Quantization for Distributed Estimation using Neural Networks

Vasileios Megalooikonomou

Department of Computer and Information Sciences
Temple University
1805 N. Broad Street
303 Wachman Hall
Philadelphia, PA 19122

and

Yaacov Yesha

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250

Corresponding author:

Vasileios Megalooikonomou
Department of Computer and Information Sciences
Temple University
1805 N. Broad Street
303 Wachman Hall
Philadelphia, PA 19122
Tel: 215.204.5774
Fax: 215.204.5082
E-mail: vasilis@cis.temple.edu

Abstract

We propose a neural network approach for the problem of quantizer design for a distributed estimation system with communication constraints in the case where the global observation model is unknown and one must rely on a training set. Our method applies a variation of the Cyclic Generalized Lloyd Algorithm (CGLA) on every point of the training set and then uses a neural network for each quantizer to represent the training points and their associated codewords. The codeword of every training point is initialized using a previously proposed regression tree approach. Simulation results show that there is an improvement of the proposed approach over using regression trees only.

1 Introduction

Networks of embedded sensors have started to become increasingly important especially due to their potentially enormous impact in environmental monitoring, product quality control, defense systems, etc. Several researchers have worked on developing protocols and systems architectures for networks of thousands of embedded devices [5, 7, 4]. At the same time, new exciting technologies such as MEMS (MicroElectroMechanical Systems) [2] devices (CMU) and Smart-Dust project [8, 15] (Berkeley) are expected to expand the capabilities of embedded devices and networks of sensors by putting a complete sensing/communication platform, including power supply, analog and digital electronics, inside a cubic millimeter.

In distributed estimation systems several separated processing nodes (i.e., sensors) observe an environment, collect information, and make estimations based on their own observations and information being communicated between the nodes. The model of a distributed estimation system that we consider here consists of a single fusion center and a number of remote sensors. This model has many applications to radar, sonar and remote-sensing systems. In this scheme, the fusion center estimates some unobserved quantities based on observations collected by remote sensors and transmitted to the center. Restrictions on this model such as the capacity constraints on the communication lines suggest some very challenging problems.

The exact model that we consider here is described below. The sensors are not allowed to communicate with each other and there is no feedback from the fusion center back to them. The communication channels are assumed to be error free. The observations from the sensors are vector quantized before the transmission to the fusion center in order to satisfy the communication constraints. Thus, the estimation is achieved via compressed information. We assume fixed length coding for the transmission. The observations at the sensors are random. Here, we consider the case where the joint probability density function is unknown.

The problem is defined as follows: For a distributed system with k sensors, find, for each sensor, a mapping from the observation space to codewords (of a certain number of bits given by the capacity constraints), and find a fusion center function that maps a vector of k codewords to an estimate vector for the unobserved quantities, so that the mean of the square of the Euclidean norm of the estimation error is minimized. There is a joint probability distribution of all observations and unobserved quantities. However, since this distribution is unknown, the design of the system is based on a training set and the mean squared error is computed based on a test set. Although the number of sensors, k, can be in general arbitrary, here we consider the two-sensor case since the method for this case can be easily extended to the more general case.

An approach based on a generalization of regression trees for the problem of quantizer design for such a distributed estimation system in the case of unknown observation statistics has been given by Megalooikonomou and Yesha [13, 14]. The same problem in the case of known probability model was considered by Lam and Reibman [10, 11]. Gubner [6] considers the problem of quantizer design for this system subject not only to communication constraints but also to computation constraints at the fusion center in the case of known observation statistics. Longo *et al* [12] consider the problem of quantization for a distributed hypothesis testing system.

In this paper, we consider the problem of quantizer design subject to communication constraints in the case where the joint probability model is unknown and only a training sequence is available. We present an approach that is based on neural networks. In this approach we first apply a variation of the Cyclic Generalized Lloyd's Algorithm (CGLA) to every point of the training set in order to find the proper codeword for every one of these points. The initial codewords are given by a regression tree approach [13]. We propose to use a neural network for each quantizer in order to represent the training points along with their associated codewords.

The rest of the paper is organized into the following sections. Some notation along with background information is discussed in Section 2. The variation of the CGLA and the neural network approach are presented in Section 3. Implementation issues regarding the calculation of the estimation error are discussed in Section 4. Simulation results are presented in Section 5.

