entropy

MDPI

Article

Distributed Vector Quantization Based on Kullback-Leibler Divergence

Pengcheng Shen, Chunguang Li * and Yiliang Luo

Received: 22 June 2015; Accepted: 23 November 2015; Published: 30 November 2015
Academic Editor: Raúl Alcaraz Martínez

College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China; equipment777@zju.edu.cn (P.S.); luoyiliang1989@163.com (Y.L.)
* Correspondence: cgli@zju.edu.cn; Tel./Fax: +86-571-87951577

Abstract: The goal of vector quantization is to use a few reproduction vectors to represent original vectors/data while maintaining the necessary fidelity of the data. Distributed signal processing has received much attention in recent years, since in many applications data are dispersedly collected/stored in distributed nodes over networks, but centralizing all these data to one processing center is sometimes impractical. In this paper, we develop a distributed vector quantization (VQ) algorithm based on Kullback-Leibler (K-L) divergence. We start from the centralized case and propose to minimize the K-L divergence between the distribution of global original data and the distribution of global reproduction vectors, and then obtain an online iterative solution to this optimization problem based on the Robbins-Monro stochastic approximation. Afterwards, we extend the solution to apply to distributed cases by introducing diffusion cooperation among nodes. Numerical simulations show that the performances of the distributed K-L–based VQ algorithm are very close to the corresponding centralized algorithm. Besides, both the centralized and distributed K-L–based VQ show more robustness to outliers than the (centralized) Linde-Buzo-Gray (LBG) algorithm and the (centralized) self-organization map (SOM) algorithm.

Keywords: distributed signal processing; Kullback-Leibler divergence; sensor network; vector quantization

1. Introduction

Vector quantization is a signal processing method which uses reproduction vectors to represent original data vectors while maintaining necessary fidelity of the data [1,2]. As one of the data compression methods which are able to reduce communication and storage burdens, vector quantization has been intensively studied in recent years. A vector quantizer is a system that maps original/input vectors into corresponding reproduction vectors drawn from a finite reproduction alphabet. Many famous vector quantization algorithms have been proposed. The Linde-Buzo-Gray (LBG) algorithm [3,4] and the self-organization map (SOM) algorithm [5–7] are two of the most popular vector quantization algorithms. Based on information theoretic concepts, vector quantization algorithms which aim to minimize the Cauchy-Schwartz (C-S) divergence or the Kullback-Leibler (K-L) divergence between the distributions of the original data and the reproduction vectors have also been devised and have been proven to perform better than the LBG and SOM algorithms [8,9].

In signal processing over networks, data are usually collected/stored at different nodes over networks and data from all nodes are needed in tasks to make use of the overall information. In traditional signal processing algorithms, we need to transmit all data to one powerful processing center to perform signal processing tasks, which is sometimes infeasible for distributed applications, especially when the data amounts are very large. The reasons are nodes over networks are

usually communication-resource-limited and sometimes power-limited (in the case of wireless sensor networks), and transmitting massive original data will consume large power/communication resources and also bring a data-privacy-leaking risk. Furthermore, when the data amount is extremely large, the center node may not be able to efficiently process the whole data. Therefore, distributed signal processing algorithms, which take these limitations into consideration, are needed in such cases. Many distributed algorithms have been proposed in recent years [10], such as the distributed parameter estimation [11–20], distributed Kalman filtering [21,22], distributed detection [23,24], distributed clustering [25,26], and distributed information-theoretic learning [27,28]. In a majority of these distributed algorithms, signal processing tasks are accomplished at each node based on local computation, local data, as well as limited information exchange among neighbor nodes. During the processing, nodes only transmit necessary information to their neighbors instead of transmitting all original data to one processing center, so as to reduce the communication complexity, protect data privacy, and provide better flexibility and robustness to node/link failures in the meantime. Figure 1 gives a brief sketch of the discussed distributed processing mechanism.

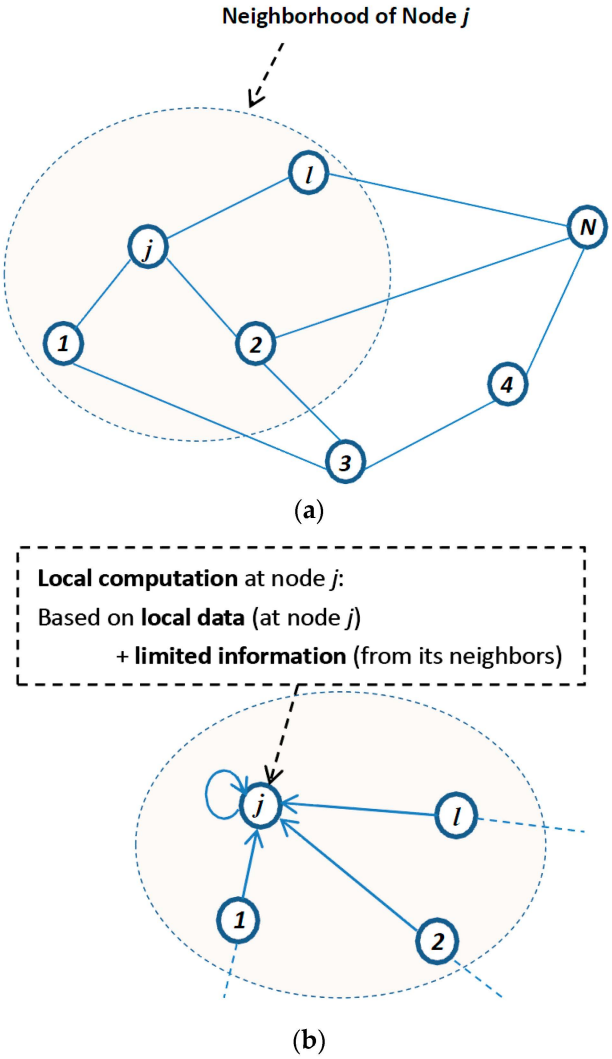

Figure 1. A brief sketch of the distributed processing mechanism: (**a**) Network structure and neighborhood of node *j*; (**b**) Local computation at node *j*.

