Pattern Recognition Letters 34 (2013) 963–969

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Decentralized Estimation using distortion sensitive learning vector quantization

Mihajlo Grbovic *, Slobodan Vucetic

Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA

ARTICLE INFO

Article history:
Received 20 April 2012
Available online 27 February 2013

Communicated by Y. Ma

Keywords:
Quantization
Distributed estimation
Non-linear estimation
Vector quantization
Aerosol retrieval

ABSTRACT

A typical approach in supervised learning when data comes from multiple sources is to send original data from all sources to a central location and train a predictor that estimates a certain target quantity. This can be inefficient and costly in applications with constrained communication channels, due to limited power and/or bitlength constraints. Under such constraints, one potential solution is to send encoded data from sources and use a decoder at the central location. Data at each source is summarized into a single codeword and sent to a central location, where a target quantity is estimated using received codewords. This problem is known as Decentralized Estimation. In this paper we propose a variant of the Learning Vector Quantization (LVQ) classification algorithm, the Distortion Sensitive LVQ (DSLVQ), to be used for encoder design in decentralized estimation. Unlike most related research that assumes known distributions of source observations, we assume that only a set of empirical samples is available. DSLVQ approach is compared to previously proposed Regression Tree and Deterministic Annealing (DA) approaches for encoder design in the same setting. While Regression Tree is very fast to train, it is limited to encoder regions with axis-parallel splits. On the other hand, DA is known to provide state-of-the-art performance. However, its training complexity grows with the number of sources that have different data distributions, due to over-parametrization. Our experiments on several synthetic and one real-world remote sensing problem show that DA has limited application potential as it is highly impractical to train even in a four-source setting, while DSLVQ is as simple and fast to train as the Regression Tree. In addition, DSLVQ shows similar performance to DA in experiments with small number of sources and outperforms DA in experiments with large number of sources, while consistently outperforming the Regression Tree algorithm.

1. Introduction

In the decentralized estimation setup, the physical quantities (e.g., temperature, pressure) are measured at distributed locations by sensor devices with limited energy and memory capacities and sent via constrained communication channels to a fusion center where a desired target variable is estimated. With advances in sensing technology, there is a growing interest in designing decentralized estimation systems with sensors that provide multivariate observations (e.g., camera sensor networks) and multi-type sensors that measure different quantities (e.g. power plants). The target variable y can be either a continuous quantity to be estimated from noisy sensor observations of the same type $x = y + noise$, a continuous function of potentially multi-type and multi-dimensional sensor observations $\mathbf{x}$ or a binary variable that indicates a certain event. In the literature, Decentralized Detection (Nguyen et al., 2005; Xiao and Luo, 2005) is used for categorical target and Decentralized Estimation for numerical target. Applications of Decentralized Estimation can be found in remote-sensing, sonar and seismology systems. It has been a focus of considerable research in the past two decades (Lam and Reibman, 1993; Fang and Li, 2010; Xiao et al., 2008; Grbovic and Vucetic, 2009; Megalooikonomou and Yesha, 2000; Rao et al., 1996; Gubner, 1993; Li, 2007).

Given a bitlength constraint on messages, the sensors can transmit only a summary message instead of the original measurement. The principal approach is to design encoders at sensor sites and a decoder at the fusion site that reconstructs the target variable from the received encoder messages. The encoder types range from linear threshold-based quantizers (Fang and Li, 2010; Xiao and Luo, 2005; Xiao et al., 2008) to non-linear decision tree- (Megalooikonomou and Yesha, 2000), nearest prototype- Grbovic and Vucetic, 2009; Rao et al., 1996 and kernel-based (Nguyen et al., 2005) quantizers. In most previous work, encoder design is carried out by assuming that the joint distribution $\mathbb{P}(\mathbf{x}, y)$ is known (Lam and Reibman, 1993; Xiao and Luo, 2005; Xiao et al., 2008; Fang and Li, 2010; Wang et al., 2009). In practice this is violated, either because the parameters of $\mathbb{P}(\mathbf{x}, y)$ are unknown or even less is known

* Corresponding author. Tel.: +1 (215) 204 7230; fax: +1 (215) 204 5082.
E-mail addresses: mihajlo.grbovic@temple.edu (M. Grbovic), slobodan.vucetic@temple.edu (S. Vucetic).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.02.011

about it. Here, we consider a more realistic scenario and address the problem using a set of empirical samples of sensor and target measurements (Megalooikonomou and Yesha, 2000; Rao et al., 1996; Grbovic and Vucetic, 2009). In many real-world applications it is reasonable to assume that certain amount of source an target data can be collected for purposes of encoder and decoder design. Finally, unlike some methods (Fang and Li, 2010; Li, 2007) that require sensors to communicate among themselves, we consider systems where each sensor communicates only with the fusion center. This setup is favorable as it does not increase the communication cost on account of solving the bitlength constraint problem.

