusing information or communicating it to an outside user. It may be considered a special case of more general, symmetric inference problems.

We will assume that the message m_{12} transmitted from S_1 to S_2 is a *function*, and specifically may be either of the local posteriors $p(x_2|y_1)$, $p(x_1|y_1)$ or likelihoods $p(y_1|x_2)$, $p(y_1|x_1)$. We also assume that both S_1 and S_2 share the prior model $p(x_1, x_2)$, in which case all four functions may be considered essentially equivalent, since given any of these functions (and similar information from S_3), it is straightforward for S_2 to calculate $p(x_2|\mathbf{y})$ using Bayes' rule. For concreteness, we will take $m_{12} = p(y_1|x_2)$; typically each message (function) is normalized for numerical stability. A primary assumption of sending a *distribution* message rather than the raw data is that the size of S_1's observation y_1 is larger than the representation size of the likelihood $p(y_1|x_2)$ (parameterized by x_2). This may be the case for a number of reasons—the y_i may be high-dimensional (e.g., high-resolution imagery), or may be a large set of accumulated data (e.g., an entire observation history).

This formulation easily extends to larger tree-structured graphs, in which case y_1 represents all information separated from sensor S_2 by S_1. Tree structured graphical models have already found application in sensor networks [11]. While certain problems on sensor networks may be described by loopy (non-tree structured) graphical models [12], inference in these situations (and thus the communications/error tradeoff) is considerably more complex and remains a subject of ongoing research.

2.1 Message Representation

While there are many possible representations for the inter-sensor messages we concentrate on sample-based representations, assuming that each message is described by a kernel density estimate [13]

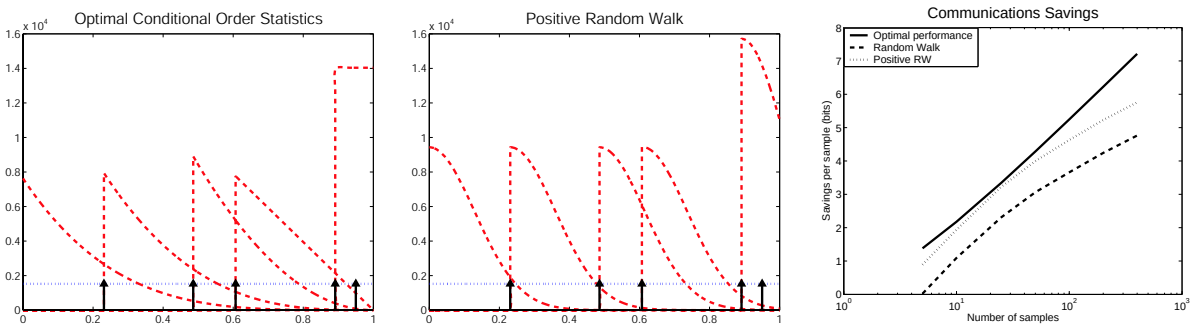
$$m(x) = \sum_i w_i \; K_{\sigma_i}(x - \mu_i) \qquad\qquad K_{\sigma_i}(x - \mu_i) = \mathcal{N}(x; \mu_i, \sigma_i^2) \tag{3}$$

where K is a Gaussian kernel with diagonal covariance matrix specified by the vector σ_i^2, making $m(x)$ a Gaussian sum. Though not necessary, assuming a diagonal covariance simplifies much of the subsequent discussion.

Gaussian sum-based messages are common in a number of applications. For example, they represent a generalization of the distribution estimates in particle filtering algorithms (in which $\sigma_i \equiv 0$) [14, 15], and more recently appear in stochastic approximations to belief propagation on general graphical models [16].

3 Lossless Transmission of Sample-Based Representations

We begin by considering the task of *lossless* encoding. We shall see that the communications cost of a sample-based representation is significantly less than might be indicated by naive application



(a) (b) (c)

Figure 2: Deterministic ordering reduces the entropy of the sample set. Optimal encoding can be accomplished via the order statistics (a), if the distribution is known; alternatively, a random walk-based conditional distribution (b) can be used instead, though the per-symbol savings (c) will be reduced.

of classical source coding results. In the following we assume that S_2 provides no feedback and that the message $m(x_2)$ takes the form (3) in which the μ_i are drawn *i.i.d.* from some distribution $p(\mu)$, and $w_i = 1/N$ for all i. The bandwidth $\sigma_i \equiv \sigma$ is assumed to be known at the receiver, either transmitted once and the cost neglected or computed deterministically from the data. The consequence of these assumptions is that we may analyze the asymptotic costs involved with the transmission of the data set $\{\mu_i\}$. Specifically, we show that the invariance of the representation to reordering of the $\{\mu_i\}$ leads to significant communication savings.

3.1 Optimal Communications

A standard information-theoretic result is that the minimum cost of transmitting large volumes of continuous-valued data can be expressed in terms of the differential entropy. That is, a *sequence* of N i.i.d. random variables $\mu_i \sim p(\mu)$ with entropy $H(p)$ can be sent up to some resolution β (in bits) with expected cost

$$(\beta + H(p)) \cdot N \tag{4}$$

Encoders which achieve this for known p include the classic Huffman and arithmetic codes [5]. For small values of N and β, quantization effects and other factors may influence the actual performance of a source coding scheme. However, for simplicity we focus here on the ideal case.

The problem of communicating a kernel density estimate is considerably simpler requiring only the transmission of the *set* of samples $\{\mu_1, \ldots, \mu_N\}$ independent of *ordering*[1]. The maximal improvement is bounded by the reduction in entropy of the reordered samples. We assume sufficient resolution such that no two samples fall within the same bin (i.e., the μ_i each differ by more than $2^{-\beta}$). We denote the complete sample sequence $\boldsymbol{\mu}_N = [\mu_1, \ldots, \mu_N]$ and its distribution by $p(\boldsymbol{\mu}_N)$. A determinstically re-ordered set (e.g., sorted by value) is denoted $\boldsymbol{\mu}_N^s = [\mu_{(1)}, \ldots, \mu_{(N)}]$ with distribution[2] $\rho(\boldsymbol{\mu}_N)$.