In terms of distributed vector quantization, the LBG and SOM algorithms have been successfully extended to the distributed case in [29] and the proposed distributed LBG and SOM algorithms

achieve performances close to that of the corresponding centralized LBG and SOM algorithms, respectively. Since the simulation results in literature on centralized vector quantization have shown that algorithms based on C-S divergence and K-L divergence can achieve better performances than the LBG and SOM algorithms [8,9], it is a natural thought to develop divergence-based vector quantization algorithms in the field of distributed processing. However, the existing divergence-based vector quantization algorithms [8,9] cannot be directly/easily extended to the distributed case due to the lack of data samples in estimating the global data distribution for each individual node (details are provided in the following section).

In this paper, we develop a distributed divergence-based vector quantization algorithm that can solve a global vector quantization problem without transmitting original data among nodes. We firstly start from the centralized case. Considering the limitations in distributed cases, we define the objective function based on the K-L divergence between the distribution of global original data and the distribution of global reproduction vectors, and then use the Robbins-Monro (R-M) stochastic approximation method [30,31] to efficiently solve the divergence-minimizing problem online. We show that the obtained iterative solution for the centralized case can be easily extended to distributed cases by introducing diffusion cooperation among nodes, which is a frequently-used technique in distributed processing [11,12]. Under the diffusion cooperation, each node cooperatively estimates the reproduction vectors with its neighbors by exchanging some intermediate estimates rather than transmitting original data. Simulations show that the local estimates obtained at different nodes are quite consistent. Besides, the performances of the distributed algorithm are very close to the corresponding centralized algorithm.

2. Distributed Vector Quantization Algorithm Using K-L Divergence

2.1. Starting from the Centralized Case

Mathematically, a vector quantizer is a mapping, q, from a K-dimensional input vector, $\mathbf{x}(n) = (x_0(n), \cdots, x_{K-1}(n))$, $n = 1, \cdots N$, to a reproduction vector $q(\mathbf{x}(n))$, and the reproduction alphabet contains M reproduction vectors, $\{\mathbf{m}_i, i = 1, \cdots, M\}$. The most important issue in the quantization is how to keep the fidelity of data as much as possible with a limited number of reproduction vectors. The information theory provides natural measures for evaluating the fidelity of data in quantization. Divergences can be used to measure the match degree between the distribution of original data $p(\mathbf{x})$ and the distribution of reproduction vectors. Low values of divergences indicate that the original data can be well represented by the reproduction vectors. Compared with the traditional fidelity measure, sum/mean of squared distortion error, the divergences go beyond the second-order moment and evaluate the data fidelity from a more holistic perspective of the whole distributions. It is expected that divergence-based vector quantization has some advantages over the squared distortion error–based quantization when the distribution of the distortion error is not Gaussian.

There are various kinds of divergences [32], and each has unique characteristics. In [8], the authors have studied the *centralized* vector quantization based on *C-S divergence*. In [9], the authors have studied the *centralized* vector quantization based on *K-L divergence*. In their method, the Parzen window method is employed to estimate the distribution of the original data $p(\mathbf{x})$ based on all the data samples, as well as the distribution of the reproduction vectors. However, in distributed cases, input data samples are distributed over the whole network and are usually impractical to be gathered together. Thus, their method may not be easily extended to the distributed field.

In this paper, *taking the limitations of distributed cases into consideration*, we propose to perform the vector quantization by minimizing the K-L divergence between the distribution of the original data

$p(\mathbf{x})$ and the distribution of the reproduction vectors. Following the thought of [8], we use the Parzen window method to estimate the distribution of reproduction vectors,

$$g(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M}\kappa_{\Theta}(\mathbf{x}-\mathbf{m}_i) \tag{1}$$

where $\kappa_{\Theta}(\cdot)$ is a kernel with parameters Θ (the choices of kernel will be discussed in the following). Given the above estimator, the objective function is written as follows:

$$D = \int p(\mathbf{x})\log\frac{p(\mathbf{x})}{g(\mathbf{x})}d\mathbf{x} = -\int p(\mathbf{x})\log\frac{g(\mathbf{x})}{p(\mathbf{x})}d\mathbf{x} \tag{2}$$

Note that the above divergence is a function of the reproduction vectors, $\{\mathbf{m}_i, i = 1, \cdots M\}$, thus the problem becomes the choice of locations for the reproduction vectors in the original data domain. Naturally, we can minimize D with respect to $\mathbf{m}_i$ as below:

$$\frac{\partial D}{\partial \mathbf{m}_i} = -\int\left[\frac{1}{g(\mathbf{x})}\frac{\partial g(\mathbf{x})}{\partial \mathbf{m}_i}\right]p(\mathbf{x})\,d\mathbf{x} \equiv 0 \tag{3}$$

The above equation depends on the global original data distribution $p(\mathbf{x})$, which is unknown in advance, and more importantly, is hard to estimate in distributed cases. Fortunately, as seen from Formula (3), with the use of K-L divergence, the partial differential formula is a mathematical expectation over $p(\mathbf{x})$. For such a situation, we can employ the Robbins-Monro stochastic approximation method [30,31] to solve the above equation effectively. The R-M method is an online iterative algorithm which directly solves the kind of equation above by using one data sample per iteration without estimating the total distribution of data. Thus, it avoids the difficulty of estimating the global data distribution, especially in the distributed cases. *This is the reason that we use K-L divergence rather than C-S divergence* [8] to design our objective function in this paper. The iterative solution to our problem given by the R-M method is simple, as below:

$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) + \alpha(n)\left[\frac{1}{g(\mathbf{x}(n))}\frac{\partial g(\mathbf{x}(n))}{\partial \mathbf{m}_i}\right]\Big|_{\mathbf{m}_i(n)} \tag{4}$$

where $\alpha(n)$ is the learning step-size. In this paper we use a monotonically decreasing step-size, $\alpha(n) = \alpha_2\exp(-n/\alpha_1)$, where α_1, α_2 are adjustable parameters, respectively.