The proposed methodology was evaluated on several synthetic problems with different number of sources and on a real-world problem of predicting aerosol optical depth (AOD) from remotely sensed data. Multisource observations in form of multispectral images are collected from sensors aboard satellites across the entire globe, while target AOD observations are collected from limited number of ground-based sensors placed at world-wide locations. The collocated data are used to train encoders for satellites and a decoder for AOD prediction, which reduces communication cost and allows prediction across the entire globe.

2. Problem setup

The general setup assumes a system of n distributed data sources $S_1,\dots,S_n$ that produce multivariate vectors $\mathbf{x}_1,\dots,\mathbf{x}_n$ drawn from probability distribution $\mathbb{P}(\mathbf{x}_1,\dots,\mathbf{x}_n)$, and a fusion center (Fig. 1). It is assumed that the random vectors $\mathbf{x}_i, i=1,\dots,n$, are related to the unobservable continuous quantity y that the fusion center needs to estimate and there exists a joint distribution $\mathbb{P}(\mathbf{x}_1,\dots,\mathbf{x}_n,y)$.

Due to the channel bandwidth and energy constraints, the ith sensor communication is limited to M_i codewords. Let α_i be the quantization function for source S_i. Instead of the original vectors, $\mathbf{x}_1,\dots,\mathbf{x}_n$, the data sources transmit quantized messages $z_1,\dots,z_n, z_i=\alpha_i(\mathbf{x}_i)$, to the fusion center, where z_i is as an integer from set $\{1,\dots,M_i\}$. Let h be the function of the fusion center that gives the estimate $y=h(z_1,\dots,z_n)$ of y, given the quantizers $\alpha_1,\dots,\alpha_n$. At the fusion center, the goal is to find the point estimate of y that minimizes the distortion measure d. The optimal estimate in this case is the conditional expectation of a random vector $Y, \mathbb{E}(Y|\mathbf{x}_1,\dots,\mathbf{x}_n)$.

Under this setup, the problem of decentralized estimation is to find the quantization functions $\alpha_1,\dots,\alpha_n$ and the fusion function h such that the estimation error $\mathbb{E}_X[d\{\mathbb{E}_Y(Y|X_1,\dots,X_n), h(\alpha_1(X_1),\dots,\alpha_n(X_n))\}]$ is minimized under given communication constraints. Special case of the estimation error under assumption that d is defined as the Mean Squared Error was formulated in (Gubner, 1993) as $error=\mathbb{E}_X[(\mathbb{E}_Y(Y|X_1,\dots,X_n)-h(\alpha_1(X_1),\dots,\alpha_n(X_n)))^2]$.

For decentralized estimation with the squared error distortion, when the joint probability distribution $\mathbb{P}(\mathbf{x}_1,\dots,\mathbf{x}_n,y)$ is known, necessary conditions for the optimal compression functions $\alpha_1,\dots,\alpha_n$ and fusion function h were derived (Lam and Reibman, 1993). Assuming, without loss of generality, that $n=2$, they are:

Condition 1
(Decoder design). Given α_1 and α_2, the optimal h is given by $h(z_1,z_2)=\mathbb{E}(Y|\alpha_1(\mathbf{x}_1)=z_1,\alpha_2(\mathbf{x}_2)=z_2)$, where $z_1\in\{1,\dots,M_1\}$ and $z_2\in\{1,\dots,M_2\}$.

Condition 2
(Encoder design). Given α_2 and h, optimal quantization function α_1 is determined as $\alpha_1(\mathbf{x}_1)=\arg\min_j\mathbb{E}_{X_2}[(\mathbb{E}_Y(Y|\mathbf{x}_1,X_2)-h(j,\alpha_2(X_2)))^2]$, where $j\in\{1,\dots,M_1\}$.

The conditions lead to the solution by the generalized Lloyds algorithm where the construction of the decentralized system is performed iteratively. One step consists of optimizing the fusion function h while fixing the local quantization functions at each sensor, whereas another step involves optimizing the quantization function at a given sensor while fixing h and the quantization functions of the remaining sensors.