[1]Interestingly, reordering has also been applied to sequence coding applications; for example, a reversible reordering procedure is used in the Burrows-Wheeler transform [17] for (discrete-alphabet) source coding to capture redundancy in non-i.i.d. sequences.

[2]Throughout this paper, we will adhere to the convention of using the symbol ρ for distributions of deterministically ordered quantities such as the $\mu_{(i)}$.

It can be shown that for any deterministic ordering [18],

$$\rho(\boldsymbol{\mu}_N) = \begin{cases} N!\; p(\boldsymbol{\mu}_N) & \boldsymbol{\mu}_N \text{ "in order": } \boldsymbol{\mu}_N = \boldsymbol{\mu}_N^s \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

This is because there are $N! = N \cdot (N-1) \cdots 1$ sets $\boldsymbol{\mu}_N$ (corresponding to $N!$ possible orderings) which map to the same $\boldsymbol{\mu}_N^s$. Consequently, it can be shown that the entropy of the sorted data $\boldsymbol{\mu}_N^s$ is given by

$$H(\rho(\boldsymbol{\mu}_N)) = H(p(\boldsymbol{\mu}_N)) - \log_2 N!$$

indicating savings up to $\log N!$ bits over the cost of sending the sequence naively.[3]

Equation (5) is a classic result from the analysis of *order statistics*, defined to be the ascending sorted values for a set of one-dimensional (1-D) random variables. Order statistics provide a natural and well-studied deterministic order in 1-D. While our notation for $\mu_{(i)}$ is consistent with order statistics, the previous analysis is independent of the method of ordering, and thus generalizes to arbitrary dimension.

Fig. 2(a) illustrates the savings by way of a simple example. We show five samples drawn from $p(\mu)$ uniformly distributed on the interval $[0, 1)$ (shown as arrows), the distribution $p(\mu)$ and the conditional distribution of the i^{th} order statistic $\rho(\mu_{(i)}|\mu_{(i-1)})$ for each i. Each $\mu_{(i)}$ has lower entropy than $p(\mu)$, a difference which translates into lower transmission costs.

3.2 Sub-optimal Encoding

Typically $p(\mu)$ is *unknown* and the distributions $\rho(\mu_{(i)})$ are *non-stationary*, i.e., they depend on i; the reduced cost of optimal transmission is due to $\rho(\mu_{(i)}|\mu_{(i-1)})$ changing predictably as a function of i and $\mu_{(i-1)}$. A (suboptimal) way to exploit this property for unknown p is to assume *conditional* stationarity, i.e., $\rho(\mu_{(i)}|\mu_{(i-1)}, \ldots)$ is constant for all i. While in general this assumption does not hold, even a simple conditionally stationary code yields some degree of predictive power. A common example is linear predictive coding (LPC) [5]. For now, we focus on 1-D distributions and ascending sample order, deferring higher-dimensional distributions to Section 5.

One example of a simple yet useful LPC is a random-walk code. For purposes of illustration, we compare two suboptimal, random walk based approximations to the optimal encoder for one-dimensional samples distributed uniformly on $[0, 1)$. The first method encodes each sample using the conditional distribution

$$\hat{\rho}(\mu_{(i)}|\mu_{(i-1)}) = \mathcal{N}(\mu_{(i)}; \mu_{(i-1)}, \sigma^2) \tag{6}$$

while the second encodes according to the one-sided random walk

$$\hat{\rho}(\mu_{(i)}|\mu_{(i-1)}) = \begin{cases} 2\mathcal{N}(\mu_{(i)}\ ;\ \mu_{(i-1)}, \sigma^2) & \mu_{(i)} \geq \mu_{(i-1)} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

where σ^2 is equal to the variance of $p(\mu)$ divided by N (i.e., $\sigma^2 = \frac{1}{12N}$) and $\mathcal{N}(\cdot)$ is a Gaussian distribution. Fig. 2(b) shows the conditional distributions induced by the one-sided random walk (7), as compared to the optimal distributions in Fig. 2(a). We may also measure the performance of each suboptimal code using the per-sample savings (in bits) achieved as a function of N; optimally, this is $\frac{1}{N} \log_2 N!$, while for any stationary code it is approximately zero. Fig. 2(c) shows curves comparing the per-sample savings of the optimal encoder, the zero-mean random walk (6), and one-sided random walk (7).

[3]Note that this is no longer accurate if we allow multiple, equal-value (at the resolution β) samples, since this would result in fewer than $N!$ possible distinct orderings.

4 Message Approximation

The analysis of the previous section provides a baseline of optimal lossless communications for sample-based message transmission. *Lossy* encoding, however, requires a measure of error between two possible messages. While source coding approaches typically consider a distortion measure on the data itself, we are interested in measures which relate to errors in the estimated posterior distribution. Ideally, we desire *local* rules which, when utilized at all sensors, lead to *global* bounds or estimates on the error at each sensor. One measure with this property is the maximum log-error

$$\Delta(m, \hat{m}) = \max_x \left| \log m(x) \,/\, \hat{m}(x) \right|, \tag{8}$$

for which a more detailed discussion is given in [19]. By controlling a similar measure d, defined as

$$d(m, \hat{m}) = \max_x \min_\alpha \left| \alpha + \log m(x) \,/\, \hat{m}(x) \right|$$

the measure Δ is well-behaved with respect to inference operations in a graphical model. The two are related for normalized messages $m, \hat{m}$ by the inequalities

$$d(m, \hat{m}) \leq \Delta(m, \hat{m}) \leq 2d(m, \hat{m})$$

Additionally, d has three important properties relevant to bounding errors:

For messages $m, \hat{m}, \tilde{m}$,	$d(m, \tilde{m}) \leq d(m, \hat{m}) + d(\hat{m}, \tilde{m})$
For $m_2 \propto m_{12} m_{32}$, $\hat{m}_2 \propto \hat{m}_{12} \hat{m}_{32}$,	$d(m_2, \hat{m}_2) \leq d(m_{12}, \hat{m}_{12}) + d(m_{32}, \hat{m}_{32})$
For $f * m = \int f(x, z) m(z) dz$, $f > 0$	$d(f*m, f*\hat{m}) \leq d(m, \hat{m})$