Remark 1 (the choices of kernel): The most commonly-used kernel $\kappa_{\Theta}(\cdot)$ in the Parzen window estimation method is the Gaussian kernel with parameters $\Theta_i = \{\mathbf{m}_i, \Sigma_i\}$,

$$G_{\Theta_i}(\mathbf{x}-\mathbf{m}_i) = \frac{1}{(2\pi)^{K/2}|\Sigma_i|^{1/2}}\exp(-(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i)/2) \tag{5}$$

where $\mathbf{m}_i$ is the mean vector and Σ_i is the covariance matrix. Inspired by the works on robust mixture modeling using t-distribution [33,34], in this paper, we also introduce the multi-dimensional Student's t-distribution as the heavy-tailed alternative choice of kernel,

$$T_{\Theta_i}(\mathbf{x}-\mathbf{m}_i) = \frac{\Gamma(\frac{v_i+K}{2})|\Sigma_i|^{-1/2}}{(\pi v_i)^{K/2}\Gamma(\frac{v_i}{2})\{1+(\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}}\Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i)/v_i\}^{(v_i+K)/2}} \tag{6}$$

with mean vector $\mathbf{m}_i$, precision matrix Σ_i, and degree of freedom v_i. If $v_i > 2$, the covariance matrix of the distribution is $v_i(v_i-2)^{-1}\Sigma_i$. As v_i tends to infinity, the t-distribution converges to the Gaussian distribution. Equipped with this heavy-tailed kernel, the vector quantization is supposed to be more robust to outliers.

2.2. Extended to Distributed Cases

When it comes to distributed cases, e.g., sensor networks, massive data are collected/stored by each sensor, and data from all sensors are needed in the quantization task to make use of the overall data information. However, due to the limited power and limited communication resource of nodes, transmitting the large amounts of data to a processing center might be a heavy burden for the nodes. In the following, we show that our iterative solution obtained for the centralized case can be easily extended to distributed cases.

We consider a general network modeled to be a connected graph with no nodes isolated. Each node is connected to several nodes which are called neighbors. Each node communicates only with its one-hop neighbors. In this case, we introduce the diffusion cooperation among nodes to develop the corresponding distributed vector quantization algorithm. The distributed estimation algorithm with the diffusion cooperation consists of two steps, local updating and fusion-based, on information exchanging.

Specifically, each node j firstly uses parts of its own input vectors to iteratively estimate its local reproduction vectors,

$$\tilde{\mathbf{m}}_i^j(n+1) = \tilde{\mathbf{m}}_i^j(n) + \alpha(n) \left[\frac{1}{g\left(\mathbf{x}^j(n) \middle| \Theta_i^j\right)} \frac{\partial g\left(\mathbf{x}^j(n) \middle| \Theta_i^j\right)}{\partial \tilde{\mathbf{m}}_i^j(n)} \right] \tag{7}$$

After the updating, each node sends its local estimates to its neighbors. Then each node combines the information from its neighbors to obtain fused estimates of reproduction vectors,

$$\mathbf{m}_i^j(n') = \sum_{l \in \{B_j, j\}} c_{jl} \ \tilde{\mathbf{m}}_i^l(n') \tag{8}$$

where B_j is the neighbor set of node j, and $\{c_{jl}\}$ are some combination coefficients satisfying $\sum_{l \in_j} c_{jl} = 1$, $c_{jl} = 0$, if $l \notin B_j$. As we see from (7) and (8), though the objective is to minimize the divergence between the global data distribution $p(\mathbf{x})$ and the distribution of reproduction vectors, we do not need to know (or estimate) $p(\mathbf{x})$, which is nontrivial in the distributed environment. Each node only needs its own data in Equation (7).

Each node repeats the above process until its fused estimates converge. In detail, we let a node come to OFF state when the maximum change of fused estimates during a period of iterations is less than a threshold. OFF nodes stop computation or communication and their neighbors use the last results transmitted by the OFF nodes to continue updating and fusion. The algorithm ends when all nodes become OFF. As the diffusion cooperation process goes on, local data information is diffused over the whole network without transmitting the original data. Finally, all nodes obtain consistent local estimates of reproduction vectors based on global data information.

For clarity, our distributed vector quantization algorithm based on diffusion cooperation is summarized as follows.

Initialization: Initialize the threshold value, the kernel parameters, and reproduction vectors for each node.
Computation: Each node performs the following process until the termination rule is satisfied.

1. Use parts of the node's local input vectors to iteratively update the estimates via Equation (7).
2. Transmit the local estimation results to neighbors.
3. Fuse the results from the neighbors to obtain fused estimates of the reproduction vectors via Equation (8).

Termination rule: The algorithm ends when the states of all nodes are OFF.

In [35,36], the authors have provided a detailed convergence analysis for a general class of distributed R-M–based algorithms. They have proved that *in a connected network*, under a set of explicit assumptions, the distributed R-M–based algorithms can converge to a consistent point (for different nodes) and the point is a critical point of the corresponding objective functions. Their results are fully applicable to our case when the threshold value is sufficiently close to zero. In practice, we usually use a small positive threshold value (large threshold values are not suggested since the algorithms would be forced to stop while the estimates are far from convergence) to make sure that the algorithm stops in a finite number of steps. In such a case, the different nodes may not converge to a strictly consistent point. Intuitively, a larger threshold value makes the algorithm stop in a smaller number of steps, but meanwhile it also enlarges the inconsistency degree of local estimates at different nodes. In the following simulations, we study the effects of the threshold value on the convergence in detail.

