CLASSIFICATION USING EFFICIENT LU DECOMPOSITION IN SENSORNETS

Zille Huma Kamal, Ajay Gupta, Leszek Lilien,
Department of Computer Science, Western Michigan University, Kalamazoo, MI, USA
{zkamal, gupta, llilen}@cs.wmich.edu

Ashfaq Khokhar
Department of Computer Science, and Electrical and Computer Engineering,
University of Illinois at Chicago, IL, USA
ashfaq@uic.edu

ABSTRACT
We consider the popular application of detection, classification and tracking and their feasibility in resource constrained sensornets. We concentrate on the classification aspect, by decomposing the complex, computationally intensive signal processing Maximum-A-Posterior (MAP) classifier into simpler computationally and communicationally load balanced procedures, using a clustering approach. LU decomposition is an efficient approach for computing the inverse of covariance matrices required in the MAP classifier. We thus explore feasibility of LU decomposition in sensornets. We present power-aware and load balanced techniques for LU decomposition of the covariance matrices in sensornets alongwith their analytical and power consumption analyses.

KEY WORDS
Classification, LU Decomposition, sensor networks, collaborative processing

1. Introduction

Sensornets have been envisioned as cost effective applications for monitoring, controlling or sensing in military, agricultural, or personal areas of interest. In any of these applications, the fundamental criteria for detecting the presence or absence of an object or an environmental condition requires, 'sensing' of the features or modalities present in the environment known as the detection phase. After detecting a stimulus, classification algorithms process the sensed data to categorically determine and identify the event (stimulus). Results of this classification phase can be used to decide if a reaction is warranted by the detected event or not. In some applications [1], [2], [3], [4], [8] the classification phase is followed by a tracking phase, where the sensornet nodes work collaboratively to track the stimulus, which could be an enemy combat vehicle or the spread of hazardous chemicals in our air or water.

Consider the scenario if this sensornet was deployed in a war-torn area with the aim of detecting and tracking enemy armored vehicles headed for an army base camp. Numerous vehicles and animals can be anticipated to trigger events in the sensornet monitoring the region. For every event triggered, a cluster of sensornet nodes will be consuming energy to identify the category (classification) that relates to the event detected. This could be overwhelming and may degrade the performance and lifetime of the sensornet as todays' sensor nodes are typically resource-constrained in terms of the radio range, processor speed, memory size and power. Sensor nodes also use batteries, so energy is a precious resource. Recharging or replacing batteries is expensive and may not even be possible in some situations. Therefore, sensornet applications, such as classification, need to be extremely energy-aware.

The classification process is also computationally the most intensive process amongst the detection, classification and tracking processes. A classic approach to classification in sensornets is to use classifiers popularly used in signal processing applications. Some such notable classifiers are Linear Vector Quantization, Support Vector Machines (SVM), k–Nearest Neighbor (kNN) and Maximum A Posterior/ Maximum Likelihood. Duarte et al in [4] instigate the use of Maximum Likelihood Classifier in sensornet applications due to the minimal storage and computation requirements when compared to the other mentioned classifiers. Hence, we use the Maximum A Posterior (MAP) classifier. Irrespective of the classifier used, there are three phases involved in the classification process, the offline training and testing, followed by the deployment phase, which is an uncontrolled form of the testing phase.

The MAP classifier computation requires the inverse of the covariance matrices that represent the categories of events expected. LU decomposition is a widely used approach to computing the inverse of the matrices [20], [25]. To make this decomposition feasible for resource constraint sensornet nodes, we use folding and exploit inherent parallelism. Folding enables load-balancing and near equal work at each node, so that all nodes are depleted nearly equally, thereby increasing the lifetime of the sensornet. To exploit fine-grained as well as coarse-grained parallelism we use a clustering approach, which allows us to decompose the large matrix operations into simpler arithmetic or vector operations.

The main contributions of this paper are to provide analytical analysis and power consumption costs for an efficient, load-balanced, parallel LU Decomposition algorithm for sensornets. It is imperative to have power consumption analysis due to the inherent need for power conservation in battery operated sensornet nodes, especially those envisioned to be deployed in impervious regions where battery replacement is cumbersome. To our knowledge, no previous work has been done to study the power consumption of Maximum A Posterior classifier on a sensornet node.

In the rest of the paper, we begin by summarizing related work in the area of collaborative processing for detection, classification and tracking in section 2. In section 3, we briefly present the computations required in the three phases of classification and discuss the need to compute the inverse of covariance matrix. We then describe our clustered approach to exploit parallelism inherent in distributed sensornets and use it for developing an efficient, load balanced LU Decomposition of the covariance matrix, and present its analytical analysis in section 4. Power consumption analysis of the decomposition is presented in section 5 followed by discussion of analytical and power consumption analysis, in section 6. We summarize our findings, and give future direction with our conclusion in section 7.

2. Related Work

Various researchers, for example [1], [3], [4], to mention a few, have studied and presented complete methodologies for detection, classification and tracking in sensor networks. However, their test bed and sensing platform was composed of WINS 3.0 Sensoria nodes ([12], [13]) that are larger in size and more powerful compared to MICA2 Motes [14], or Intel Motes [17] or the envisioned dust size COTS Dust ([15],[16]). In this paper, we use sensors or nodes interchangeably to generalize small resource constrained motes, like MICA2 or Intel Motes. In [5], we conducted an analytical study to evaluate the feasibility of executing computational intensive classification algorithms and concluded that there was an acute tradeoff between accuracy and efficiency. Furthermore, it became apparent that collaborative signal processing algorithms for classification – an approach presented in [1], is computationally and communicationally intensive and unpractical for today's resource constraint sensornet nodes.