Together, these properties enable one to bound the influence of a particular message approximation on any other message or posterior marginal in the graphical model, and more generally bound the error from a set of message approximations by the sum of their individual errors. The measure $\Delta(\cdot)$ is a strict measure; for example, it requires nearly *identical* tail behavior for continuous distributions. There are several means to ensure this for Gaussian mixture distributions. For example, if the mixture components which determine the tail behavior can be identified they may be preserved to high precision. Another possibility is that a *single* mixture component determines the tail behavior of the overall distribution, as when one very broad (possibly low-weight) component dominates the rest of the distribution in very low-likelihood regions. Such components are often added to model outlier processes [20, 12], and may be either deterministically added at the receiver or transmitted with high precision.[4]

Evaluating the error $\Delta(\cdot)$ between two Gaussian mixtures is also non-trivial. It may be performed either by discretization and direct evaluation in relatively low (1-2) dimensions, or via gradient search. Although gradient search can be susceptible to local maxima, the form of (8) as the ratio of Gaussian mixtures leads one to expect that the global maximum may be found with relative ease by local optimization from each of the mixture centers, similar to that outlined by [21]. Thus, estimating $\Delta(\cdot)$ requires $\mathcal{O}(N)$ operations. For comparing a *single* Gaussian to a kernel density estimate (a task important for the next section), one may also *estimate* $\Delta(\cdot)$ by simply evaluating the magnitude of the log-ratio at each kernel center μ_i and selecting the largest.

[4] A similar solution involves the addition of a small but non-zero constant to the distribution estimate (or imposition of a threshold away from zero), essentially modeling the inclusion of a small uniform (rather than Gaussian) outlier component.

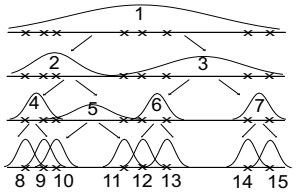


Figure 3: A one-dimensional KD-tree representing a mixture of 8 Gaussian kernels, and caching means and covariances at each level (resulting in a hierarchy of Gaussian approximations). The nodes have been labeled by the numbers $\mathtt{1}, \ldots \mathtt{15}$ for the discussion in the text.

Figure 4: The Gaussian mixture approximation derived from the density set $S = \{\mathtt{4}, \mathtt{10}, \mathtt{11}, \mathtt{6}, \mathtt{7}\}$ on the KD-tree in Fig. 3

5 A Multi-scale Description

Given a measure of message error, we discuss a method to trade off communications and fidelity. Multi-scale descriptions have been proven to be useful in many data compression applications [22, 23]; they first capture large-scale phenomena then encode a series of "refinements" to describe finer details. Generally, the multi-scale description is hand-selected (e.g., wavelet decompositions, Fourier coefficients, etc.), and the refinement information forms a tree-like structure that can then be optimized to trade off representation size with reconstruction quality.

We consider a particular multi-scale description of the kernel density estimate based on *KD-trees* ("k-dimensional trees"), a common data structure used for storing and summarizing sets of real-valued samples. KD-trees cache various statistics of subsets of the samples, in our case estimates of their mean and covariance, which we use to define a multi-resolution Gaussian mixture model. This is, of course, only *one* possible encoding choice, but we shall see that it results in an elegant approximation algorithm. We first describe a class of multi-scale mixture models defined by a KD-tree, then show how it may be applied both to encode and to control error in the message.

5.1 KD-tree Gaussian Mixtures

KD-trees [24] are a well-known data structure for rapidly performing locality-based computations on large sets of continuous-valued points. They are binary tree structures whose leaf elements each store (in our application) one μ_i and associated covariance σ_i^2. Internal (non-leaf) nodes store any of a number of sufficient statistics of the data represented by their children, allowing many computations to be performed without accessing the individual leaf nodes and leading to considerable speedups. Here, we use the KD-tree structure to define a hierarchy of Gaussian approximations to subsets of the message $m(x)$. The sufficient statistics stored at each node are the mean and covariance of the Gaussian sum defined by the node's children, along with a weight w_i representing the total weight contained in its subtree (i.e., for an equal-weight kernel density estimate, the number of leaf nodes below divided by the total number of leaf nodes). This is easily computed recursively for each node i, since $w_i = w_{i_l} + w_{i_r}$ and

$$\mu_i = \frac{w_{i_l}}{w_i}\mu_{i_l} + \frac{w_{i_r}}{w_i}\mu_{i_r} \qquad \sigma_i^2 = \frac{w_{i_l}}{w_i}(\sigma_{i_l}^2 + \mu_{i_l}^2) + \frac{w_{i_r}}{w_i}(\sigma_{i_r}^2 + \mu_{i_R}^2) - \mu_i^2 \tag{9}$$

where i_l, i_r are the left- and right-hand children of node i and the sum is computed element-wise; recall that the σ^2 are assumed diagonal.

There are many methods of constructing KD-trees [25, 26]; it is not our purpose to investigate the relative merits of these methods, nor do we require that any particular method be used. In general, the simplest construction algorithms work by a top-down set-splitting procedure, for example dividing the data into (nearly[5]) equal sets along some cardinal axis chosen either deterministically or according to the covariance of the data contained in its subtree.

A simple one-dimensional example of a KD-tree which caches means and covariances is shown in Fig. 3. In this figure, the nodes have been numbered $\mathtt{1} \ldots \mathtt{15}$, where node $\mathtt{1}$ is the root and $\mathtt{8} \ldots \mathtt{15}$ are the leaves. The *descendants* of a node i are the nodes located below it in the tree (e.g., the descendants of node $\mathtt{2}$ are $\{\mathtt{4}, \mathtt{5}, \mathtt{8}, \mathtt{9}, \mathtt{10}, \mathtt{11}\}$), while its *ancestors* are the nodes located above it (so that the ancestors of node $\mathtt{4}$ are $\{\mathtt{1}, \mathtt{2}\}$).