2.3. Communication Complexity Analysis

In this subsection, we provide an analysis of the communication complexity of our distributed algorithm. In each iteration loop, each node transmits M local estimates of reproduction vectors to its neighbors. Let $|B_j|$ denote the number of its neighbors, and then the communication complexity for one node in one iteration loop is $O(|B_j| M)$. Let T_j denote the number of iterations executed by the node, then the total communication complexity for the node is $O(|B_j| T_j M)$. On the other hand, the traditional centralized algorithm needs to gather all input data to a processing center. Let N_j denote the number of input vectors of node j, then the communication complexity is $O(N_j H_j)$, where H_j is the number of hops from the node to the central processor. Since the number of input vectors usually is very large compared with the other quantities, the proposed distributed algorithm can significantly reduce the communication complexity in such cases.

3. Numerical Experiments

We study the performance of our proposed algorithms by simulations in this section. We denote our distributed vector quantization using K-L divergence as d-KL and denote the corresponding centralized vector quantization as c-KL. We employ two types of kernel in the simulations, which are the Gaussian kernel (g kernel) and the Student's t kernel (t kernel), respectively. For comparison, the simulation results obtained by centralized LBG (c-LBG) and centralized SOM (c-SOM) algorithms are presented. Since the simulation results in [29] show that the performances of the distributed LBG and distributed SOM are very close to those of the c-LBG and c-SOM, here we do not additionally provide the simulation results of the distributed LBG and distributed SOM. In addition, for the distributed cases, the case without cooperation among nodes is also tested and denoted as nc-KL.

3.1. Data Generation and Evaluation Indexes

We use the noisy double-moon data as the synthetic experimental data, which are widely used as a benchmark in signal processing and machine learning [8,28,29]. The noise distribution considered is heavy-tailed, which leads to data samples containing more outliers than those under Gaussian noise. Though, according to the analysis in Section 2.3, our d-KL algorithm has advantages over the c-KL on communication complexity when the data sample amount is large, for easy implementation of the simulation, we set a relatively small number of data samples for an individual node, which is 700 (in each run, for each node, 200 samples are used as training data and the other 500 samples are used as testing data).

Moreover, we test our distributed algorithm under the cases of node unbalance. The unbalance degree is classified into five levels. In level 1, for each node, the input data amounts of two different moons (up, down) are equal. In level 2, for five nodes, the percentages of input data of (up, down) moons are (60%, 40%), respectively, and for the other five nodes, the percentages are (40%, 60%). The percentages are (70%/30%, 30%/70%) for level 3, (80%/20%, 20%/80%) for level 4, and

(90%/10%, 10%/90%) for level 5, respectively. The total percentages of two different moons over the whole network are kept equal.

Generally, the performance of a vector quantization algorithm is measured by the average (or total) distortion between input vectors and their corresponding reproduction vectors, $D(q) = \frac{1}{N}\sum_{n=1}^{N} d(\mathbf{x}(n), q(\mathbf{x}(n)))$, where $d(\cdot,\cdot)$ is a distortion measure function. We choose the commonly used Euclidean distance as the distortion measure for the convenience of comparison. For the distributed algorithms, we evaluate the inconsistency degree of local estimates at different nodes by $\frac{1}{MK}\sum_{i=1}^{M}\sum_{k=1}^{K} \mathrm{std}_j(\overline{\mathbf{m}}_i^j(k))$, where $\overline{\mathbf{m}}_i^j(k)$ stands for the normalized (normalized to the maximum $\left|\mathbf{m}_i^j(k)\right|$ among all nodes) k-th component of the i-th estimated reproduction vector, and function $\mathrm{std}_j(\cdot)$ calculates the standard deviation of the corresponding estimates over all nodes. Besides, to provide more information about the convergence rate, we report the average number of iteration steps of the network, which is calculated as $\frac{1}{J}\sum_{j=1}^{J} NoIS_{OFF}(j)$, where $NoIS_{OFF}(j)$ stands for the number of iteration steps (NoIS) needed before the node j becomes OFF.

3.2. Results

There are some crucial parameters in our proposed algorithms, such as the degree of node unbalance, the threshold value, the number of reproduction vectors, the network structure, *etc.* In the following simulations, we study the effects of the various parameters on the proposed algorithms in detail.

Firstly, we fix the threshold value as 0.002, the number of reproduction vectors as 10, and test the performance of the algorithms under different degrees of node unbalance. Here we consider a network composed of 10 nodes, which are randomly distributed in a region. We let each node connect to its nearest two nodes, and then randomly add some long-range connections with a probability of 0.1. In this paper, we set the linear combination coefficients according to the Metropolis rule [37], which is an efficient and commonly used rule in designing the combination coefficients for distributed processing [11,15]. For the distributed quantization algorithms, we let each node process 20% of its training data in one updating-fusion iteration loop. In Figure 2, we present the quantization error of the algorithms under different data unbalance levels as described above. All the results are the averages of 20 independent trials. For distributed algorithms, the quantization errors are averaged results over the whole network. Since the total percentages of two moons are always balanced, the node unbalance would not affect the performance of the centralized algorithms. Their quantization error is presented as straight lines. The distortion performances of the c-KLs, including types of both g kernel and t kernel, are better than those of c-LBG and c-SOM. As expected, c-KL with t kernel outperforms c-KL with g kernel slightly. The distortions of our distributed algorithm (with diffusion cooperation) are very close to and sometimes even lower than (similar phenomena are also found in studies on distributed clustering [25,28]) that of the corresponding centralized algorithm. Besides, its performances are hardly affected by the node unbalance. In comparison, the distortion of the non-cooperative distributed algorithm increases quickly with the unbalance degree. Figure 3 shows the learning curves of the first components of the 10 reproduction vectors for the d-KL (t kernel) algorithm under unbalance level 3 in one trial. We see that the local estimated reproduction vectors at different nodes are quite consistent when the algorithm converges.