2 Background

In order to attack the problem of quantizer design for a distributed estimation system in the case where only a training set, $\mathcal{T}$ is available, one can use the training set with the CGLA to assign the best codeword to every training point. The CGLA was introduced by Longo *et al* [12]. This algorithm starts with an initial guess of quantizers and fusion center and iteratively improves them by finding the optimal component given the others. It leads to an estimation error that converges and is very sensitive to initialization of the labels (codewords) that correspond to every training point. In the method that we propose we use the system produced by the regression tree approach proposed by Megalooikonomou and Yesha [13, 14] in order to initialize the labels of the training points. This approach involves growing and pruning of regression trees along with labeling techniques, for iteratively decreasing the estimation error.

Let X_1^q and X_2^r be the random observation vectors at the sensors and θ the unobservable continuous quantity that the fusion center tries to estimate. We use the vector notation X_k^p, for the sensor k, as a shorthand for $(X_k[1], X_k[2], \ldots, X_k[p])$. The notation that we use in this paper is summarized in Table 1. Let $\{(X_1^q, X_2^r)^{(t)}, \theta^{(t)}; t = 1, \ldots, M\}$ be the training set, $\mathcal{T}$ of size M that represents the statistics of the source, Q_k the quantizer for the sensor k, and $\hat{X}_k^{p,t}$ the transmitted value for the observation $X_k^{p,t}$ to the fusion center. The task of the fusion center is to estimate the unobserved quantity θ based on the $\hat{X}_k^{p,t}$ it receives.

Let h be the function of the fusion center that gives the estimate of θ and $P_{Q_1} = \{U_i; i = 1, \ldots, N\}$ and $P_{Q_2} = \{V_j; j = 1, \ldots, L\}$, be the partition regions for the quantizers Q_1 and Q_2, respectively, that are produced using the regression tree method. The first quantizer has codewords (labels) $i = 1, \ldots, N$ and the second has codewords $j = 1, \ldots, L$. Let $l(X_k^{p,t})$ be the label of the point $X_k^{p,t}$. The labels, $l(X_1^{q,t})$ and $l(X_2^{r,t})$, produced by the regression tree method are:

$$l(X_1^{q,t}) = \sum_{i=1}^{N} (i-1) I_{U_i}(X_1^{q,t}) \tag{1}$$

$$l(X_2^{r,t}) = \sum_{j=1}^{L} (j-1) I_{V_j}(X_2^{r,t}) \tag{2}$$

where $I_A(x)$ denotes the indicator function of a set $A \subset \Re^d$ of dimension d, i.e. $I_A(x) = 1$

if x is in A and $I_A(x) = 0$ otherwise.

The fusion center h has the following value for each pair of codewords (labels) i, j:

$$h(i,j) = \frac{1}{|\mathcal{R}_{i,j}|} \sum_{t:\left(X_1^q, X_2^r\right)^{(t)} \in \mathcal{R}_{i,j}} \theta^{(t)} \tag{3}$$

where $\mathcal{R}_{i,j}$ is the following subset of the training set:

$$\mathcal{R}_{i,j} = \{(X_1^q, X_2^r)^{(t)} : l(X_1^{q,t}) = i, l(X_2^{r,t}) = j\} \tag{4}$$

The estimation error, can then be expressed as follows:

$$Error = \frac{1}{M} \sum_{t=1}^{M} \left(\theta^{(t)} - h(l(X_1^{q,t}), l(X_2^{r,t}))\right)^2 \tag{5}$$

2.1 Growing and pruning of the regression trees

The regression trees [1] are decision trees with queries of the form $X_k[i] < c_j$ (for an observation variable $X_k[i]$ and a constant c_j) where each leaf R_i is labeled by an estimation value $h(i)$ which is generally constant (see Figure 1). For observations of dimension d the leaves of the regression tree correspond to d-dimensional rectangles. These trees are formed by iteratively splitting subsets of the training set into descendant disjoint subsets. The constraint that is imposed from the separate encoding scheme when building the regression trees is that one tree cannot have different splits based on answers to queries on the other tree. The growing of the trees is based on the decrease of the error in the estimation of the parameter θ and it is cooperative, i.e., we grow one tree taking into account the tree for the other sensor (except from the first tree that we grow).

In order to grow right-sized trees, pruning (by recombining leaves that are siblings) is also involved in the growing procedure. The pruning algorithm that is used is the *recursive optimal pruning algorithm* (ROPA) proposed by Kiang *et al* [9]. Their algorithm is an extension of the Generalized BFOS (GBFOS) algorithm proposed by Chou *et al* [3] that optimally prunes a TSVQ codebook and which is a generalization of an algorithm by Breiman *et al* [1] for classification and regression trees. The purpose of the pruning of the original regression trees in the case of fixed rate quantization is to get a subtree with a given number of leaves and with estimation error that is as small as possible.