KD-trees summarize large sets of data via cached statistics enabling many computations without accessing the raw data. For example, KD-trees have been applied to improve the speed of EM for learning mixture models [27]. In contrast to their typical application, we use the tree structure to find a simpler description which has overall bounded approximation error. The KD-tree defines a class of Gaussian mixture models, parameterized as follows: define an admissible density set S to be any set of nodes in the KD-tree such that, for every node $i \in S$, S contains neither descendants nor ancestors of i, and for every *leaf* node j, either j or some ancestor of j is contained in S. The Gaussian sum defined by any such S yields an approximation to the original (finest-scale) Gaussian sum,

$$\hat{m}_S(x) = \sum_{i \in S} w_i \, \mathcal{N}(x \; ; \; \mu_i, \sigma_i^2) \quad \approx \quad m(x) = \sum_{\text{leaves } i} w_i \, \mathcal{N}(x \; ; \; \mu_i, \sigma_i^2) \tag{10}$$

of varying degrees of coarseness depending on the selection of nodes in S. For example, the Gaussian sum given by $S = \{\mathtt{4}, \mathtt{10}, \mathtt{11}, \mathtt{6}, \mathtt{7}\}$ is shown in Fig. 4. The admissibility conditions essentially require that each leaf node be represented by one and only one Gaussian mixture element. Due to the tree-structure, it is possible to efficiently optimize both communications cost and approximation error over the class of admissible density sets; computing these two cost functions on the KD-tree form the subjects of the next two sections.

5.2 KD-tree Communications Cost

We now consider the design of an encoder for the kernel density estimate defined by a parameter set $\{\mu_i, \sigma_i^2\}$ which respects the clustering induced by the KD-tree, and allows us to calculate the communications benefit of sending only certain coarse-scale approximations (as opposed to the entire representation). We note that, since the KD-tree representation of the $\{\mu_i\}$ is constructed deterministically from the data, there is no more randomness in the KD-tree than in the data set as a whole. In other words, to communicate the KD-tree description, we may simply communicate all μ_i and rebuild (bottom-up); thus, an optimal representation of the entire KD-tree should cost no more than sending the samples themselves.

We prefer, however, to send the approximations in a coarse-to-fine manner; sending the upper levels of the KD-tree first enables us to simply cease transmitting refinements for a given branch whenever it is deemed "good enough", thus reducing the communications required. To this end, we design an encoder (based on the methods discussed in Section 3.2) which first transmits the mean and covariance of the root node, then encodes each node's refinement information in a manner consistent with the current, coarse-scale density estimate of that subtree.

[5]Sets with an odd number of points are typically split according to some convention, for example placing the additional point in the left-hand branch.

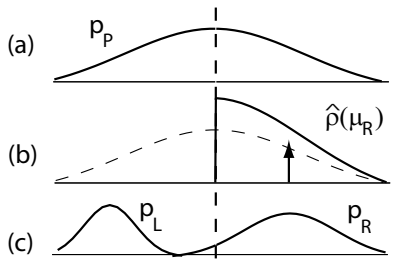


Figure 5: Transmitting a KD-tree in top-down fashion. (a) Given the mean μ_p and covariance σ_p^2 of a coarse-scale estimate, (b) we encode the right-hand mixture component according to (12)-(13); the encoding distribution $\hat{\rho}(\mu_r)$ is shown as solid, while the transmitted value of μ_r is indicated by the arrow. (c) Having decoded the right-hand component, the receiver may simply solve for the left-hand component using (9).

Suppose that we have transmitted a coarse (single Gaussian) approximation $q_p(x) = \mathcal{N}(x; \mu_p, \sigma_p^2)$ of some subtree rooted at node p. The receiver must be able to recover the means μ_l and μ_r and covariances σ_l^2, σ_r^2 of node p's left and right children (denoted l and r respectively) to refine the subtree by one level. Equation (9) yields two equations with four unknowns; thus it is sufficient[6] to transmit only μ_r, σ_r^2. Furthermore, if the leaf nodes all have equal, known bandwidths, the cost of refinement at the finest scales is smaller still; for example, if *both* children are leaves, refinement is essentially free (two equations with two unknowns). Eventually, we consider transmitting only a *subset* of the possible branchings; for our analysis we neglect the small overhead required to indicate the choice of which refinements are sent (which requires at most one bit per node retained).

The manner of encoding the right- (or left-) hand mean and covariance is a design choice; clearly, the summarization statistics μ_p, σ_p^2 can be used to construct an encoding distribution known at both sender and receiver. For example, a simple choice reminiscent of Section 3.2's random-walk encoder is given by

$$\hat{\rho}(\mu_r | \mu_p, \sigma_p^2) = \begin{cases} 2\mathcal{N}(\mu_r; \mu_p, \sigma_p^2) & \mu_r \geq \mu_p \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$\hat{\rho}(\sigma_r^2 | \sigma_p^2, \mu_p, \mu_r) = \mathcal{N}(\sigma_r^2; \hat{\sigma}_r^2, \hat{\sigma}_r^2/2) \qquad \hat{\sigma}_r^2 = \sigma_p^2 + \mu_p^2 - \frac{w_r}{w_p}\mu_r^2 - \frac{w_l}{w_p}\mu_l^2 \tag{12}$$

For non-diagonal covariances, this involves a more complex encoding distribution. If all leaf nodes have the same bandwidth σ, a slight improvement to the mean value's encoder can be obtained by using the variance of the *samples* rather than the variance of their represented distribution, i.e., define $\bar{\sigma}_p^2 = \sigma_p^2 - \sigma^2$ and use

$$\hat{\rho}(\mu_r | \mu_p, \sigma_p^2) \begin{cases} 2\mathcal{N}(\mu_r; \mu_p, \bar{\sigma}_p^2) & \mu_r \geq \mu_p \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Using any such predictive encoder, we may pre-compute the communications cost of all $N-1$ potential refinement actions while constructing the KD-tree, allowing one to easily check (for example) whether a particular set S is within a communications budget. Finally, note that this procedure generalizes easily to densities in arbitrary dimension; the KD-tree provides a natural

[6]With a slight caveat: due to the averaging process, given μ_p, μ_r to precision β, μ_l is computed to precision $\beta - 1$; thus we may require an additional bit of information for μ_l.