The problem addressed in this paper arises when only a set of examples from the underlying distribution $\mathbb{P}(\mathbf{x}_1,\mathbf{x}_2,y)$ is available, $D=\{(\mathbf{x}_{1i},\mathbf{x}_{2i},y_i), i=1,\dots,N\}$, where $\mathbf{x}_{1i}$ is a K_1-dimensional vector and $\mathbf{x}_{2i}$ a K_2-dimensional vector. Given D and MSE as the distortion measure, the expectation of the estimation error can be formulated as

$$error=\frac{1}{N}\sum_{i=1}^{N}(y_i-h(\alpha_1(\mathbf{x}_{1i}),\alpha_2(\mathbf{x}_{2i})))^2. \tag{1}$$

We propose an iterative procedure for decoder and encoder design that minimizes (1) with respect to h and α_1,α_2.

3. Methodology

Let us first consider the decoder design. Following *Condition 1* and assuming that only D is available, the optimal fusion function h for each pair of codewords z_1,z_2, can be estimated by taking the average of target values y_i from examples in D that satisfy $\alpha_1(\mathbf{x}_{1i})=z_1\wedge\alpha_2(\mathbf{x}_{2i})=z_2$,

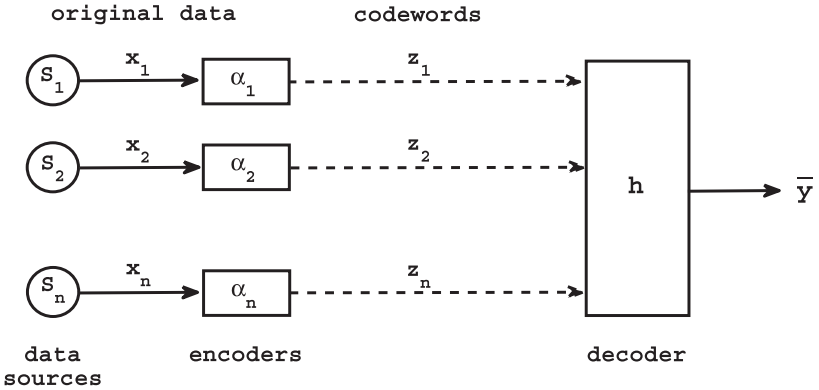

Fig. 1. Illustration of the decentralized estimation system.

$$h(z_1, z_2) = average\{y_i : \alpha_1(\mathbf{x}_{1i}) = z_1 \wedge \alpha_2(\mathbf{x}_{2i}) = z_2\}. \quad (2)$$

The disadvantage of the resulting lookup table h is that it can easily become too large due to its exponential growth in the number of sensors and the message cardinality. Alternatives, such as the Direct Sum (Gubner, 1993) method for estimating h, might be more appropriate in applications with large number of sensors.

For design of quantizer α_1, given quantizer α_2 and the fusion function h, the challenge is in partitioning the space X_1 into M_1 regions such that the estimation error is minimized. Analogously, quantizer α_2 can be designed given α_1 and h. One viable approach is based on the recursive partitioning of the X_1 space using the Regression Trees (Megalooikonomou and Yesha, 2000). An alternative, studied in this paper, is the multi-prototype approach.

In the multi-prototype approach, quantizer α_1 consists of M_1 regions, where each region $R_j, j = 1, \ldots, M_1$ is represented as a union of Voronoi cells defined by a set of prototypes. Thus, α_1 is completely defined by a set of P prototypes $\{(\mathbf{m}_k, c_k), k = 1, \ldots, P\}$, where $\mathbf{m}_k$ is a K-dimensional vector in input space and c_k is its assignment label, defining to which of the M_1 regions/codewords it belongs. The input $\mathbf{x}_{1i}$ is compared to all prototypes and its codeword z_{1i} is assigned as the assignment label of its nearest prototype, $z_{1i} = c_l, l = \arg\min_k(d_e(\mathbf{x}_{1i}, \mathbf{m}_k))$, where d_e is the Euclidean distance.

Following this setup, given D, we can restate *Condition 2* as finding the optimal set of prototypes to minimize the overall distortion of quantizer α_1

$$L = \sum_{i=1}^{N} d(y_i, h(z_{1i}, \alpha_2(\mathbf{x}_{2i}))), \quad (3)$$

where distortion d is the squared error,

$$d(y_i, h(z_{1i}, \alpha_2(\mathbf{x}_{2i})) = (y_i, h(z_{1i} - \alpha_2(\mathbf{x}_{2i}))^2. \quad (4)$$

Finding $\{(\mathbf{m}_k, c_k), k = 1, \ldots, P\}$ that minimize (3) is challenging. We consider several approaches.

