DESIGN OF NEURAL NETWORK QUANTIZERS FOR A DISTRIBUTED ESTIMATION SYSTEM WITH COMMUNICATION CONSTRAINTS

Vasileios Megalooikonomou and Yaacov Yesha

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

ABSTRACT

We consider the problem of quantizer design in a distributed estimation system with communication constraints at the channels in the case where the observation statistics are unknown and one must rely on a training set. The method that we propose 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 along with their associated codewords. The codeword of every training point is initialized using a regression tree approach. Simulations show that the combined approach i.e. building the regression tree system and using its quantizers to initialize the neural networks provides an improvement over the regression tree approach except in the case of high noise variance.

1. INTRODUCTION

The model of a distributed estimation system that we consider consists of a single fusion center with 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 parameters based on observations collected by remote sensors and transmitted to the center. Some 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. Although the number of sensors can be in general arbitrary, here we consider the two-sensor case since the same solution can be easily extended to the more general case. The observations at the sensors are random. Here, we consider the case where the joint probability density function is unknown.

A solution based on a gereralization 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 [4]. The same problem in the case of known probability model was considered by Lam and Reibman [2, 3]. Gubner [1] 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.

Here, 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 CGLA on every point of the training set in order to find the proper codeword for every one of them. The initial codewords are the ones that are given by the regression tree approach [4]. We then use a neural network for each quantizer in order to represent the training points along with their associated codewords.

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, someone can use the training set with the CGLA in order to assign the best codeword to every training point.

The CGLA leads to an estimation error that converges and it 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 [4] in order to initialize the labels of the training points. This approach involves growing and pruning of regression trees along with some 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 $(X_k[1], X_k[2], \ldots, X_k[p])$. 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 we produce 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:

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

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

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:

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

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

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

The estimation error, can then be expressed as follows:

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

2.1. Growing and pruning of the regression trees

The regression trees 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 is labeled by an estimation value which is generally constant. For observations of dimension d the leaves of the regression tree correspond to d-dimensional rectangles.

The regression trees are formed by iteratively spliting 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 size trees, pruning (by recombining leaves that are siblings) is also involved in the growing procedure.

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), 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 desicion 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.

3. THE METHODS

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 we use the training set with a variation of the CGLA in order 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. We initialize the labels of the training points with the quantities $l(X_k^{p,t})$ for quantizer k. The quantizers of the regression tree approach as was described earlier can be viewed as a collection of rectangles provided by a recursive partitioning procedure along with their associated labels (codewords) such that rectangles that have the same label to form a quantizer partition. 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 [4]. 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 $\leftarrow$1.

2. j $\leftarrow$0.

3. $l(X_k^{p,t}) \leftarrow j$.

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. $l(X_k^{p,t}) \leftarrow \arg\min_{j:0\ldots(n_k-1)}(error[j])$, 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.

This method requires the whole training set to be stored in each one of the sensors. Someone could group neighboring points that have the same label after the application of the lh-CGLA in the same rectangle. However, there is no guarantee on the number of rectangles (intervals in the 1-dimensional case) produced by the algorithm. The worst case is when the number of rectangles is equal to the number of training points. So there is no easy way to describe the regions except for storing the entire training set along with the associated codewords. In the next section we overcome these problems by proposing the use of a neural network for each one of the quantizers of the distributed estimation system.

3.2. Neural network quantizers

We use a neural network to represent the training points and their associated codewords for each quantizer. We reduce the space required for storing all the training points along with their associated codewords. The neural network also provides the required smoothing that otherwise has to be done using a convolution of the training set with a smoothing kernel or using a k-nearest neighbor algorithm.

The neural network that we use is a two-layer feedforward network and the learning rule is the backpropagation with momentum and adaptive learning rate. For the first layer we use a hyperbolic tangent transfer fuction and for the second layer we use a linear transfer function. This kind of networks has been proven capable of approximating any function with finite number of discontinuities with arbitrary accuracy.