2.2 Labeling of the rectangles

After growing the trees, the rectangles (that correspond to leaves of the regression trees) are labeled using an algorithm that is related to the Cyclic Generalized Lloyd Algorithm (CGLA) [12], the s-CGLA, in order to combine the rectangles into the required number of partition regions. Then the trees are grown from the beginning in order to improve over the previous trees including in this procedure the labels that have been assigned to the rectangles.

Labeling is used to denote the assignment of codewords to the rectangles of the regression trees in the sense that rectangles that have the same label form a quantizer partition region and use the same codeword for the transmission. One labeling technique is the s-CGLA that considers together groups of training samples. A second labeling technique is the lh-s-CGLA that changes the fusion center temporarily whenever there is a decision that has to be made in order to calculate the effect of every possible change and also keeps the fusion center table updated all the time.

Let n_k be the number of codewords and $m_k \geq n_k$ be the number of leaves for quantizer k. Let also $l(r)$ be the label of a specific rectangle r. Given the partition regions P_{Q_1}

and P_{Q_2}, for X_1^q and X_2^r respectively, the optimal fusion center h is given by:

$$h(m,n) = \frac{1}{|\mathcal{R}'_{m,n}|} \sum_{t:\left(X_1^q, X_2^r\right)^{(t)} \in \mathcal{R}'_{m,n}} \theta^{(t)} \tag{6}$$

where $\mathcal{R}'_{m,n}$ is the following subset of the training set:

$$\mathcal{R}'_{m,n} = \{(X_1^q, X_2^r)^{(t)} : l(r(X_1^{q,t})) = m,\ l(r(X_2^{r,t})) = n\} \tag{7}$$

The estimation error, err_i, contributed by the following subset, $\mathcal{R}''_i$, of the training set (based on X_1^q)

$$\mathcal{R}''_i = \{(X_1^q, X_2^r)^{(t)} : r(X_1^{q,t}) = i\} \tag{8}$$

is given by:

$$err_i = \frac{1}{|\mathcal{R}''_i|} \sum_{t:\left(X_1^q, X_2^r\right)^{(t)} \in \mathcal{R}''_i} \left(\theta^{(t)} - h(l(i), l(r(X_2^{r,t})))\right)^2 \tag{9}$$

The total estimation error is then given by:

$$Error = \sum_{i:0\ldots m_1-1} err_i \tag{10}$$

The estimation error can also be expressed using a similar formula and the corresponding subsets based on X_2^r.

The main component of lh-s-CGLA performs the following for each sensor k [13] until the reduction on the estimation error given by Equation 10 is less than a given threshold:

```
for (each rectangle $i$ from 0 to $m_k - 1$) {
 for (each label $j$ from 0 to $n_k - 1$) {
   $l(i) \leftarrow j$;
   calculate $h$ using Equation 6;
   calculate estimation error, $err_i[j]$, using Equation 9;
 }
 $l(i) \leftarrow \arg\min_{j:0\ldots(n_k-1)}(err_i[j])$;
 calculate $h$ using Equation 6;
}
```

The breakpoint initialization method is used to initialize the labels of the rectangles. This method initializes the labels by first pruning even more the final pruned subtrees, F_1 and F_2 to the number of labels n_1 and n_2 getting the trees, F_1' and F_2' and then assigning the same label to all the rectangles of the trees F_1 and F_2 that correspond to one rectangle of the trees F_1' and F_2'.

3 The methods

In order to design quantizers for a distributed estimation system in the case where only a training set, $\mathcal{T}$, is available, we use the training set with a variation of the CGLA to assign the best codeword to every point of the training set. Then we use a neural network to represent the training points and their associated codewords for each quantizer.

3.1 A variation of the CGLA

We use a variation of the CGLA on every point of the training set. The CGLA is very sensitive to the initialization of the codewords. One could randomly initialize the codewords of the training points (i.e., with probability $1/n_k$ we use a label in $0, \ldots, (n_k - 1)$). However, this approach leads to an estimation error that converges to a point that is worse than the one that it converges to if the codewords are initialized using the previously described regression tree approach. The output of the regression tree approach is a collection of rectangles along with their associated labels (codewords) such that rectangles that have the same label form a quantizer partition.