Arora et al, in [7], instigate the use of influence fields for classification, which are more efficient than signal processing classifiers. While, [9] and [10] present yet another alternate approach to the signal processing classification scheme, in which mobile agents such as robots or humans are used to gather data collected at sensor nodes and transfer it to a command/base station for processing and compare it with the traditional client/server computing paradigm. While this approach may be appropriate in certain situations, it seems infeasible classification methodology for applications that are envisioned to be deployed in monitoring discommodious, insecure regions. For efficiency and power conservation, [8] directs the use suboptimal classifiers.

Another aspect of classification is the aggregation of information. Fusion of information has been dealt with in [11] and [6]. Authors in [11] use fusion rules to track targets without discussing challenges in identifying or classifying targets, where as Dynamic Fusion (DFuse) APIs of [6] are meant for higher end computing devices such as IPAQ and dynamically relocate roles, such as filtering and collage amongst nodes in sensornet to suit purpose/application.

Our main contribution is to present an analytical analysis, followed up by power consumption analysis, for classification using LU Decomposition in order to study the feasibility of this type of classification in resource constraint sensornets.

We begin our discussion, by briefly presenting the background details of the various phases involved in classification and insights to the different classifiers available. Next, we delineate in detail the data distribution, computations and communications required in the classifier, using our clustered approach. This will enable us to utilize parallelism in computation and communication that is inherent in distributed sensornets. The discussion is followed up by analytical analysis and power consumption analysis of LU Decomposition.

3. Background

Classification is the process of determining a category given a sample of the environment. There are three phases in the classification process, training, testing and deployment. In the training phase, the sensors in the sensornet are manually/intentionally exposed to N known sample (readings or measurements of environment) event feature matrices from each of the k predetermined categories. The user interactively classifies the events for the sensors. An event is an $f \times d$ feature matrix, where f is the number of features monitored in the environment and d is the dimension of the temporal processing. In terms of signal processing, f represents the bands measured. Based on these sample event feature matrices for a category, each sensor locally computes a mean and covariance matrix for that category. The mean, μ, and covariance, δ, matrices are formulated as

$$\mu_{i,j} = \frac{1}{N}\sum_{\alpha=1}^{N} M_{i,j}^{\alpha}$$

Equation 1

$$\delta_{i,j} = \frac{1}{N-1}\sum_{\alpha=1}^{N}\left(M_{i,j}^{\alpha} - \mu_{i,j}\right)\left(M_{i,j}^{\alpha} - \mu_{i,j}\right)^{T}$$

Equation 2

$M_{i,j}^{\alpha}$ is the event feature matrix from the training phase at sensor i for category j, if n_0 is the total number of sensors in the sensornet then $1 \le i \le n_0$ and $1 \le j \le k$ and α indicates event feature matrices that are collected in the training phase.

In the testing phase, the mean and covariance matrices are when sensors classify test sample event feature matrices into one of the k predetermined categories. In doing so, a belief probability can be constructed to represent a sensor's accuracy. The testing phase is the controlled version of the deployment phase.

In the deployment phase, sensors are deployed in the region to be monitored, unmonitored themselves, and repeat the classification process as in the testing phase and reach a decision as to whether the object detected is of interest or not and consequentially to be tracked or not. This scenario is illustrated in figure 2.

In the classification phase it is imperative to choose the right classifier. Duarte et al in [4] give insights as to the feasibility of Maximum Likelihood Classifier in sensornet applications due to the minimal storage and computation requirements in comparison to k–Nearest Neighbor (kNN), Support Vector Machines (SVM) and Linear Vector Quantization. We build on this assertion. Furthermore, if we assume equi-probable likelihoods, which is justified in [22], we can rewrite the Maximum Likelihood classifier as the Maximum A Posterior (MAP) classifier. The MAP classifier can be further simplified, making general assumptions [5], and presented as computation of the mahalanobis distance between the mean of each category and the event/object detected as presented in equation 3 and pictorially represented in figure 1. Therefore, we classify the object detected into the category that minimizes the mahalanobis distance (minimum $\left\|MAP_{i,j}\right\|_F$ value).

$$MAP_{i,j}^{\beta} = \left(M_{i,j}^{\beta} - \mu_{i,j}\right)^{T}\delta_{i,j}^{-1}\left(M_{i,j}^{\beta} - \mu_{i,j}\right)$$

$$\left\|MAP_{i,j}^{\beta}\right\|_F = \sum_{p=1}^{d}\sum_{q=1}^{d} MAP_{i,j}^{\beta}(p,q)$$

Equation 3

Again, $M_{i,j}$ represents an event feature matrix collected at sensor i for some *unknown* category j, and $\left\|MAP_{i,j}\right\|_F$ represents the frobenius norm of the $d \times d$ MAP matrix. Here, β is used to indicate event feature matrices that are collected during the testing phase.

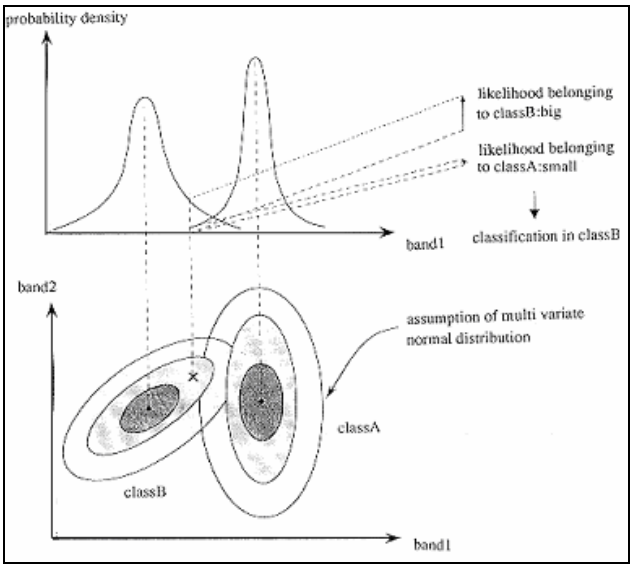

Figure 1 – Maximum Likelihood and Mahalanobis distance [21]