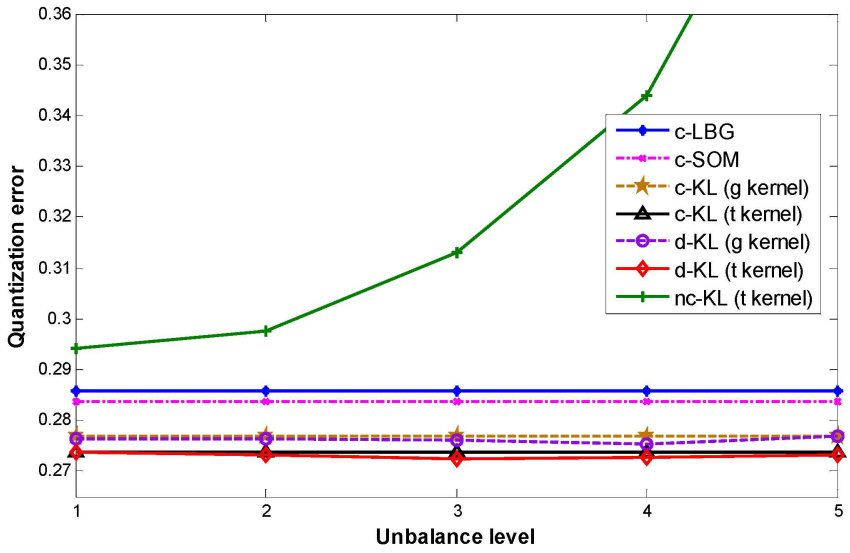

Figure 2. The distortion performances of vector quantization algorithms under different levels of unbalance.

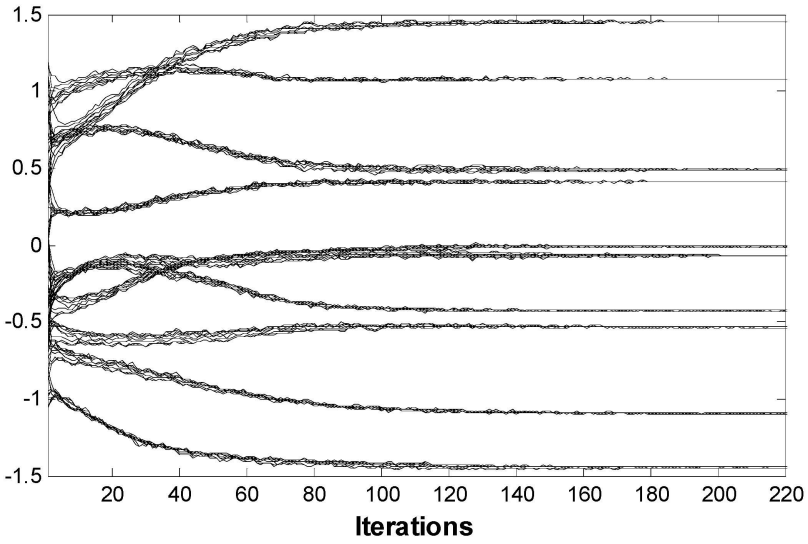

Figure 3. The learning curves of reproduction vectors for the d-KL (t kernel) algorithm under unbalance level 3.

Next, for the d-KL algorithm, we let the proportion of input data processed at each node per iteration vary from 10% to 100% under unbalance level 3, and we study the corresponding quantization error and inconsistency degree of local estimates at different nodes. The results are presented in Figure 4. We see that as the proportion increases, the quantization error of d-KL does not change obviously, while the inconsistency degree increases slightly with the proportion. The reason is that, in each iteration loop, though the fusion step still introduces global data information when the proportion is large, the ratio of global information to local information decreases with the increase of the (local data) proportion, which finally leads to the increasing difference in fused estimates among different nodes. Nevertheless, the overall inconsistency degree is quite low. Besides, the d-KL with t kernel sustainedly outperforms the d-KL with g kernel under different proportions.

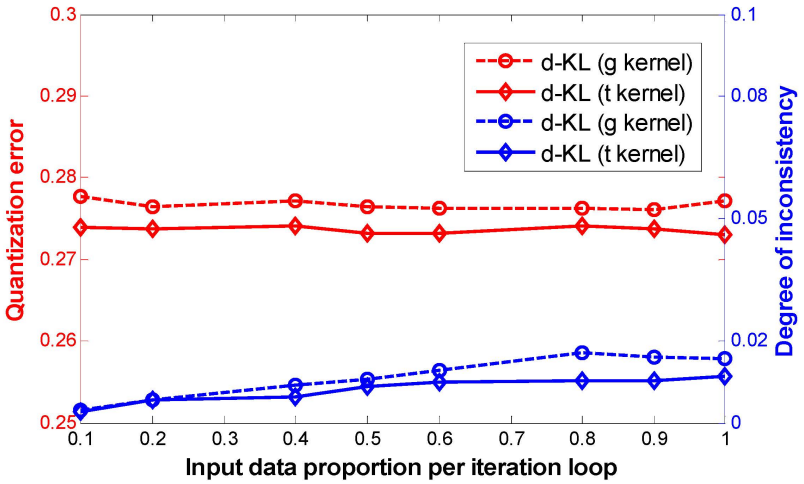

Figure 4. The quantization error and inconsistency degree of the d-KL algorithm under different settings of data proportion.