We initialize the labels of the training points with the quantities $l(X_k^{b,t})$ for quantizer k. Then we use the lh-CGLA in order to decide about the best label for every one of these points. This algorithm does not consider groups of training points as the lh-s-CGLA. It considers individual points. Let the index t go through all the training points and the index j go through all the possible labels. Let also n_k be the number of codewords and m_k be the number of rectangles of the quantizer k. The lh-CGLA algorithm performs the following for each sensor k:

lh-improve labels

```
1. t ←1.
2. j ←0.
3. $l(X_k^{b,t}) \leftarrow j$.
4. calculate $h$ using Equation 3 and the estimation error, error[j], using Equation 5.
5. $j \leftarrow j+1$, if $j < n_k$ go to step 3.
6. $l(X_k^{b,t}) \leftarrow \arg\min_{j:0\ldots(n_k-1)}(error[j])$, calculate $h$ using Equation 3.
7. $t \leftarrow t+1$, if $t <= M$ go to step 2 else stop.
```

The above procedure is repeated until the reduction on the estimation error becomes less than a given threshold. In order to decide about the best label for a point this lookahead algorithm changes temporarily the fusion center in order to calculate the effect of every possible change. Moreover, it also keeps the fusion center table updated all the time.

3.2 Neural network quantizers

The neural network that we use is a two-layer feed-forward network and the learning rule is backpropagation with momentum and adaptive learning rate. The momentum method decreases the probability that the network will get stuck in a shallow minimum in the error surface and helps decrease training times. The adaptive learning rate decreases training time by keeping the learning rate reasonably high while insuring stability. For the first layer we use a hyperbolic tangent transfer function and for the second layer we use a linear transfer function. This kind of networks has been proven capable of approximating any function with a finite number of discontinuities with arbitrary accuracy.

The quantizer for each sensor $k = 1, 2$ is a neural network, NN_k. The system that we propose for the case of 2 codewords for each quantizer is depicted in Figure 2. The only input of the neural network at the sensor k is its observation, X_k^b. The output of the neural network is the associated label $l'(X_k^b)$ for this observation, where $l'(X_k^b)$ is the final label of the point X_k^b after the application of lh-CGLA. We use the unary representation for the outputs of the neural network, so the number of outputs for NN_k is n_k where n_k is the number of codewords for quantizer k. Let S_1 be the number of neurons of the first layer of the neural network. We use n_k neurons for the second layer (the output layer). The weight and bias matrices for the two layers are the parameters of the neural network. The dimensions for the weight and bias matrices are $S_1 \times 1$, $S_1 \times 1$, $n_k \times S_1$, and $1 \times n_k$ for the first and the second layer respectively. The number of parameters, P, used for the description of the two-layer neural network is

$$S_1(n_k + 2) + n_k. \tag{11}$$

For the training of the neural network NN_k for quantizer k we use the following pair of input-output for every training point t

$$(X_k^{b,t}, u(l'(X_k^{b,t}))) \tag{12}$$

where $u(x)$ is the unary representation of x.

Let $f(X_k^{b,t})$ be the output vector of the neural network for input $X_k^{b,t}$ after the training. This output vector may not be in unary form so we select the *max* of its elements and we report this as the codeword for quantizer k (this is performed by the Codeword Selector module in Figure 2). We use the same notation $u(.)$ for the unary transformation of the output vector. The transmitted value from the sensor k to the fusion center is

$$\hat{X}_k^{b,t} = u(f(X_k^{b,t})). \tag{13}$$

The fusion center table h that we use is the one that was produced using the regression tree approach. The estimation error is then expressed as

$$Error = \frac{1}{M}\sum_{t=1}^{M}\Big(\theta^{(t)} - h(u(f(X_1^{q,t})), u(f(X_2^{r,t})))\Big)^2. \tag{14}$$

4 Implementation issues

In order to find the best label for every point of the training set we use the lh-CGLA algorithm described in Section 3.1. Here we discuss important implementation issues related to the calculation of the estimation error that is performed repeatedly in lh-CGLA.

Recalling that $l(X_1^q)$ and $l(X_2^r)$ are the codewords for the observations X_1^q and X_2^r respectively before the use of the neural network, the estimation of the parameter θ at the fusion center can be expressed as follows:

$$\hat{\theta}^{(t)} = h(l(X_1^{q,t}), l(X_2^{r,t})) \tag{15}$$

for a given point t of the training set. The fusion center h is given by Eq. 3. The estimation error that we try to minimize by improving the labeling scheme using the lh-CGLA is:

$$Error = \frac{1}{M}\sum_{t=1}^{M}\Big(\theta^{(t)} - \hat{\theta}^{(t)}\Big)^2. \tag{16}$$

The fusion center table h can be computed in $O(M)$ steps, where M is the number of training points. We assume that a table of size M that gives the label associated with every training point is available. Such a table can be constructed in $O(1)$ steps. The estimation error can then be computed in $O(M)$ steps.