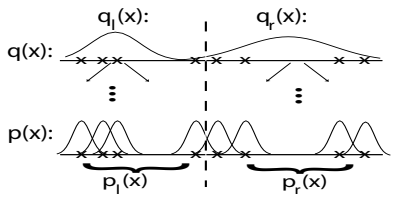


Figure 6: Subdividing the KD-tree Gaussian mixture. An error measure between the densities $p(x) = p_l(x) + p_r(x)$ and $q(x) = q_l(x) + q_r(x)$ can be bounded by error measures on the individual pairs $p_l(x), q_l(x)$ and $p_r(x), q_r(x)$.

ordering (top-down; approximately equal size, binary splits), just as the sorted order did in 1-D. Positivity requirements such as appear in (11) and (13), if present, are only included for the dimension along which the data has been split at each level, since this is the only dimension in which the KD-tree provides ordering information.

5.3 KD-tree Approximation Error

Many algorithms attempt to choose "important" samples or otherwise reduce the number of kernel centers in the density estimate. Some attempt to minimize KL-divergence [28] or L_2 (integrated squared) error [8]. We add the maximum log-error described in Section 4 to this list, describing a simple and computationally efficient bounded-error approximation scheme, based on the multiscale approximations stored in a KD-tree data structure.[7]

The key to performing the optimization efficiently (without enumerating all possible density estimates) is to decompose the error of any particular admissible density set S, with KD-tree Gaussian mixture q_S, into an error measure on *only* individual elements $s \in S$. This allows us to determine if a sub-tree of the hierarchy has already been approximated "sufficiently well", or whether we may need to refine it (eventually improving the quality of our approximate density estimate in that sub-region). In particular, we split the true distribution into two parts, $p(x) = w_l p_l(x) + w_r p_r(x)$, and similarly for any approximation of those parts $q(x) = w_l q_l(x) + w_r q_r(x)$. An illustration of this decomposition is shown in Fig. 6.

The relationship of error between p and q to the error between p_l, q_l and p_r, q_r depends on the form of error measure between p and q. For the maximum log-error $\Delta(p, q)$ we may use the inequality

$$\Delta(w_l p_l + w_r p_r \,,\, w_l q_l + w_r q_r) \leq \max\left[\Delta(p_l \,,\, q_l) \;,\; \Delta(p_r \,,\, q_r)\right]. \tag{14}$$

In consequence, we may separately consider the approximation of each sub-tree $s \in S$ by the single Gaussian approximation stored at s. Evaluating the error between the top and leaf nodes of each subtree gives a bound on the error in the full density estimate (i.e., that the total error is bounded by the largest error over $s \in S$).

This decomposition leads to a fast, bounded-error approximation algorithm—to approximate p with at most some error ϵ using only a few kernels, we construct a KD-tree representation of p, retaining Gaussian approximations of the sub-trees at each level. Then, beginning with the topmost level, we evaluate the error associated with this approximation. If it does not meet the required quality ϵ, we select and refine the subtree s with largest error bound contribution $\Delta(p_s, q_s)$ to improve the approximation. As the bound on Δ is dominated by the *maximum* error in any subtree, this procedure exactly optimizes the bound. It is also very fast—even for $\epsilon = 0$ (the worst

[7] Interestingly, this algorithm can also be applied as a greedy optimization of many other types of error measures, for example the Kullback-Leibler divergence, L_1, and L_2 measures, though we omit details due to space constraints.

case) it requires at most $\mathcal{O}(N \log_2 N)$ time, since comparing a node p to its N_p children may be performed in $\mathcal{O}(N_p)$ evaluations, giving $\mathcal{O}(N + 2 \cdot \frac{N}{2} + 4 \cdot \frac{N}{4} + \ldots) = \mathcal{O}(N \log_2 N)$ total operations. In practice, for fixed $\epsilon > 0$ the algorithm requires about $\mathcal{O}(N)$ time, consistent with the intuition of stopping at an approximately constant depth determined by the complexity of the true distribution.

Alternatively, instead of minimizing communications subject to some error tolerance ϵ, we may use the same method to select a *tradeoff* between the two quantities by iteratively refining the subtree with largest error bound and keeping track of the set S with the best relative communications and error values (note that splitting does not always immediately improve the error!). It is also easy to modify this procedure to optimize the error bound subject to some communications constraint, whether specified in bits or by the number of retained Gaussian components.

Finally, as mentioned in Section 4, finite error Δ requires exact agreement in the distribution's tails. Thus, whatever method is applied to make (8) finite for a given Gaussian mixture (addition of a single broad component, flat threshold, or preservation of tail-dominant components) should also be applied to each subtree's approximation. This is easily done by adding the same common modes to each subset approximation, i.e., if $p_p = w_l p_l + w_r p_r$ and we add an outlier component p_0, we assign p_0 to each side in proportion to their weight: $p_p + p_0 = w_l(p_l + p_0) + w_r(p_r + p_0)$. This ensures that all subtrees match their finest scale subsets well enough to have non-trivial (finite) error bounds.

6 Additional Considerations

We have restricted the discussion as a means of illustrating the primary issues. Some additional considerations are worth noting, although a full analysis is beyond the scope of this discussion. Some of these are straightforward to incorporate into the analysis, while others pose significant challenges.

The first such aspect is the determination of the quantization level β. For lossless coding, β is determined by the resolution at which each sensor manipulates and stores data. For lossy coding, however, it is a parameter choice. Since large bandwidth components generally require less fidelity than smaller bandwidth components, methods which allow variable resolution and/or range (e.g., at different scales in the representation) may yield further savings. However, for such methods one must also consider any additional overhead in communicating β.

Secondly, some applications (particularly those described by loopy graphical models) may involve iterative message passing between sensors: sensors transmit local information then subsequently retransmit new information (received from other sensors) to refine the message. While we have focused on exploiting redundancy in the set of samples underlying a sample-based representation, exploiting redundancy in the sequence of messages for iterative (multi-transmission) problems may also be used to reduce communications. *Censoring* (opting not to send) messages which are "sufficiently close" to the previous version is a simple example of one such scheme; however, general methods for exploiting this structure remain a challenging research direction.

7 Experiments

This section describes a few example applications of KD-tree based density approximation and the tradeoff between error and communications. First, we show the process of approximation on a single message, then examine the performance in two example multi-sensor systems: a distributed particle filtering application, and estimation of a spatially dependent non-Gaussian random process.