Secondly, we fix the unbalance level as 3, the proportion of input data processed at each node per iteration as 20%, and we let the threshold value vary from 0.0005 to 0.02 to study the effect of the threshold value on the convergence of the d-KL (since, in the above simulations, the d-KL with t kernel sustainedly outperforms the d-KL with g kernel, we take the d-KL with t kernel as a representative in the following simulations). Other parameters are kept the same as those used in the above simulations. The corresponding results are shown in Figure 5. We see that, in such a range, the quantization error of d-KL is kept nearly the same, which indicates that the algorithm converges to similar sets of reproduction vectors. Besides, as the threshold value increases, the average number of iteration steps decreases while the inconsistency degree of the local estimates increases. This phenomenon is reasonable since a relatively large threshold value allows the nodes to become OFF at a relatively early stage of convergence (when the estimates at nodes are still varying in a relatively wide range) and thus reduce the average number of iteration steps. Correspondingly, the information of estimates exchanging among nodes is reduced, resulting in a larger inconsistency degree.

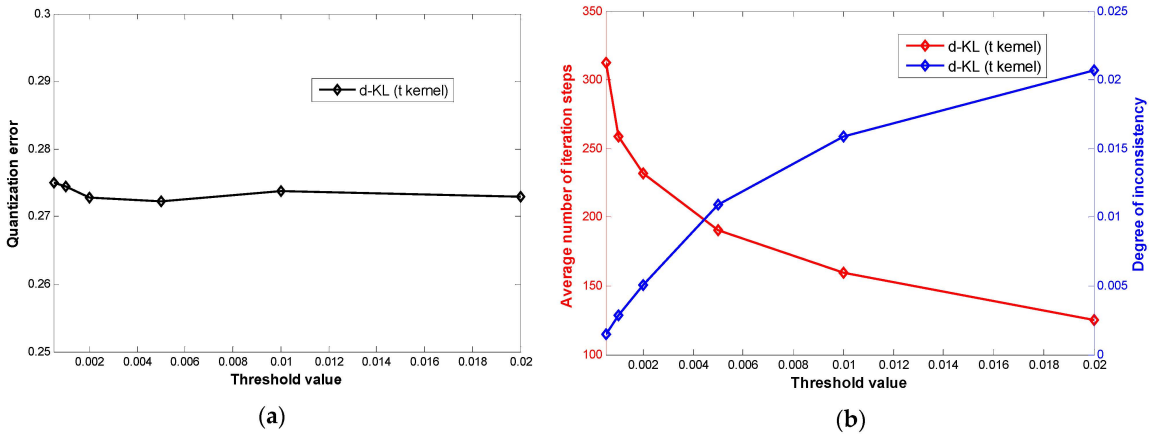

Figure 5. Performances of the d-KL algorithm under different threshold values. (**a**) Quantization error; (**b**) Average number of iteration steps and inconsistency degree.

Thirdly, we study the performances of the algorithms under different numbers of reproduction vectors ranging from six to 16. The threshold value is set as 0.002 and other parameters are kept

the same as those used above. The result is shown in Figure 6. We see that the quantization error decreases with the increasing number of reproduction vectors. The distortion performances of c-KL and d-KL outperform other algorithms within the testing range, and the performance of d-KL is very close to that of c-KL. This result shows that the effectiveness of our proposed quantization algorithms is robust to the choice of the amount of reproduction vectors.

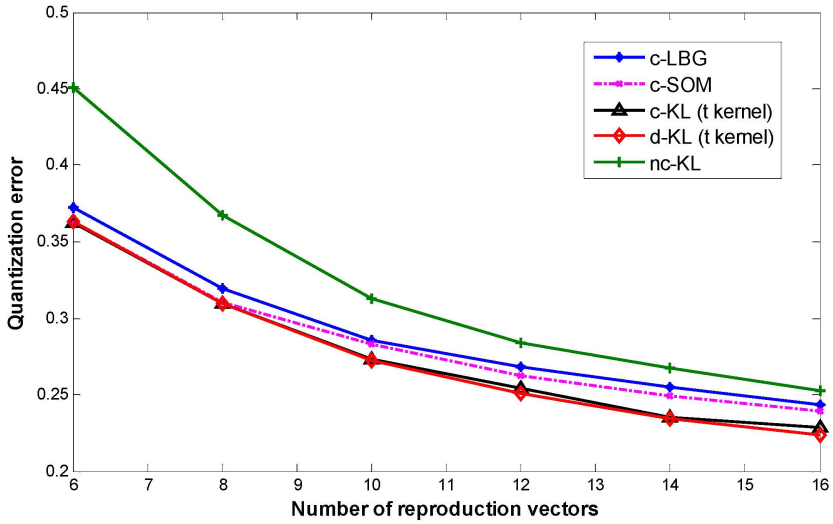

Figure 6. The distortion performances of vector quantization algorithms under different numbers of reproduction vectors.