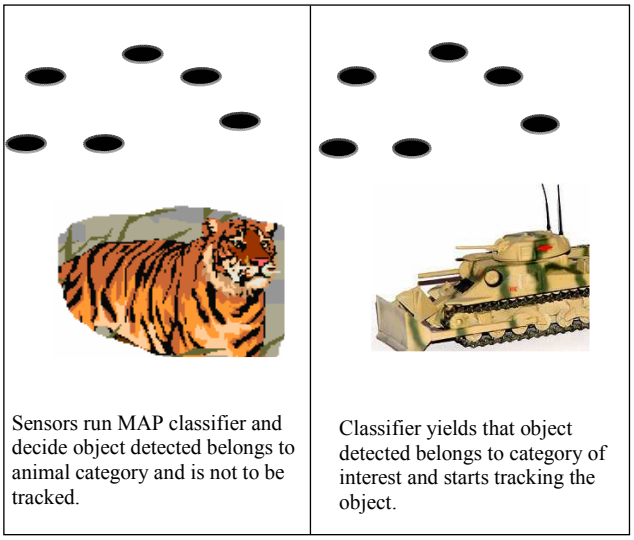

Figure 2 – Classification during deployment phase

3.1 Data Collection

Typically, f is around 50 and d is around 512, which means we are asking our resources constrained sensors to work with **large matrices** and **matrix operations**. This depletes precious resources (energy) of the sensornet, making it infeasible to monitor an important, cumbersome area over a long period of time. Therefore, we design dense clusters of sensors within the sensornet.

Each cluster will be comprised of d sensors, the larger dimension of the event feature matrices. This will enable us to exploit parallelism to accomplish the computational intensive classification algorithm, quickly, efficiently and in a load balanced manner. We illustrate this clustering in figure 3.

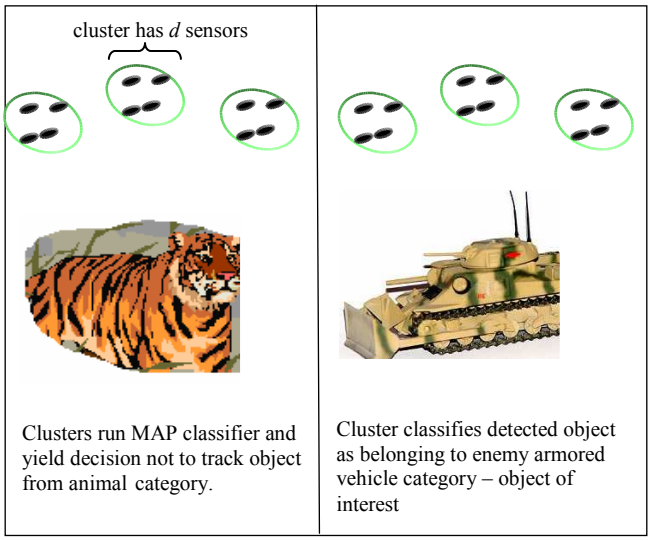

Figure 3 – Clustering to exploit parallelism

The $f \times d$ event feature matrix requires d-dimension (point) FFT on each of the f features. A load-balanced and efficient way to compute FFT distributively in a sensornet is presented in [18]. Each of our d sensors in a cluster, stores a column of an event feature matrix, that is, f elements, as illustrated in figure 4, where x is the x^{th} column of the event matrix, each element being a component of FFT.

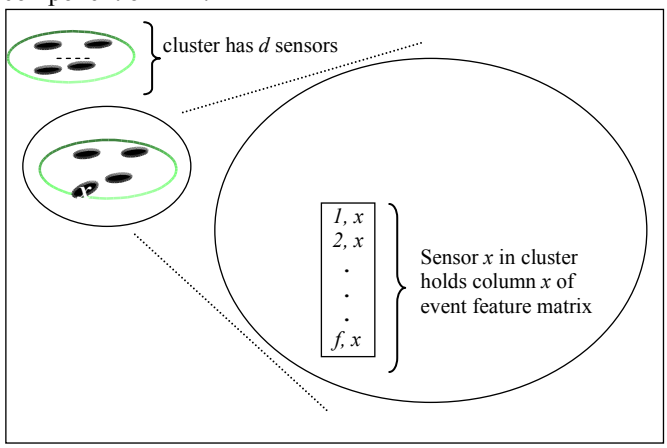

Figure 4 - Data distribution in a cluster node

3.2 Computation of MAP Classifier

For the MAP classifier, the most complex computation is the inverse of the covariance matrix, as formulated in equation 3, for which we need to find fast, stable and efficient algorithms. A school of thought may consider offloading the computation of the inverse of the covariance matrices to a more powerful node such as a personal computer. However, this will bind the classifier to be static and inhibit its ability to dynamically update its mean and covariance matrices, when deployed, to yield more accurate results.

Therefore, to enable dynamic updates, we scrutinize existing techniques for efficiently and accurately computing the inverse of the covariance matrix. Traditional inverse computation of an $f \times f$ matrix requires $O(f^3)$ operations [20], which is slow, inefficient, un-balanced, serial and tedious. Other less computationally intensive methods include LU decomposition of the matrix. We decompose the covariance matrix into lower (L) and upper (U) triangular matrices, so $\delta = LU$. With this decomposition, the inverse can be formulated as the product

$$\delta_{i,j} = LDL^T \Rightarrow \delta_{i,j}^{-1} = LD^{-1}L^T \qquad \text{Equation 6}$$

where D is a diagonal matrix, whose diagonal entries are the diagonal entries of U. This allows the following simplification

$$MAP_{i,j} = \left(M_{i,j} - \mu_{i,j}\right)^T LD^{-1}L^T \left(M_{i,j} - \mu_{i,j}\right)$$
$$\Rightarrow MAP'_{i,j} = \left(M'_{i,j} - \mu'_{i,j}\right)^T D^{-1} \left(M'_{i,j} - \mu'_{i,j}\right) \qquad \text{Equation 7}$$

The multiplication with D^{-1} is in essence a scalar multiplication of column x of a matrix with $1/\lambda_x$ the diagonal entry in column x of D.