Given h and the labels of the points for X_2^r, $l(X_2^r)$, the best label for the point $X_1^{q,j}$ is u if it satisfies:

$$\{Error|l(X_1^{q,j}) = u, l(X_2^r)\} \leq \{Error|l(X_1^{q,j}) = v, l(X_2^r)\} \quad \forall v \neq u. \tag{17}$$

The naive way to find the best label for a given point is to change temporarily the label of that point considering all the possible labels, recalculate the fusion center table h and the estimation error for each one of them and choose the one that gives the smaller estimation error. This procedure requires $O(n_k M^2)$ steps for one pass of quantizer k. However, the objective is to measure the goodness of a label for a certain point. Also the change of the label of a single point affects only a small part of table h and a small part of the calculation of the estimation error. The estimation error can also be written as follows:

$$Error = \frac{1}{M}\left(\sum_{t=1}^{M}\theta^{(t)2} + \sum_{t=1}^{M}\hat{\theta}^{(t)2} - 2\sum_{t=1}^{M}\theta^{(t)}\hat{\theta}^{(t)}\right). \tag{18}$$

The first term is a constant quantity for the experiment and can be calculated once in $O(M)$ steps. The second term can be further expressed as:

$$\sum_{t=1}^{M}\hat{\theta}^{(t)2} = \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} obs(i,j)h(i,j)^2 \tag{19}$$

where $obs(i,j)$ is the number of observations of the (i,j)-th entry of the fusion center table.

With the change of label of a single point for a quantizer, only two entries of the fusion center table are changed. If the sum of $\theta^{(t)}$ for each subset $\mathcal{R}_{i,j}$

$$sum\_\theta(i,j) = \sum_{t:\left(X_1^q,X_2^r\right)^{(t)}\in\mathcal{R}_{i,j}} \theta^{(t)} \tag{20}$$

of the training set is known then the new second term of Eq. 18 that considers the change of a label can be calculated in $O(1)$. A similar argument is used to show that the last term of Eq. 18 can also be updated in order to take into account the change of a label of a single point for a quantizer in $O(1)$ steps. Here we assume that the previous value of this term is also available. The update of the global variables that we use also takes constant time. The new estimation error after one change of label of a certain point for a quantizer can now be calculated in $O(1)$. One pass of quantizer k has now been reduced to $O(n_k M)$ steps.

5 Simulation results and discussion

In the simulations we consider the case where the observations at the quantizers are scalar quantities of the form

$$x_k = \theta + n_k, \ k = 1, 2 \tag{21}$$

where the noises n_k at the sensors are Gaussian distributed with correlation coefficient ρ and marginal distributions $N(0, \sigma_n^2)$, where σ_n^2 is the variance of the noises. The parameter θ has Gaussian distribution $N(0, 1)$ and is independent of the noises $n_k, \ k = 1, 2$. The quantizers are designed using a training set $\mathcal{T}$ of $10,000$ samples and are tested on a test set $\mathcal{T}'$ of $10,000$ samples that is independent of $\mathcal{T}$ although it is constructed the same way as $\mathcal{T}$ is. We report results on the test set $\mathcal{T}'$ unless otherwise stated. We use the breakpoint initialization of labels in the regression tree approach. The value 0.005 was used for the error threshold in all experiments. The number of epochs that were used to train the neural networks is $10,000$.

Prior to comparing the performance of regression tree and neural network quantizers we run experiments to examine the effect of increasing the number of leaves while keeping the number of labels constant. Figure 3 shows that initially performance improves as the number of leaves increases until we reach a certain number of leaves for each regression tree quantizer. From that point, and due to overtraining, quantizers with more leaves start behaving worse than those with less leaves (recall that we measure performance on $\mathcal{T}'$ and not on $\mathcal{T}$). Another important observation is that as the number of leaves and/or ρ and σ_n^2 increase, the quantizers become non-breakpoint although initially all of them started as breakpoint (see Table 2).

In Table 3 we compare the performance of the regression tree approach with 8, 32, 64, and 128 leaves with that of the neural network. We use 2 codewords for each quantizer. With 8 leaves and 2 codewords the maximum number of parameters for each quantizer is 15 (7 parameters to describe the split points and 8 parameters for the corresponding codewords). However, due to grouping of consecutive intervals that have the same label to a single interval, the actual average number of parameters used for both quantizers is 7. Increasing the number of leaves increases the average number of parameters used by the regression tree approach. The neural network approach uses 5 (=3+2) neurons which is 14 parameters for each quantizer according to Equation 11. The initialization of labels prior to lh-CGLA uses the regression tree approach with 8 leaves and 2 codewords for each quantizer. We present results for several values of σ_n^2 and for $\rho = 0.85$. The average number of parameters used for both quantizers in each case is also presented. The lowest value of the prediction error in each row of every table is highlighted.

The neural network is better than the regression tree approach with 8, 64, and 128 leaves for all values of the noise variance that we tested. For 32 leaves the neural network is better than the regression tree approach for 11 out of the 13 noise variance values and slightly worse, i.e., 0.2189 vs 0.2186 and 0.3368 vs 0.3357, for the other 2 values.