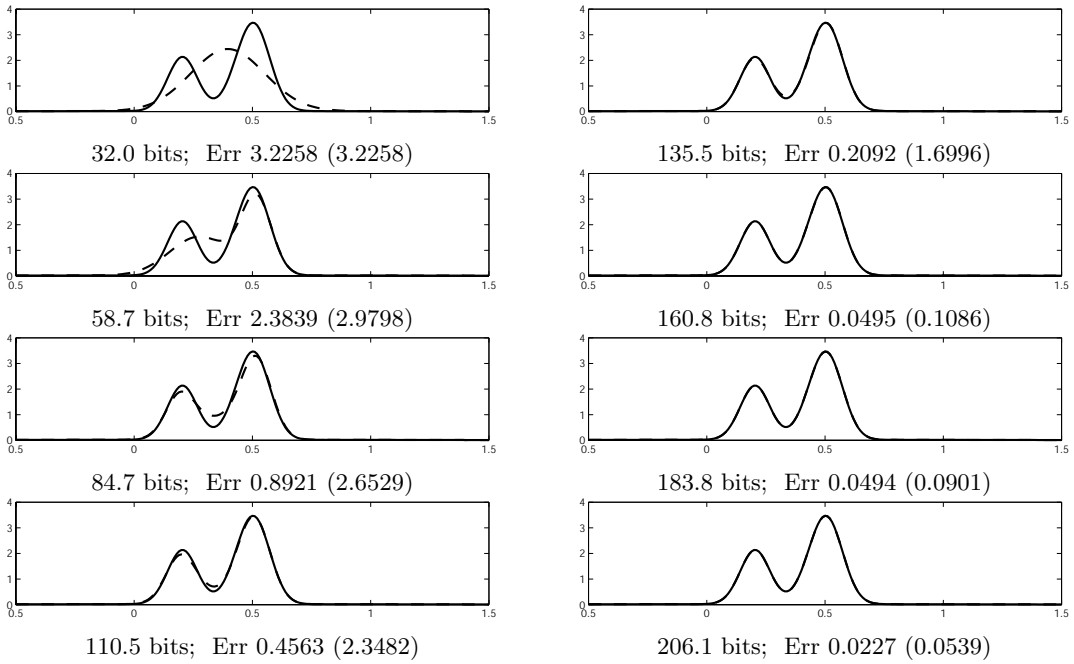


Figure 7: Sequence of KD-tree based approximations (dashed) to a 100-kernel density estimate (solid) of decreasing error and increasing communications cost (with $\beta = 16$ bits). Listed are the transmit cost in bits, and the actual error and tree-decomposed bound on Δ.

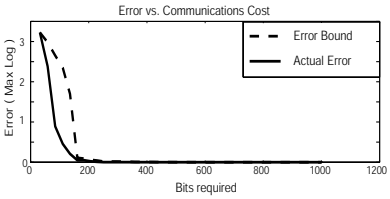
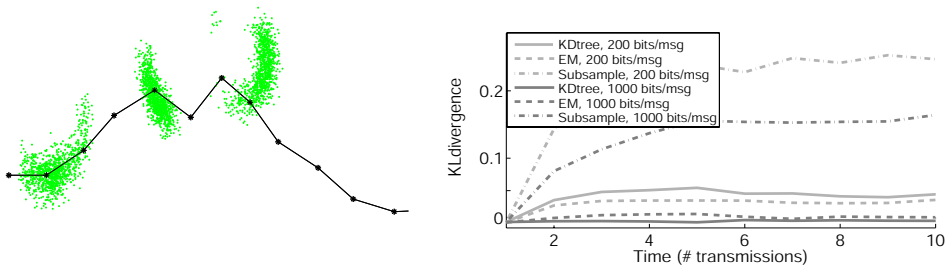


Figure 8: Comparing transmitted density error (both the tree-structured bound and actual error) versus total communications cost (in bits)—very few bits are required to transmit most of the density's information.

7.1 Single Message Approximation

We begin with a fixed kernel density estimate, showing the sequence of approximations as communication constraints are relaxed. The original kernel density estimate is made up of 100 samples. We transmit all parameters up to $\beta = 16$ bits of resolution (i.e., naive, lossless encoding requires 1600 bits). Fig. 7 compares the first eight Gaussian sums of the approximation sequence to the original density using our KD-tree splitting algorithm. Communications cost and max-log error are listed for each approximation. We also show the sequence of improvements in the error bound (and the actual resulting error) as a function of the number of bits required for transmission. Fig. 8 shows the rate at which the resulting error (solid) and bound (dashed) decline as communications increase. Beyond a certain communications level, we gain little for additional expenditures of energy.

Finally, we might also ask if changing β can improve performance. There are a few pitfalls in naively changing β; for example, if β is chosen too small we require special precautions when representing the bandwidth so as not to round any bandwidth to zero. Also, the choice of β affects the optimization over mixture components. For example, if β is decreased to only 8 bits, we improve



(a) (b)

Figure 9: Particle filtering example. (a) One sample path $\{x_t\}$, along with the samples used to estimate the posterior distribution at times $t = 2, 5, 8$. (b) Average increased KL-divergence at each time step for approximate message-passing (over ideal baseline). Careful approximations (EM and KD-tree) perform much better than subsampling; the KD-tree approximations (solid) outperforms EM at moderate bit-rates.

performance at moderate error levels—sending 6 Gaussians costs about 62 bits, with error bound of 2.23 and actual error of .330, compared to 2 Gaussians costing 59 bits, with error bounds 2.98 and actual error 2.38 at $\beta = 16$ bits. However, at some point the quantization error dominates—e.g., in this example with $\beta = 8$, sending more than 6 Gaussians never improved the inference error.

7.2 Distributed Particle Filtering

Particle filtering is often used for single- or multi-target tracking involving highly non-Gaussian observation likelihoods and potentially non-linear dynamics. In the context of wireless sensor networks, it is often appealing to perform the computations involved *locally* at one of the sensors (called the *leader* node, and typically chosen to be nearby to the target itself), removing the need to export data from the network at each time frame and reducing the distance over which observations and information must be communicated [4]. However, this creates a need to transmit the state descriptions (particle sets) from one sensor to the next as the target migrates from one region to another. Message approximation techniques can thus be applied to conserve bandwidth, saving power and extending sensor lifetime.