Our goal now is to present how to perform this LU decomposition efficiently and to conduct an analytical analysis in the clustered sensornet, which we present in the next section.

4.0 LU Decomposition and its Analytical Analysis

The LU decomposition algorithm with partial pivoting for stability is presented in figure 5, below. A square matrix δ is decomposed so that the lower triangle hold entries of L and the principal diagonal and all entries above it are the entries of U.

The LU Decomposition is composed of three steps, the pivot search, division, and elimination steps. The pivot search and division step is column-oriented, where as the elimination step is row-oriented computations. There are two possibilities of data distribution among the sensors of a cluster: row-wise and column-wise.

We consider both row-wise and column-wise data distribution, and compare the computational and communicational costs incurred in each of the steps of the decomposition to identify efficient data distribution and decomposition computation. We use a folding technique that allows us to achieve higher efficiency since all sensors are kept busy in all iterations of the algorithm, making LU Decomposition load-balanced and efficient.

In our analysis, we take our data values to be floats stored in 4 bytes, and our communication costs provide for the fact that multiple messages may be required to transmit f elements. We use the term pack message, to imply that

payload of message is full to capacity. The payload of a TinyOS packet is 29 bytes [23].

```
procedure LU_Decomposition (δ)
(1)     for p = 0 to f-1 do
(2)         max = δ[p,i]
(3)         for q = p+1 to f do
(4)             if δ[q,p] > max
(5)                     record  in PIVOT(p)
                    /*pivot search step*/
(6)         end for
(7)         if PIVOT(p) <> p
(8)             swap δ[p,*] & δ[PIVOT(p),*]
(8)         for q = p+1 to f do
        /*division step*/
(9)             δ[q,p] = δ[q,p]/ δ[p,p]
(10)        end for
(11)        for q = p+1 to f do
(12)            for r = p+1 to f do
        /*elimination step*/
(13)            δ[q,r] -= δ[q,p] * δ[p,r]
(14)            end for
(15)        end for
(16)    end for
end procedure
```

Figure 5 – LU Decomposition Algorithm [20, modified to fit our example]

4.1 Row wise distribution of data with folding

The folding technique applied to row-wise data distribution is illustrated in figure 6, where, each sensor holds two rows of the symmetric covariance matrix. The costs incurred in each step of the algorithm, in iteration x at a sensor x are:

- Pivot search – sensor x broadcasts a message with the pivot element. The same message is packed with as many elements of the row that can be accommodated in that message. All sensors receive the message and extract the first element in the message and compare it with the local element in column x (pivot position). If the local pivot element is greater than the pivot element received, then this sensor broadcasts a message that is also packed (with as much of the row as possible – as described earlier).

  In the worst case, every sensor receives $f-x$ messages, where all $f-x$ elements in column x are compared to find the pivot element. Now, the sensor that owns the largest element can swap its row with sensor x, where both sensors broadcast their entire row. While, all other sensors also receive the pivot row, reducing the need for an additional row broadcast for division and elimination steps of the iteration.

- Division and Elimination Step – all sensors but sensor x, compute the local multipliers. These multipliers are then used locally together with the pivot row for the elimination steps.

Thus, it is seen that the computational and communicational costs move top to bottom with row-wise distribution of the matrix. Table 1, in appendix A tabulates the exact communicational and computational costs incurred, in terms of number of operations, number messages sent and number of messages received, as per the outline above.

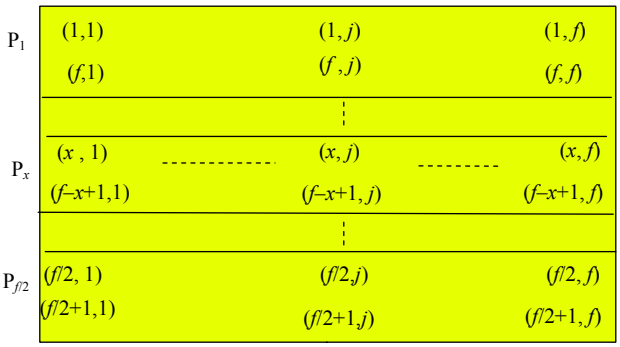

Figure 6 – Row wise data distribution with folding

4.2 Column wise Distribution of data with folding

We compare the costs incurred in row-wise data distribution with those of column-wise data distribution. Here, we distribute two columns of the covariance matrix to each sensor in the cluster, as depicted in figure 7. Here, communicational and computational costs move left to right.

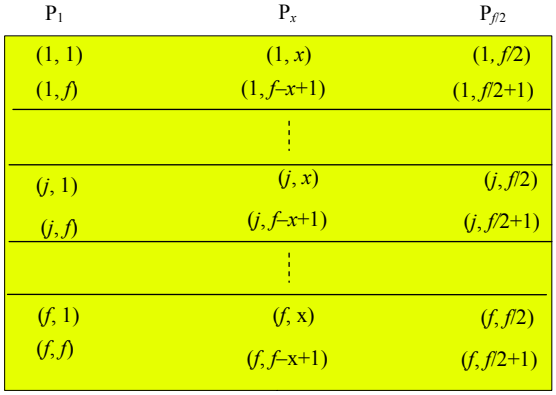

Figure 7 – Column wise data distribution with folding

Similarly, the costs incurred in each step of the algorithm, in iteration x at a sensor x are:

- Pivot Search - is a local comparison of all $f-x+1$ elements, which implies, $f-x$ comparisons. In the worst case, there will be row swap in all iterations. For a row swap, sensor x broadcasts a message with two indices, representing the two element positions to be swapped and the pivot element. All sensors receive the broadcast and locally swap the two elements.