Summarizing, the neural network approach improves over the regression trees for almost every value of the noise variance that we tested. For the very small number of noise variances for which the neural network is not the best, its performance is very close to the best performance.

Acknowledgments

We thank the anonymous referees for very helpful comments.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth Inc., Belmont, CA, USA, 1984.

[2] L. R. Carley, G. R. Ganger, and D. F. Nagle. MEMS-Based Integrated-Circuit Mass-Storage Systems. *Communications of the ACM*, 43(11):73–80, Nov. 2000.

[3] P. A. Chou, T. Lookabaugh, and R. M. Gray. Optimal Pruning with Applications to Tree-Structured Source Coding and Modeling. *IEEE Transactions on Information Theory*, IT-35(2):299–315, Mar. 1989.

[4] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), Salt Lake City, Utah*, May 2001.

[5] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM), Seattle, Washington*, pages 263–270, Aug. 1999.

[6] J. A. Gubner. Distributed Estimation and Quantization. *IEEE Transactions on Information Theory*, IT-39(4):1456–1459, Jul. 1993.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM), Boston, Massachusetts*, pages 56–67, Aug. 2000.

[8] J. Kahn, R. Katz, and K. Pister. Emerging Challenges: Mobile Networking for Smart Dust. *Journal of Communications and Networks*, 2(3):188–196, Sept. 2000.

[9] S.-Z. Kiang, R. L. Baker, G. J. Sullivan, and C.-Y. Chiu. Recursive Optimal Pruning with Applications to Tree Structured Vector Quantizers. *IEEE Transactions on Image Processing*, 1(2):162–169, Apr. 1992.

[10] W.-M. Lam and A. R. Reibman. Quantizer Design for Decentralized Estimation Systems with Communications Constraints. In *Proceedings of the 23rd Annual Conference on Information Sciences and Systems*, pages 489–494, Baltimore, MD, USA, Mar. 1989.

[11] W.-M. Lam and A. R. Reibman. Design of Quantizers for Decentralized Estimation Systems. *IEEE Transactions on Communications*, 41(11):1602–1605, Nov. 1993.

[12] M. Longo, T. D. Lookabaugh, and R. M. Gray. Quantization for Decentralized Hypothesis Testing under Communication Constraints. *IEEE Transactions on Information Theory*, IT-36(2):241–255, Mar. 1990.

[13] V. Megalooikonomou and Y. Yesha. Quantization for Distributed Estimation with Unknown Observation Statistics. In *Proceedings of the 31th Annual Conference on Information Sciences and Systems*, Baltimore, MD, USA, Mar. 1997.

[14] V. Megalooikonomou and Y. Yesha. Quantizer Design for Distributed Estimation with Communication Constraints and Unknown Observation Statistics. *IEEE Transactions on Communications*, 48(2):181–184, 2000.

[15] B. Warneke, M. Last, B. Liebowitz, and K. Pister. Smart Dust: Communicating with a Cubic-Millimeter Computer. *IEEE Computer Magazine*, pages 44–51, Jan. 2001.

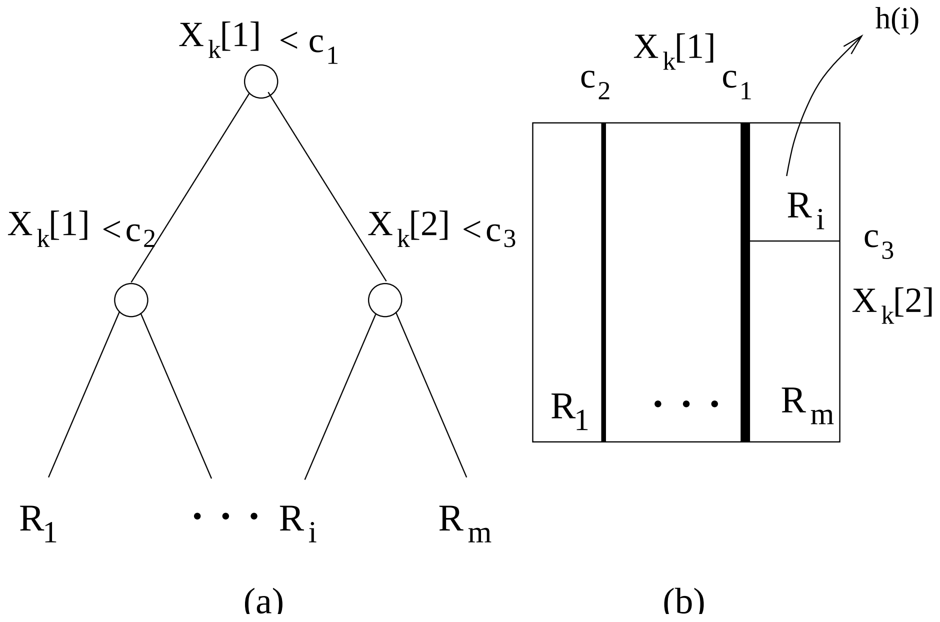

Figure 1: A regression tree of two variables (a) and its partitions (b).