Fourthly, we study the effect of network structures on the performance of d-KL in detail. We consider networks of different degrees of connectivity and of different topology. For the former case, we generate the network of 10 nodes in a way similar to that used above except that we let the probability of long-range connections P_{long} vary from 0 (resulting in a regular network) to 1 (resulting in a full-connected network). For the latter case, we generate four types of networks of 40 nodes in a way similar to that employed in [15], which are the Watts-Strogatz small-world network (WS), the Barabási-Albert scale-free network (BA), the regular network (RG), and the Erdos-Rényi random network (ER), respectively. In the latter case, the total amounts of connections/links are set as 80. We test the d-KL algorithm on the various networks and show the simulation results in Figure 7 and Table 1. From Figure 7 we see that as the P_{long} increases, the quantization error of d-KL does not change much, while both the average number of iteration steps and the inconsistency degree of local estimates monotonically decrease. The reason is that a large value of P_{long} increases the degrees of connectivity of the network and makes the information spread more efficiently over the network. Thus, at each iteration step, an individual node can utilize more information provided by its neighbors to help its local estimation, which finally makes the algorithm converge faster and makes the inconsistency degree of local estimates smaller. From Table 1 we see that the ER random network has the best performances in terms of all three evaluation indexes, especially for the index of inconsistency degree. It is because the ER random network has the shortest average path length among the four types of networks and has the highest efficiency in information spreading. In contrast, the regular network which has the longest average path length, results in the lowest performances. Similar comparison results have also been observed in the study of distributed least mean squares algorithms [15]. These results give a guideline for the design of the network topology in distributed vector quantization.

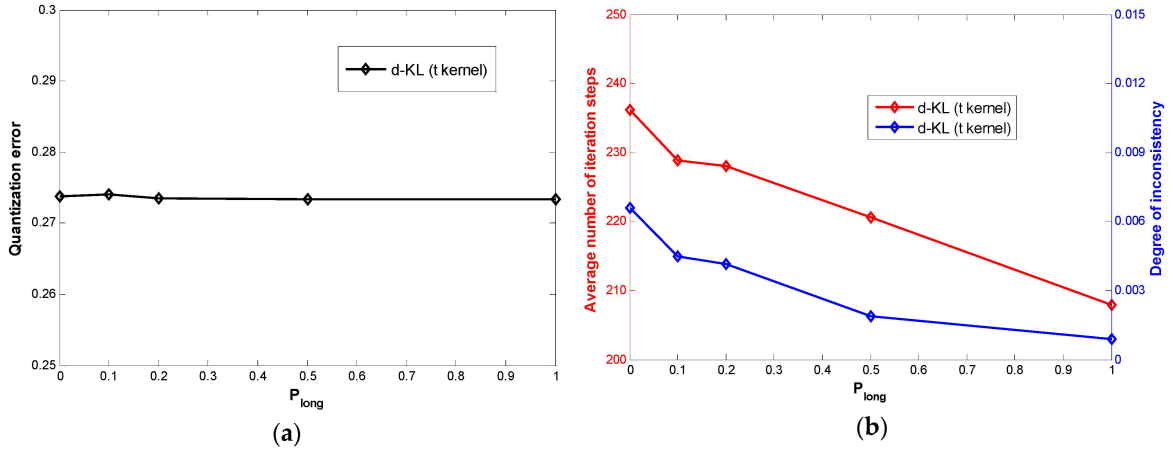

Figure 7. Performances of the d-KL algorithm under different values of P_{long}. (**a**) Quantization error; (**b**) Average number of iteration steps and inconsistency degree.

Table 1. Performances of the d-KL algorithm under networks of different topology.

	Quantization Error	Average NoIS	Inconsistency Degree
WS	0.273	227	8.23×10^{-3}
ER	0.273	225	4.43×10^{-3}
BA	0.273	232	7.93×10^{-3}
RG	0.274	236	1.22×10^{-2}

4. Conclusions

In this paper, we have developed a distributed K-L–based vector quantization algorithm. We only transmit limited intermediate estimates rather than original data, and thus communication complexity is reduced and data privacy is protected to some extent. Simulation results show that the d-KL algorithm can achieve performances similar to the corresponding c-KL algorithm. Besides, both the centralized and distributed K-L–based VQ algorithms show more robustness to outliers than the centralized LBG and the centralized SOM algorithm.

Acknowledgments: This work was supported by the National Natural Science Foundation of China (Grant Nos. 61171153, 61471320, and 61571392) and the National Program for Special Support of Eminent Professionals.

Author Contributions: Pengcheng Shen and Chunguang Li conceived and designed the experiments; Pengcheng Shen and Yiliang Luo performed the experiments; Pengcheng Shen and Chunguang Li wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gray, R.M. Vector quantization. *IEEE ASSP Mag.* **1984**, *1*, 4–29. [CrossRef]
2. Gray, R.M.; Neuhoff, D.L. Quantization. *IEEE Trans. Inf. Theory* **1998**, *44*, 2325–2383. [CrossRef]
3. Linde, Y.; Buzo, A.; Gray, R.M. An algorithm for vector quantizer design. *IEEE Trans. Commun.* **1980**, *28*, 84–95. [CrossRef]
4. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [CrossRef]
5. Kohonen, T. The self-organizing map. *Proc. IEEE* **1990**, *78*, 1464–1480. [CrossRef]
6. Nasrabadi, N.M.; Feng, Y. Vector quantization of images based upon the Kohonen self-organizing feature maps. In Proceedings of the 1988 IEEE International Conference on Neural Networks, San Diego, CA, USA, 24–27 July 1988; pp. 101–108.
7. Vesanto, J.; Alhoniemi, E. Clustering of the self-organizing map. *IEEE Trans. Neural Netw.* **2000**, *11*, 586–600. [CrossRef] [PubMed]