- Division Step – sensor x then computes the multipliers in the division step. These multipliers are then broadcast in $\left\lceil \frac{4(f-x)}{29} \right\rceil$ packed messages.

- Elimination Step – all other sensors receive the messages with the multipliers and can compute the local resident elimination.

Table 2, in appendix A, indicates the communicational and computational costs incurred by a sensor in the cluster, with folding and column-wise data distribution.

4.3 Cost of computing MAP with the decomposed covariance matrix

Each of the n_0 clusters will compute k decompositions, one for each covariance matrix of a category. That is, in the order of $O(f^2k)$ computations and communications, over $f/2$ sensors in a cluster, as per tables 1 and 2 in appendix A. Therefore, across the n_0 clusters in the sensornet cost of decomposing the covariance matrices is $O(f^2kn_0)$ for categories. This cost is not recurring, unless it is dynamically updating the covariance matrix to achieve higher accuracy. By using the decomposition of the covariance matrix for a category we can compute the MAP matrix as

$$MAP_{i,j} = \left(M_{i,j} - \mu_{i,j}\right)^T LD^{-1}L^T\left(M_{i,j} - \mu_{i,j}\right)$$

where D is the diagonal entries of U. And the mahalanobis distance can be computed as

$$\left\|MAP_{i,j}\right\|_F = \sum_{p=1}^{d}\sum_{q=1}^{d} MAP_{i,j}(p,q)$$

If we let $Z = \left(M_{i,j} - \mu_{i,j}\right)^T L$, then $\left(M_{i,j} - \mu_{i,j}\right)L^T = Z^T$. So we only need to compute Z (or Z^T). Therefore, the cost of computing Z can be decomposed into:

- fd subtractions for computing $M_{i,j} - \mu_{i,j} = Y$
- fd dot products for computing $L^T Y$, where each dot product costs $2f-1$ operations.

Thus, total cost for computing Z (or Z^T) is $2f^2d$. The communicational cost is $O(f)$, exchanging all the rows (or columns) of E (or E^T).

Thus, computing the MAP matrix can be defined as:

- d^2 dot products, each costing $2f-1$ basic arithmetic operations
- scalar multiplication of the columns of Z with the diagonal entries of U.

This requires broadcast of all the columns of matrix Z, a communicational cost of $O(d)$.

Thus, the total computational cost of computing MAP is $O(2d^2f + 2df^2 - d^2)$ with a communicational cost of $O(d+f) \approx O(d)$, since $d >> f$, over the d sensors in a cluster.

5.0 Power Consumption Analysis

We now consider power consumption of the sequential and parallel versions of the LU Decomposition in a sensornet. Various authors [19], [24], have studied the average power consumption of a Mica type sensor node in various modes such as computation (only CPU active), transmission of a message, reception of a message, etc. We ignore the costs related to computing the mean and covariance as those computational costs are insignificant compared to the costs incurred in the MAP classifier. We use the power ratings presented in [24], where the power consumption is approximately, 10 mA when radio is transmitting, about 7 mA when radio is receiving and about 8mA when only the CPU is operational (that is, arithmetic logic unit operations)

Earlier we considered analytical analysis with respect to the number of messages sent, received and number of basic arithmetic operations, that is computations performed, we now incorporate this analysis for power consumption based on the above given power consumption ratings.

5.1 Sequential LU Decomposition

The LU decomposition algorithm running locally on one sensor, not a cluster, in the network would not incur any communication costs as all data would be stored locally, however the sensor would incur $\frac{4}{3}f^3 + 3f^2 - \frac{7}{3}f - 1$ CPU operations, with worst case assumption that rows are swapped in every step of the iteration. Therefore, the total power consumed at a sensor is $\frac{32}{3}f^3 + 24f^2 - \frac{56}{3}f - 1$ mA.

5.2 Parallel LU Decomposition

The analytical analysis for the parallel LU decomposition algorithm with respect to the row-wise and column-wise data distribution, were tabulated in table 1 and 2 of appendix A, respectively. These analytical results are used for the power consumption analysis presented in the sections that follow.

5.2.1 Row-wise distribution and folding

In the worst case, the number of messages sent $= \frac{4}{29}(2f) + f + 1$, which implies the power consumed $= 10\left(\frac{4}{29}(2f) + f + 1\right)$ mA.

Similarly, the number of messages received was = $\frac{1}{29}\left(\frac{3f^2}{2}-f\right)+\frac{3f^2}{8}-\frac{f}{4}$ that is equivalent to $7\left(\frac{1}{29}\left(\frac{3f^2}{2}-f\right)+\frac{3f^2}{8}-\frac{f}{4}\right)$ mA or power.

Computational costs incurred were $2f^2+f-3$ operations, which is $8(2f^2+f-3)$ mA of power consumption.

Thus, the maximum power (in mA) consumed at a sensor is

$$8(2f^2+f-3)+$$
$$7\left(\frac{1}{29}\left(\frac{3f^2}{2}-f\right)+\frac{3f^2}{8}-\frac{f}{4}\right)+10\left(\frac{4}{29}(2f)+f+1\right)$$

5.2.2 Column-wise distribution with folding

Similarly, table 2 in appendix A, shows at worst, number of messages sent are $\frac{4}{29}\left(\frac{f}{2}\right)+\frac{4}{29}\left(\frac{f}{2}-1\right)+2$, which is, $10\left(\frac{4}{29}\left(\frac{f}{2}\right)+\frac{4}{29}\left(\frac{f}{2}-1\right)+2\right)$ mA of power consumed.