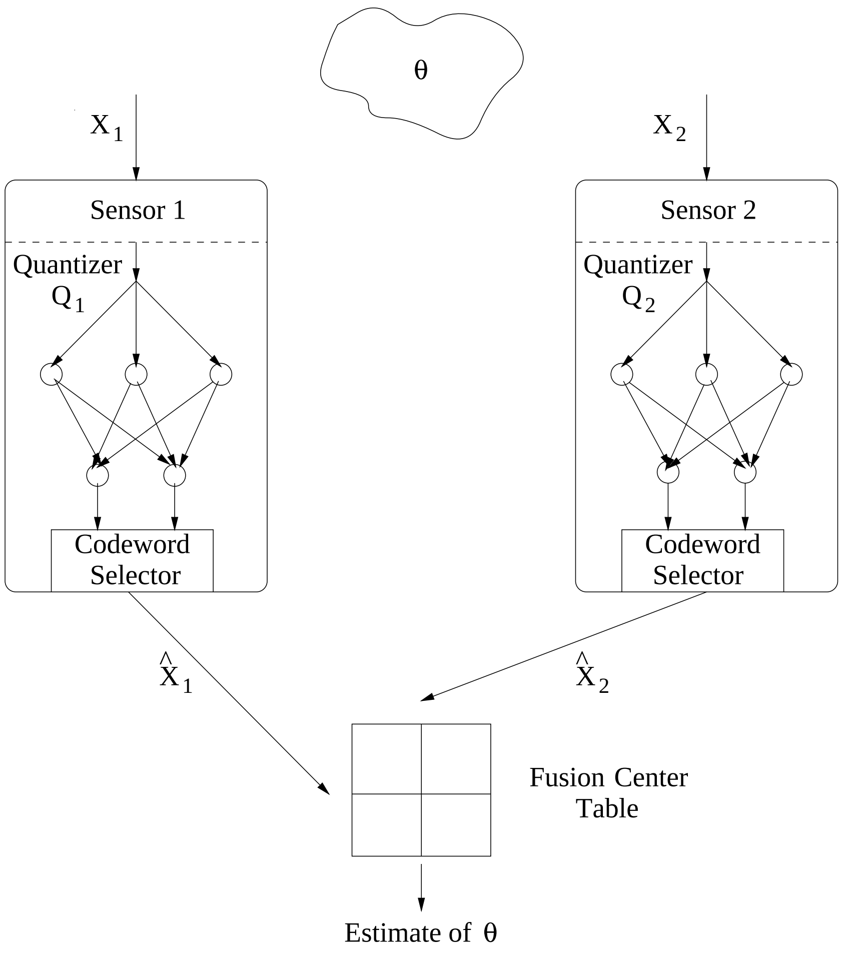

Figure 2: A neural network quantizer for each sensor of the distributed estimation system (2 codewords for each quantizer).

Symbol	*Definition*
Q_k	quantizer for sensor k
P_{Q_1}, P_{Q_2}	partition regions for quantizers Q_1, Q_2 $= \{U_i; i = 1, \ldots, N\}$, $\{V_j; j = 1, \ldots, L\}$, respectively
n_k, m_k	number of codewords, number of leaves for Q_k
NN_k	neural network quantizer for sensor k
h	fusion center function
θ, $\hat{\theta}$	unobservable quantity the system tries to estimate and its estimate
X_k^p	random observation vector at sensor k
$\hat{X}_k^p$	transmitted value for observation X_k^p
$\mathcal{T}$	the training set, $\{(X_1^q, X_2^r)^{(t)}, \theta^{(t)}; t = 1, \ldots, M\}$
$\mathcal{R}_{i,j}$, $\mathcal{R}'_{m,n}$, $\mathcal{R}''_i$	various subsets of the training set $\mathcal{T}$
$l(X_k^{p,t})$	label of the point $X_k^{p,t}$
$l(r)$	label of points in rectangle r
$obs(i,j)$	number of observations of the (i,j)-th entry of the fusion center table
$Error$	error in estimating θ
$u(x)$	unary representation of a number x

Table 1: Symbol Table.