The quantizer for each sensor $k = 1, 2$ is a neural network, *NN*$_k$. The only input of the neural network at the sensor k is its observation, X_k^p. The output of the neural network is the associated label $l'(X_k^p)$ for this observation, where $l'(X_k^p)$ is the final label of the point X_k^p 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). We use the notation $c_k : k = 1, \ldots, M$ for the elements of the weight and bias matrices, i.e., the parameters of the neural network. The number of parameters used for the description of the two-layer neural network is

$$S_1(n_k + 2) + n_k. \tag{6}$$

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:

$$(X_k^{p,t}, u(l'(X_k^{p,t}))) \tag{7}$$

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

Let $f(X_k^{p,t})$ be the output vector of the neural network for input $X_k^{p,t}$ 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. 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:

$$\hat{X}_k^{p,t} = u(f(X_k^{p,t})) \tag{8}$$

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:

$$Error = \frac{1}{M}\sum_{t=1}^{M}\left(\theta^{(t)} - h(u(f(X_1^{q,t})), u(f(X_2^{r,t})))\right)^2 \tag{9}$$

4. SIMULATION RESULTS AND DISCUSSION

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

$$x_k = \theta + n_k,\ k = 1, 2 \tag{10}$$

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 $5,000$ samples and are tested on a test set $\mathcal{T}'$ of $5,000$ samples that is independent of $\mathcal{T}$ although it is constructed the same way as $\mathcal{T}$. 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 $10,000$. Here, we compare the performance of the regression tree approach, the neural network approach and the full training set approach where all the training points along with their associated labels have to be stored. We use 2 codewords for each quantizer.

For the regression tree approach we report the case where 8 and 32 leaves are used. 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 each quantizer is 4. Increasing the number of leaves to 32 increases the average number of parameters for the regression tree approach to 5. We have to mention here that we did not observe any significant improvement by increasing the number of leaves beyond 32 in this case.

The neural network approach uses $5(=\ 3\ +\ 2)$ neurons which is 14 parameters for each quantizer according to Equation 6. 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 $\rho = 0.85$.

In Table 1 we compare the performance of the regression tree approach with that of the neural network. The second method is superior for low noise variance. This table contains a column for the performance of the system after the lh-CGLA is applied. Notice that although the performance of lh-CGLA on the test set is poor, the final performance of the neural network approach on the test set is better than that of the regression tree approach. Keep in mind though that the regression tree approach was used to give the initial assignment of labels before the lh-CGLA is applied. Although there is an improvement in the regression tree approach if more leaves are used we noticed that there is also a corresponding additional improvement in the neural network approach, if during the initilialization, the output from the regression tree approach with more leaves, is used.

From the third column of Table 1 it is also apparent that the approach that stores all the training points along with their associated labels behaves worse than the neural network approach that uses much less parameters. However, the algorithm that was used to calculate the error on the test set for the full training set approach was the nearest neighbor so some reduction in the estimation error is expected if a method of convolution of the training set with a suitable smoothing kernel, or a k-nearest neighbor algorithm with the proper k value, is used. Storage and time limitations on the sensors may suggest that this approach that stores the full training set is not feasible.

σ_n^2	bp_init, $(2, 2)$ labels, $\rho = 0.85$			
	regr.tree 8 leaves	regr.tree 32 leaves	lh-CGLA	NN
0.001	0.1230	0.1224	0.1250	0.1219
0.005	0.1253	0.1253	0.1298	0.1247
0.010	0.1616	0.1616	0.1372	0.1289
0.050	0.1950	0.1685	0.2081	0.1652
0.100	0.2221	0.2747	0.2928	0.2038
0.150	0.2677	0.2478	0.3682	0.2419
0.200	0.2811	0.2769	0.4214	0.2733
0.300	0.3663	0.3354	0.5787	0.3347
0.400	0.3913	0.3781	0.6704	0.3768
0.500	0.4411	0.4352	0.7491	0.4289
0.600	0.4713	0.4713	0.8379	0.4803
0.700	0.5244	0.4886	0.8930	0.5166
0.800	0.5299	0.5276	0.9797	0.5391
0.900	0.5619	0.5628	1.0563	0.5756
1.000	0.5877	0.5806	1.0944	0.5832

Table 1: Performance comparison of the neural network approach (5 neurons) and the regression tree approach.

5. REFERENCES

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

[2] 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, Baltimore, Maryland*, pages 489–494, Mar. 1989.

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

[4] 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, Maryland*, pages 138–143, Mar. 1997.