$\frac{4}{29}\left(\left(\frac{f}{2}\right)\left(\frac{3f-6}{4}\right)\right)+f-3$ is the number of messages received that is proportional to $7\left(\frac{4}{29}\left(\left(\frac{f}{2}\right)\left(\frac{3f-6}{4}\right)\right)+f-3\right)$ mA of power consumed.

Computational costs in terms of arithmetic operations (additions and multiplications) were $\frac{3f^2}{2}+2f-4$ that is $8\left(\frac{3f^2}{2}+2f-4\right)$ mA of power consumed.

Therefore, the maximum power consumed at a sensor is

$$8\left(\frac{3f^2}{2}+2f-4\right)+$$
$$7\left(\frac{4}{29}\left(\left(\frac{f}{2}\right)\left(\frac{3f-6}{4}\right)\right)+f-3\right)+$$
$$10\left(\frac{4}{29}\left(\frac{f}{2}\right)+\frac{4}{29}\left(\frac{f}{2}-1\right)+2\right) \text{ mA.}$$

6.0 Discussion and comparison

The covariance matrix is the variance between features, and hence it is a square matrix with dimension *f*, where *f* is the number of modalities or features sensed in the environment. Thus, the inverse of the covariance matrix is also square with dimension *f* and hence the analytical analysis is directly dependent on the number of features observed.

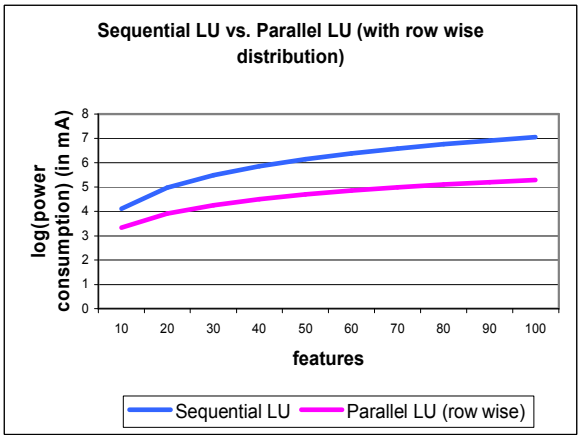

Figure 8 – Comparison of parallel row-wise LU decomposition with sequential LU decomposition

Therefore, the total power consumed is plotted against varying values *f*. Figure 8 and 9 illustrate the power consumption is parallel row-wise and column-wise data distribution compared to sequential LU decomposition

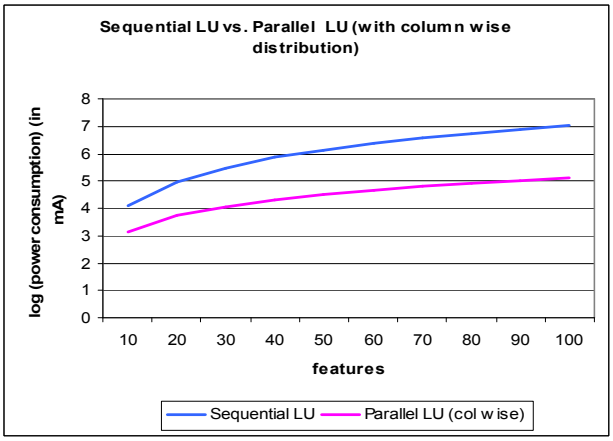

Figure 9 – Comparison of parallel column-wise LU decomposition with sequential LU decomposition

6.1 Comparison

A comparison of the number of messages sent, received and number of operations performed in row-wise and column-wise data distribution in table 1 and 2 in appendix A indicate that the LU decomposition algorithm with the

applied folding technique is nearly load-balanced (thereby depleteing the resources of all the sensors equally which results in an increase in network lifetime).

Recall that LU decomposition is composed of three steps, the pivot search, division, and elimination steps. The pivot search and division step is column-oriented, where as the elimination step is row-oriented computations. Thus, one may hypothesize that the data distribution used should be column-oriented to minimize communication costs. Furthermore, our hypothesis was to use column-wise data distribution to minimize communication, as per the total number of messages exchanged during the algorithm, and from figure 10 it is evident that this hypothesis holds true. With column-wise data distribution we minimize messages exchanges by orders of magnitude.

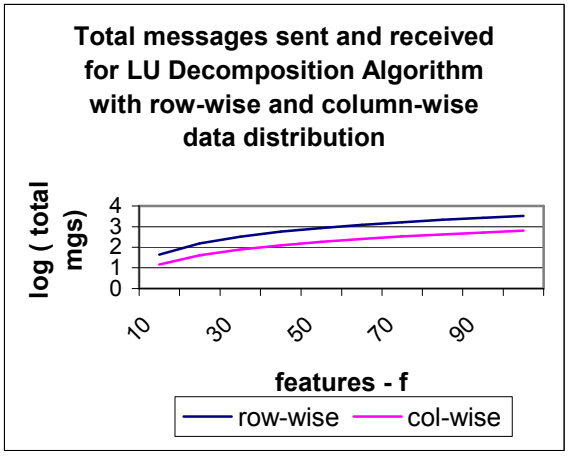

Figure 10 - Comparing total communication costs in terms of messages sent and received with row-wise and column-wise data distribution

The column-wise data distribution also indicates orders of magnitude lower power consumption when compared to row-wise and sequential data distribution, figure 11.

Both, the row-wise as well as column-wise distributions parallelize well, and reduce computational costs from $O(f^2)$, to $O(f^3)$, the computational cost for classic inverse computation algorithm.

Furthermore, the order of magnitude of the computations and communications required indicate that these collaborative signal processing algorithmic approaches may not yet be practical for sensornets (at least the ones consisting of mica motes).