8. Lehn-Schiøler, T.; Hegde, A.; Erdogmus, D.; Principe, J.C. Vector quantization using information theoretic concepts. *Nat. Comput.* **2005**, *4*, 39–51. [CrossRef]
9. Hegde, A.; Erdogmus, D.; Lehn-Schioler, T.; Rao, Y.N.; Principe, J.C. Vector-Quantization by density matching in the minimum Kullback-Leibler divergence sense. In Proceedings of the IEEE International Joint Conference on Neural Networks, Budapest, Hungary, 25–29 July 2004; Volume 1, pp. 105–109.
10. Sayed, A.H. Adaptive Networks. *Proc. IEEE* **2014**, *102*, 460–497. [CrossRef]
11. Lopes, C.G.; Sayed, A.H. Diffusion Least-Mean Squares Over Adaptive Networks: Formulation and Performance Analysis. *IEEE Trans. Signal Process.* **2008**, *56*, 3122–3136. [CrossRef]
12. Cattivelli, F.S.; Lopes, C.G.; Sayed, A.H. Diffusion recursive least-squares for distributed estimation over adaptive networks. *IEEE Trans. Signal Process.* **2008**, *56*, 1865–1877. [CrossRef]
13. Schizas, I.D.; Mateos, G.; Giannakis, G.B. Distributed LMS for consensus-based in-network adaptive processing. *IEEE Trans. Signal Process.* **2009**, *57*, 2365–2382. [CrossRef]
14. Mateos, G.; Schizas, I.D.; Giannakis, G.B. Distributed recursive least-squares for consensus-based in-network adaptive estimation. *IEEE Trans. Signal Process.* **2009**, *57*, 4583–4599. [CrossRef]
15. Liu, Y.; Li, C.; Tang, W.K.; Zhang, Z. Distributed estimation over complex networks. *Inf. Sci.* **2012**, *197*, 91–104. [CrossRef]
16. Liu, Y.; Yang, C.; Tang, W.K.; Li, C. Optimal topological design for distributed estimation over sensor networks. *Inf. Sci.* **2014**, *254*, 83–97. [CrossRef]
17. Kar, S.; Moura, J.M.F. Convergence rate analysis of distributed gossip (linear parameter) estimation: Fundamental limits and tradeoffs. *IEEE J. Sel. Top. Signal Process.* **2011**, *5*, 674–690. [CrossRef]
18. Kar, S.; Moura, J.M.F.; Ramanan, K. Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication. *IEEE Trans. Inf. Theory* **2012**, *58*, 3575–3605. [CrossRef]
19. Liu, Y.; Li, C.; Zhang, Z. Diffusion sparse least-mean squares over networks. *IEEE Trans. Signal Process.* **2012**, *60*, 4480–4485. [CrossRef]
20. Liu, Z.; Liu, Y.; Li, C. Distributed Sparse Recursive Least-Squares over Networks. *IEEE Trans. Signal Process.* **2014**, *62*, 1386–1395. [CrossRef]
21. Olfati-Saber, R. Distributed Kalman filter with embedded consensus filters. In Proceedings of the Joint IEEE Conference Decision Control/European Control Conference, Seville, Spain, 6–9 December 2005; pp. 8179–8184.
22. Carli, R.; Chiuso, A.; Schenato, L.; Zampieri, S. Distributed Kalman filtering using consensus strategies. *IEEE J. Sel. Areas Commun.* **2008**, *26*, 622–633. [CrossRef]
23. Aldosari, S.A.; Moura, J.M. Topology of sensor networks in distributed detection. In Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, Toulouse, France, 14–19 May 2006; pp. 1061–1064.
24. Aldosari, S.A.; Moura, J.M. Distributed detection in sensor networks: Connectivity graph and small world networks. In Proceedings of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 30 October–2 November 2005; pp. 230–234.
25. Forero, P.A.; Cano, A.; Giannakis, G.B. Distributed clustering using wireless sensor networks. *IEEE J. Sel. Top. Signal Process.* **2011**, *5*, 707–724. [CrossRef]
26. Datta, S.; Giannella, C.R.; Kargupta, H. Approximate distributed k-means clustering over a peer-to-peer network. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1372–1388. [CrossRef]
27. Li, C.; Shen, P.; Liu, Y.; Zhang, Z. Diffusion Information Theoretic Learning for Distributed Estimation over Network. *IEEE Trans. Signal Process.* **2013**, *61*, 4011–4024. [CrossRef]
28. Shen, P.; Li, C. Distributed Information Theoretic Clustering. *IEEE Trans. Signal Process.* **2014**, *62*, 3442–3453. [CrossRef]
29. Li, C.; Luo, Y. Distributed Vector Quantization over Sensor Network. *Int. J. Distrib. Sensor Netw.* **2014**, *2014*. [CrossRef]
30. Robbins, H.; Monro, S. A stochastic approximation method. *Ann. Math. Stat.* **1951**, *22*, 400–407. [CrossRef]
31. Yin, H.; Allinson, N.M. Self-organizing mixture networks for probability density estimation. *IEEE Trans. Neural Netw.* **2001**, *12*, 405–411. [CrossRef] [PubMed]
32. Cichocki, A.; Amari, S.-I. Families of alpha- beta- and gamma- divergences: Flexible and robust measures of similarities. *Entropy* **2010**, *12*, 1532–1568. [CrossRef]

33. Peel, D.; McLachlan, G.J. Robust mixture modeling using the t distributions. *Stat. Comput.* **2000**, *10*, 335–344. [CrossRef]
34. Wei, X.; Li, C. The infinite Student's t-mixture for robust modeling. *Signal Process.* **2012**, *92*, 224–234. [CrossRef]
35. Bianchi, P.; Fort, G.; Hachem, W.; Jakubowicz, J. Performance of a distributed Robbins-Monro algorithm for sensor networks. In Proceedings of the 19th EUSIPCO, Barcelona, Spain, 29 August–2 September 2011; pp. 1030–1034.
36. Bianchi, P.; Fort, G.; Hachem, W.; Jakubowicz, J. Convergence of a distributed parameter estimator for sensor networks with local averaging of the estimates. In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Praha, Czech Republic, 22–27 May 2011; pp. 3764–3767.
37. Xiao, L.; Boyd, S. Fast linear iterations for distributed averaging. *Syst. Control Lett.* **2004**, *53*, 65–78. [CrossRef]