$\backslash\ \rho$	4 leaves			8 leaves			16 leaves			32 leaves		
$\sigma_n^2\backslash$	0	0.5	0.85	0	0.5	0.85	0	0.5	0.85	0	0.5	0.85
0.005	-	-	-	+	+	+	+	+	+	+	+	+
0.050	-	-	-	+	+	+	+	+	+	+	+	+
0.100	-	-	-	+	+	+	+	+	+	+	+	+
0.150	-	-	-	+	+	+	+	+	+	+	+	+
0.200	-	-	-	+	+	+	+	+	+	+	+	+
0.300	-	-	-	-	+	+	+	+	+	+	+	+
0.400	-	-	-	-	+	+	-	+	+	+	+	+
0.500	-	-	-	-	+	+	-	+	+	+	+	+
0.600	-	-	-	-	-	+	-	+	+	+	+	+
0.700	-	-	-	-	-	+	-	+	+	+	+	+
0.800	-	-	-	-	-	+	-	+	+	+	+	+
0.900	-	-	-	-	-	+	-	+	+	+	+	+
1.000	-	-	-	-	-	+	-	+	+	+	+	+

Table 2: Effect of the number of leaves to quantizers being breakpoint (-) or non-breakpoint (+) for several values of σ_n^2 and for $\rho = 0, 0.5$, and 0.85. [The quantizers were initialized as breakpoint and 4 labels were used for each one].

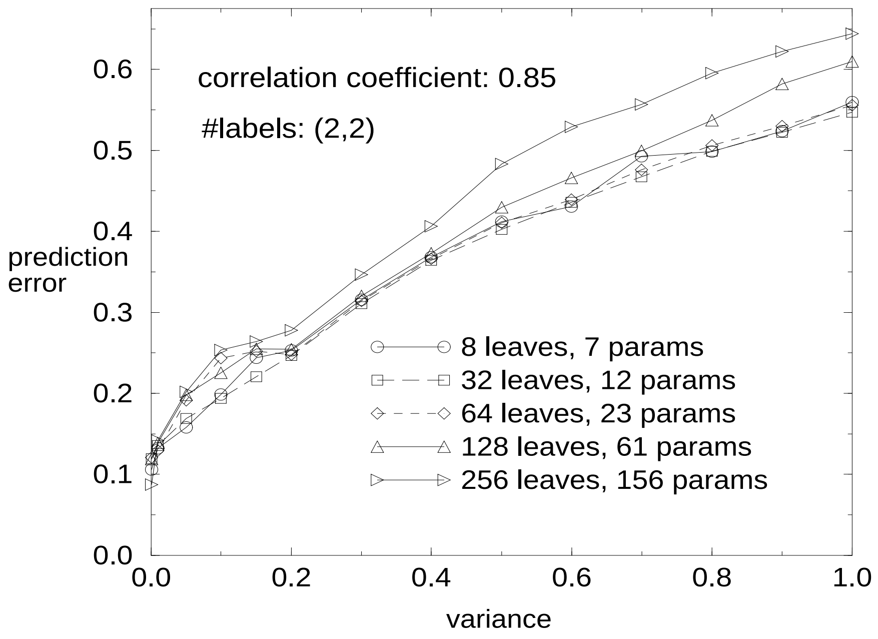

Figure 3: Effect of the number of leaves on the performance of the quantizers.

σ_n^2	bp_init, $(2,2)$ labels, $\rho = 0.85$				
	regr.tree 8 leaves 7 params	regr.tree 32 leaves 12 params	regr.tree 64 leaves 23 params	regr.tree 128 leaves 61 params	NN 5 neurons 28 params
0.001	0.1311	0.1442	0.1455	0.1436	**0.1209**
0.050	0.1831	0.1941	0.2165	0.2230	**0.1715**
0.100	0.2233	**0.2186**	0.2686	0.2504	0.2189
0.150	0.2691	0.2457	0.2771	0.2799	**0.2409**
0.200	0.2782	0.2717	0.2729	0.2795	**0.2698**
0.300	0.3405	**0.3357**	0.3392	0.3455	0.3368
0.400	0.3929	0.3894	0.3913	0.3977	**0.3835**
0.500	0.4368	0.4274	0.4355	0.4547	**0.4256**
0.600	0.4558	0.4607	0.4644	0.4910	**0.4558**
0.700	0.5179	0.4926	0.5009	0.5245	**0.4863**
0.800	0.5233	0.5237	0.5312	0.5621	**0.5213**
0.900	0.5486	0.5473	0.5548	0.6069	**0.5331**
1.000	0.5844	0.5723	0.5807	0.6348	**0.5670**

Table 3: Performance comparison of the neural network approach and the regression tree approach.