7.0 Conclusion

In this paper, we used a clustering approach for LU decomposition as a technique to try to overcome the computationally intensive inverse computation of covariance matrix that is needed in the Maximum A Posterior classifier.

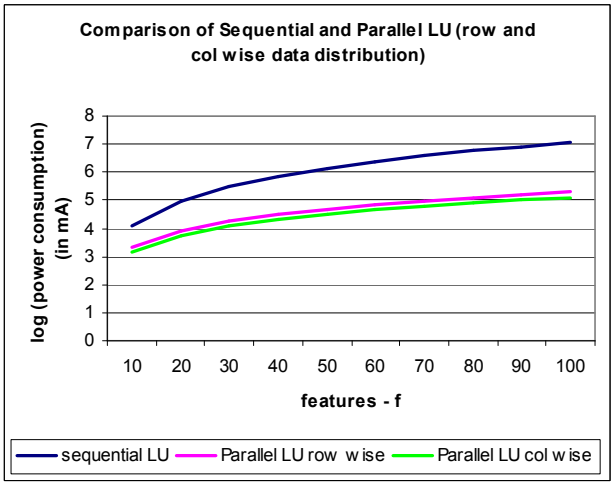

Figure 11 – Comparison of parallel LU decomposition with row wise and column-wise data distribution and sequential LU decomposition

This clustering approach enabled us to exploit parallelism that is inherent in distributed sensornets. With parallelism, we can achieve faster results and can ensure that no one node is extensively depleted of its power resources. Furthermore, if we do not distribute the workload amongst the sensornet nodes equally, then the expiration of the node which had the ability to compute the inverse will also significantly degrade the performance and effectiveness of the sensornet (sensornet may become disconnected). Alternatively, one may employ exponentially more powerful nodes (such as base-stations) within each cluster; however, this would significantly decrease the ad-hoc nature of sensornet deployment.

We also used a folding technique to yield higher efficiency of sensornet nodes, which ensures that no node is idle during execution of the LU decomposition algorithm and that workload is approximately load balanced across the sensornet nodes.

We extrapolate the analytical analysis to infer power consumption of the LU decomposition algorithm with row-wise and column-wise data distribution. Power consumption analysis should be an important aspect of algorithm development for resources constraint sensornets. Primarily because sensornets are envisioned to be used in applications and regions previously unimagined and unreachable, therefore, if the algorithms used are not power efficient it may not be possible to replace the batteries or recharge the energy of the sensornets nodes, reducing the effectiveness of the sensornets.

To our knowledge, we are the first to conduct analytical and power consumption analysis for LU decomposition in sensornets.

These analyses give insight to the feasibility / infeasibility of collaborative signal processing classifiers for sensornets applications. Therefore, our future work includes a new approach to the classification process so that it is not dependent on signal processing algorithms, as in [8], or an alternate approach to inverse computation, or an alternate computation of the Maximum A Posterior classifier that eliminates the need for inverse of covariance matrices altogether.

Acknowledgements

Research is supported in part by the National Science Foundation, under grants ACI-0000442, ACI-0203776, and MRI- 0215356, by the Department of Education grant R215K020362, and a Congressional Award, administered by the US Department of Education, Fund for the Improvement of Education. The authors would also like to acknowledge Western Michigan University for its support and contributions to the Wireless Sensor Network Research Laboratory, Computational Science Center and Information Technology and Image Analysis (ITIA) Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies or institutions. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies or institutions.

References

[1] D. Li, K. Wong, Y. Hu, and A. Sayeed, Detection, Classification, tracking of targets in micro-sensor networks. *IEEE Signal Processing Magazine*, March 2002.

[2] A. D'Costa and A. M. Sayeed, Collaborative signal processing for distributed classification in sensor networks. Lecture Notes in Computer Science (Proceedings of IPSN'03), (Springer-Verlag, Berlin Heidelberg), pp. 193–208, (F. Zhao and L. Guibas (eds.), April 2003.

[3] M. Duarte and Y. H. Hu, Distance Based Decision Fusion in a Distributed Sensor Network. International Symposium on Information Processing in Sensor Networks (IPSN) 2003.

[4] Marco Duarte and Yu-Hen Hu, Vehicle Classification in Distributed Sensor Networks. Journal of Parallel and Distributed Computing, Vol. 64 No. 7, pp. 826-838, 2004.

[5] Z. H. Kamal, M. A. Salahuddin, A. Gupta, M. Terwilliger, V. Bhuse, and B. Beckmann, Analytical Analysis in Decision and Data Fusion. The Proceedings of 2004 Conference on Embedded Systems and Applications, June 2004.

[6] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran, DFuse: A Framework for Distributed Data Fusion. ACM SenSys 2003, Los Angeles, CA, Dec. 2003.

[7] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora and M. Miyashita, A Line in the sand: A wireless sensor network for target detection, classification, and tracking. Computer Networks (Elsevier), pp. 605-634, 2004.

[8] J. H. Kotecha, V. Ramachandran, and A. M. Sayeed, Distributed Multitarget Classification in Wireless Sensor Networks. IEEE Journal on Selected Areas in Communications, Vol. 23, No. 4, April 2005.

[9] Y. Xu, and H. Qi, Distributed Computing Paradigm for Collaborative Signal and Information Processing in Sensor Networks. Journal of Parallel and Distributed Computing, August 2004.

[10] H. Qi, Y. Xu, and X. Wang, Mobile-agent-based collaborative signal and information processing in sensor networks. Proceedings of IEEE, Special Issue on Sensor Networks and Applications 91 (8) (2003) 1172–1183.