We examine the tradeoff between communications and error in this situation by considering a simple two-dimensional particle filtering application. We simulate an object moving in two dimensions, with dynamics

$$x_t = x_{t-1} + v_0[\cos\theta_t; \sin\theta_t] + w_t$$

where w_t is Gaussian and θ_t uniform:

$$w_t \sim N(0, \sigma_w^2 I) \qquad\qquad \theta_t \sim U\left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$$

At each time step t a single sensor (the leader) updates the estimated distribution using a range measurement from its location s_t:

$$y_t = \|x_t - s_t\| + d_t \qquad\qquad d_t \sim N(0, \sigma_d^2)$$

The leader node is changed after each update (i.e., at each time step), and the updated distribution estimate communicated to a new sensor. The distribution at each time step is typically unimodal but non-Gaussian; see Fig. 9(a).

Since at each time step t, the leader node must communicate its (particle-based) distribution estimate $\hat{p}(x_t)$ to another sensor, we may compare methods of compressing this message. First, note that it is typically cheaper to send $\hat{p}(x_t|y_t,\ldots)$ than $\hat{p}(x_{t+1}|y_t,\ldots)$ since there is less uncertainty (entropy) in the former. We compare three simple message-compression schemes, each parameterized by the total number of bits B required per message: subsampling, Expectation-Maximization (EM), and the KD-tree optimization described in Section 5.

For subsampling, we simply re-draw sufficiently few samples that the expected cost of sending the sample set (as defined in Section 3) is less than B bits. EM is commonly used to fit Gaussian mixture models; however, since efficient encoding of such a mixture model is an open question, we simply choose the number of components to require fewer than B bits in a naive (direct fixed-point) representation. Finally, the KD-tree method applies the simple hierarchical encoding described previously, refining while the transmission cost is less than B bits. All values are maintained at resolution $\beta = 16$ bits, and we compare the resulting distributions (at each time step) to those obtained by exact transmission of all samples.

We compute the average increase in KL-divergence due to message approximation by finding the divergence between the estimated posteriors and those obtained using exact messages, minus the KL-divergence found by using exact transmission but an alternate initial particle set. The results, for each approximation method over 500 Monte Carlo trials and $B \in \{200, 1000\}$ bits, is shown in Fig. 9(b). For a given bit budget, smart approximation of the distribution (using either EM or KD-tree based methods) performs considerably better than simple subsampling. At extremely low bit rates (200 bits $\approx$ 2-3 components), EM may perform better, but by moderate bit rates the KD-tree method is nearly as good as exact transmission, and requires less computation (about one-tenth the computation time for 1000 bits).

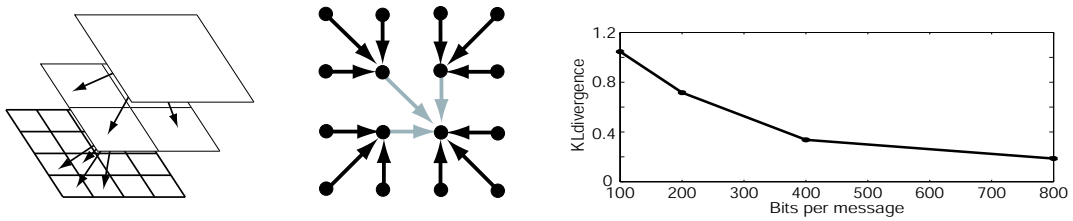
The differences between EM and the KD-tree method are likely due to a few factors. First, the two methods are optimizing different measures of error (KL-divergence vs. maximum log-error). This appears to be an advantage for average-case performance; in fact, on this problem (in experiments omitted for space) a greedy KD-tree optimization to minimize KL-divergence also outperformed EM at $B = 200$ bits. EM also entails a less constrained optimization, which is beneficial for few mixture components but may suffer from local maxima with many components. Finally, the KD-tree's extra constraints are used to define an efficient encoding, so that for a given bit budget it typically has more mixture components than a naive encoding of a mixture model found via EM. Perhaps given a more efficient encoder of arbitrary Gaussian mixtures, EM's performance would be similarly improved; this is one important direction for further research.

7.3 Non-Gaussian Field Estimation

We next consider another use of sensor networks—to fuse a collection of spatially separated observations. Suppose that we have a collection of sensors, arranged in an 8×8 regular grid. Each sensor i obtains an observation about a random variable, denoted x_i, which is known to vary slowly in space but posesses a few sharp transitions.

We employ a multi-resolution quad-tree model to capture the interactions between the x_i. Similar models have been used with considerable success for efficient estimation of Gaussian fields [29]. To be precise, we associate each of the x_i to the finest-scale (leaf) node of a quadtree which corresponds, in spatial arrangement, to sensor i. Each non-leaf node of the tree is also associated with a random variable; we will use $\gamma^1 x_i$ to indicate the random variable associated with the parent node i, $\gamma^2 x_i$ to be the parent of that node, and so forth. For notational simplicity we define $\gamma^0 x_i = x_i$.

To capture local smoothness with the possibility of sharp transitions, we model the interactions



(a) (b) (c)

Figure 10: Non-Gaussian field example. (a) An example (4×4) quad-tree structure. (b) Allocating the nodes in the quad-tree to sensors; responsibility for each parent node is assumed by one of the children. Arrows indicate the upward message sweep, from leaf nodes to root. (c) Error, in terms of KL-divergence, of the solution as a function of the allowed number of bits per message.

between variables by a simple mixture of Gaussians:

$$\begin{aligned} p(\gamma^s x_i | \gamma^{s-1} x_i) &= p(\gamma^{s-1} x_i | \gamma^s x_i) \\ &= .9N(\gamma^s x_i - \gamma^{s-1} x_i; 0, \sigma_s^2 I) + .1N(\gamma^s x_i - \gamma^{s-1} x_i; 0, I); \end{aligned}$$

where the σ_s^2 controls the smoothness, and depends on the scale s; we select $\sigma_1 = .05$, $\sigma_2 = .1$, and $\sigma_3 = .2$. The smaller, high-variance mode allows for some sharp disagreements between neighboring x_i.