[11] J. M. F. Moura, J. Lu, and M. Kleiner, Intelligent Sensor Fusion: A Graphical Model Approach. IEEE International Conference on Acoustic, Speech and Signal Processing, Hong Kong, April 2003

[12] Sensoria Corporation, http://www.sensoria.com/

[13] WINS 3.0 Sensing Platform, http://www.sensoria.com/pdf/WINS-3.0-Wireless-Sensing-Platform.pdf

[14] Crossbow Technology Incorporated, http://www.xbow.com/

[15] Dust Networks, http://www-bsac.eecs.berkeley.edu/archive/users/warneke-brett/SmartDust/index.html

[16] B. Warneke, M. Last, B. Leibowitz, K.S.J. Pister, Smart Dust: Communicating with a Cubic-Millimeter Computer. IEEE Computer, Jan. 2001, Computer Society, Piscataway, NJ. pp. 44-51.

[17] Intel Corporation, 2006, http://www.intel.com/research/exploratory/motes.htm

[18] T. Canli, M. Terwilliger, A. Gupta and A. Khokar, Power-Time Efficient Algorithm for Computing FFT in Sensor Networks. ACM SenSys '04, Baltimore, MD, November 2004.

[19] M. Srivastava, Sensor node platforms and energy issues. Mobicom 2002 Tutorial.

[20] A. Grama, An Gupta, G. Karypis, and V. Kumar, Intorduction to Parallel Computing Second Edition. Addison Wesley, New York, NY, 2003

[21] Online Resource, Last Accessed March 12, 2006 http://www.profc.udec.cl/~gabriel/tutoriales/rsnote/cp11/11-7-1.gif

[22] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd Edition. John Wiley & Sons, Canada, 2001.

[23] TinyOS Tutorial, 23 September 2003, online resource, last accessed March 14, 2006 http://www.tinyos.net/tinyos-1.x/doc/tutorial/ lesson4.html

[24] V. Shnayder, M. Hempstead, B. Chen,G. W. Allen, and M. Welsh, Simulating the Power Consumption of Large Scale Sensor Network Applications. ACM SenSys 04.

[25] S. J. Leon, Linear Algebra with Applications, 5th Edition. Prentice Hall, Upper Saddle River, NJ, 1998.

Appendix A

$$\frac{4}{29}(2f)+f+1+\frac{1}{29}\left(\frac{3f^2}{2}-f\right)+\frac{3f^2}{8}-\frac{f}{4}\text{ messages sent or received}=\frac{75f^2}{232}+\frac{115f}{116}+1\text{ messages}$$

communicated with row-wise.

$$\frac{4}{29}\left(\frac{f}{2}\right)+\frac{4}{29}\left(\frac{f}{2}-1\right)+2+\frac{4}{29}\left(\left(\frac{f}{2}\right)\left(\frac{3f-6}{4}\right)\right)+f-3\text{ messages sent or received}=\frac{3f^2}{58}+\frac{60f}{58}-\frac{33}{29}\text{ messages}$$

exchanged.

PE	Columns held	Messages Received	Messages Sent	Computation
1	1, f	$\frac{4}{29}(f-1)+f-1$	$\frac{4}{29}(2f)+f+1$	$2f^2+2f-3$
2	2, f–1	$\frac{4}{29}(2f-3)+2f-3$	$\frac{4}{29}(2f)+f+1$	$2f^2+3f-7$
. .	. .	. .	. .	. .
i	i, f–i+1	$\frac{4}{29}\left(fi-\frac{i(i+1)}{2}\right)+fi-\frac{i(i+1)}{2}$	$\frac{4}{29}(2f)+f+1$	$(f-i+1)(2f+i-2)$ $+2fi-i(i+1)+f-1$
. .	. .	. .	. .	. .
$f/2-1$	$f/2$-1, $f/2$+2	$\frac{1}{29}\left(\frac{3f^2}{2}-3f\right)+\frac{3f}{4}\left(\frac{f}{2}-1\right)$	$\frac{4}{29}(2f)+f+1$	$2f^2+\frac{f}{2}-7$
$f/2$	$f/2$, $f/2$+1	$\frac{1}{29}\left(\frac{3f^2}{2}-f\right)+\frac{3f^2}{8}-\frac{f}{4}$	$\frac{4}{29}(2f)+f+1$	$2f^2+f-3$

Table 1 – Computation and communicational costs in terms of number of operations and messages transmitted, with row-wise distribution.

PE	Columns held	Messages Received	Messages Sent	Computation
1	1, f	$\frac{2}{29}(f-1)\left(\frac{f}{2}\right)+f-3$	$\frac{4}{29}(f-1)+1$	f^2+3f-4
2	2, f–1	$\frac{2}{29}(f-2)\left(\frac{f-1}{2}\right)$ $+f-3$	$\frac{4}{29}(f-2)+2$	f^2+5f-8
.				
I	i, f–i+1	$\frac{4}{29}\left((f-i)\left(\frac{f+i-3}{2}\right)\right)$ $+f-3$	$\frac{4}{29}(f-i)$ $+\frac{4}{29}(i-1)+2$	$f^2-2i^2+f+2i+2fi-4$
. .	. .	. .	. .	. .
$f/2-1$	$f/2$-1, $f/2$+2	$\frac{4}{29}\left(\left(\frac{f}{2}+1\right)\left(\frac{3f}{4}-2\right)\right)$ $+f-3$	$\frac{4}{29}\left(\frac{f}{2}+1\right)+\frac{4}{29}\left(\frac{f}{2}-2\right)$ $+2$	$\frac{3f^2}{2}+2f-8$
$f/2$	$f/2$, $f/2$+1	$\frac{4}{29}\left(\left(\frac{f}{2}\right)\left(\frac{3f-6}{4}\right)\right)$ $+f-3$	$\frac{4}{29}\left(\frac{f}{2}\right)+\frac{4}{29}\left(\frac{f}{2}-1\right)$ $+2$	$\frac{3f^2}{2}+2f-4$

Table 2 – Computation and communicational costs with column wise data distribution and folding.