Once again, we choose to represent the two-dimensional likelihood messages $p(y_i|x_i)$ as sample-based density estimates with $N = 1000$ samples, performing fusion of the messages by sampling from their products [16, 30]. In the quad-tree structure, optimal inference can be performed via a simple two-pass sequence: first, messages are passed upward from the leaf nodes to the root and fused at each level, then the fused results are sent back downwards to the leaves.

We impose the statistical quad-tree structure shown in Fig. 10(a) onto the physical sensor and communications structure by associating each of the "virtual" parent nodes to the same sensor as one of its four children. Thus, at each level in the upward sweep, three nodes transmit (and thus may wish to approximate) their messages to the "parent"; in the downward sweep, the parent node transmits (simultaneously, by broadcast) to the other children. The upward sweep is depicted in Fig. 10(b). After this process concludes, most sensors have sent only one message, while a few (about $\frac{1}{4}$) have sent two.

We compare the quality of the fusion results as a function of the number of bits allocated to each message, applying the KD-tree based approximation of Section 5. Fig. 10(c) shows the resulting KL-divergence between estimated and true posterior distributions as a function of the number of bits, averaged over 500 Monte Carlo trials. As with the particle filtering example, reasonably good results are obtained even for relatively little communications (less than 1000 bits per message).

8 Conclusions

Power-limited wireless sensor networks must be able to perform inference in a communications-constrained environment. We consider an important subset of this general task, that of inference on continuous-valued random variables using sample-based representations, the most common example of which is particle filtering. We discuss the cost of transmitting such representations, both exactly and approximately.

For exact transmissions, we showed that the representation's invariance to reordering can be used to reduce the required communications cost, and that to do so we must take advantage of

predictable non-stationarity in the distribution of the deterministically-ordered samples. We also described a simple sub-optimal linear predictive encoder which provided some of these benefits.

To treat approximate (lossy) transmissions, we applied the KD-tree data structure to the tasks of both encoding and density approximation, demonstrating how communications cost may be efficiently balanced with errors in inference. We then showed several examples demonstrating lossy encoding for distributed inference, including a distributed implementation of particle filtering and a multi-resolution model for estimating a non-Gaussian random field.

Many important questions remain, however. For example, feedback from the receiver, side information, and previously transmitted messages provide important sources of information which we have not exploited. Furthermore, although we have described a few sub-optimal encoding methods as examples, we expect that further investigation can lead to substantial improvement in methods of communicating sample-based distributions and their approximations.

References

[1] S. Kumar, F. Zhao, and D. Shepherd. Collaborative signal and information processing in microsensor networks. *IEEE Signal Proc. Mag.*, 19(2):13–14, March 2002.

[2] H. Gharavi and S. Kumar. Special issue on sensor networks and applications. *Proc. IEEE*, 91(8):1151–1153, August 2003.

[3] Neha Jain, M. Dilip Kutty, and Dharma P. Agrawal. Energy aware multipath routing for uniform resource utilization in sensor networks. In *IPSN*, pages 473–487, April 2003.

[4] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Proc. Mag.*, 19(2):61–72, March 2002.

[5] A. Gersho and R. M. Gray. *Vector quantization and signal compression.* Kluwer, Boston, 1991.

[6] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[7] M. Gastpar and M. Vetterli. Source-channel communication in sensor networks. In L. Guibas and F. Zhao, editors, *IPSN*. Springer-Verlag, 2003.

[8] Mark Girolami and Chao He. Probability density estimation from optimally condensed data samples. *IEEE Trans. PAMI*, 25(10):1253–1264, October 2003.

[9] S. L. Lauritzen. *Graphical Models.* Oxford University Press, Oxford, 1996.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufman, San Mateo, 1988.

[11] M. A. Paskin and C. E. Guestrin. Robust probabilistic inference in distributed systems. In *UAI 20*, 2004.

[12] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky. Nonparametric belief propagation for self-calibration in sensor networks. *Submitted to IEEE J. Sel. Areas Comm.*, 2004.

[13] B.W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, New York, 1986.

[14] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[15] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian bayesian tracking. *IEEE Trans. SP*, 50(2):174–188, February 2002.

[16] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *CVPR*, 2003.

[17] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, Palo Alto, California, 1994.

[18] R. H. Randles and D. A. Wolfe. *Introduction to the Theory of Nonparametric Statistics*. Wiley, New York, 1979.

[19] A. T. Ihler, J. W. Fisher III, and A. S. Willsky. Message errors in belief propagation. Technical Report 2602, MIT, Laboratory for Information and Decision Systems, 2004.

[20] M. Isard. PAMPAS: Real–valued graphical models for computer vision. In *CVPR*, 2003.

[21] M. A. Carreira-Perpinan. Mode-finding for mixtures of gaussian distributions. *IEEE Trans. PAMI*, 22(11):1318–1323, 2000.

[22] J. M. Shapiro. Embedded image-coding using zerotrees of wavelet coefficients. *IEEE Trans. SP*, 41(12):3445–3462, 1993.

[23] John C. Kieffer. A tutorial on hierarchical lossless data compression. In *Modelling Uncertainty*, volume 46 of *Internat. Ser. Oper. Res. Management Sci.*, pages 711–733. Kluwer, Boston, MA, 2002.

[24] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Comm. ACM*, 18(9):509–517, September 1975.

[25] Stephen M. Omohundro. Five balltree construction algorithms. Technical Report TR-89-063, ICSI, U.C. Berkeley, 1989.

[26] Andrew Moore. The anchors hierarchy: Using the triangle inequality to survive high-dimensional data. In *UAI 12*, pages 397–405. AAAI Press, 2000.

[27] Andrew Moore. Very fast em-based mixture model clustering using multiresolution kd-trees. In *NIPS 11*, pages 543–549, 1999.

[28] M. Aitkin and D. B. Rubin. Estimation and hypothesis testing in finite mixture models. *J. R. Stat. Soc. B*, 47(1):67–75, 1985.

[29] A. Willsky. Multiresolution markov models for signal and image processing. *Proc. IEEE*, 90(8):1396–1458, August 2002.

[30] A. T. Ihler, E. B. Sudderth, W. T. Freeman, and A. S. Willsky. Efficient multiscale sampling from products of Gaussian mixtures. In *NIPS 17*, 2003.