Hard Classification Approach. One approach to minimize (3) is to convert the problem of encoder design to classification. To design α_1 in this manner, in each iteration we use the original training data set D to create a new data set, $D_1 = (\mathbf{x}_{1i}, q_{1i}), i = 1, \ldots, N$, where q_{1i} is the codeword with the smallest error, $q_{1i} = \arg\min_j(y_i - h(j, \alpha_2(\mathbf{x}_{2i})))^2$. The goal then becomes finding a set of prototypes, $\{(\mathbf{m}_k, c_k), k = 1, \ldots, P\}$, that minimize the classification error on D_1. Nearest Prototype Classification algorithms, such as Learning Vector Quantization (LVQ) (Kohonen, 1990), can be used for this purpose (Grbovic and Vucetic, 2009).

Let us consider the LVQ2 algorithm (Kohonen, 1990) which starts from an initial set of prototypes, and reads the training data points sequentially to update the prototypes. LVQ2 considers only the two closest prototypes. Three conditions have to be met to update the two closest prototypes: (1) Class of the prototype closest to $\mathbf{x}_{1i}$ has to be different from z_{1i}, (2) Class of the second closest prototype has to be equal to z_{1i}, and (3) $\mathbf{x}_{1i}$ must satisfy the "window rule" by falling near the hyperplane at the midpoint between the closest ($\mathbf{m}_A$) and the second closest prototype ($\mathbf{m}_B$). These two prototypes are then modified as

$$\mathbf{m}_A^{t+1} = \mathbf{m}_A^t - \eta(t)(\mathbf{x}_{1i} - \mathbf{m}_A^t),$$

$$\mathbf{m}_B^{t+1} = \mathbf{m}_B^t + \eta(t)(\mathbf{x}_{1i} - \mathbf{m}_B^t), \quad (5)$$

where t counts how many updates have been made, and $\eta(t)$ is a monotonically decreasing function of t.

Let d_A and d_B be the distances between $\mathbf{x}_{1i}$ and $\mathbf{m}_A$ and $\mathbf{m}_B$. Then, the "window rule", that was introduced to prevent divergence (Kohonen, 1990), is satisfied if $\min(d_B/d_A, d_A/d_B) > w$, where w is a constant commonly chosen between 0.4 and 0.8.

Soft Classification Approach. The potential issue with the hard classification approach is that it enforces assignment of a data point to the codeword with the minimum squared error (4). This can be too aggressive, considering that there might be other codewords resulting in a similar squared error. The soft classification approach addresses this issue. Instead of assigning a data point to the closest prototype, let us consider a probabilistic assignment $p_{ij} = \mathbb{P}(z_{1i} = j|\mathbf{x}_{1i})$ defined as the probability of assigning i-th data point to j-th codeword. The objective (3) can now be reformulated as

$$L = \sum_{i=1}^{N}\sum_{j=1}^{M_1} p_{ij} d(y_i, h(z_{1i}, \alpha_2(\mathbf{x}_{2i}))) = \sum_{i=1}^{N}\sum_{j=1}^{M_1} p_{ij} e(i,j), \quad (6)$$

where $e(i,j) = d(y_i, h(z_{1i}, \alpha_2(\mathbf{x}_{2i})))$ was introduced to simplify the notation. Objective (6) is an approximation of (3) that allows us to find a computationally efficient solution. We use a mixture model to calculate the assignment probabilities p_{ij}. Let us assume that the probability density $\mathbb{P}(\mathbf{x}_1)$ of the observation at the first sensor can be described by mixture

$$\mathbb{P}(\mathbf{x}_1) = \sum_{k=1}^{P} \mathbb{P}(\mathbf{x}_1|\mathbf{m}_k)\mathbb{P}(\mathbf{m}_k), \quad (7)$$

where $\mathbb{P}(\mathbf{x}_1|\mathbf{m}_k)$ is a conditional probability that prototype $\mathbf{m}_k$ generates observation $\mathbf{x}_1$ and $\mathbb{P}(\mathbf{m}_k)$ is the prior probability. We represent the conditional density function $\mathbb{P}(\mathbf{x}_1|\mathbf{m}_k)$ as the Gaussian distribution with mean $\mathbf{m}_k$ and standard deviation σ_s. Let us denote $g_{ik} \equiv \mathbb{P}(\mathbf{m}_k|\mathbf{x}_{1i})$ as the probability that i-th data point was generated by k-th prototype. By assuming that all prototypes have the same prior, $\mathbb{P}(\mathbf{m}_k) = 1/P$, and using the Bayes' rule, g_{ik} can be updated as

$$g_{ik} = \frac{\exp(-(\mathbf{x}_{1i} - \mathbf{m}_k)/2\sigma_s^2)}{\sum_{l=1}^{P}\exp(-(\mathbf{x}_{1i} - \mathbf{m}_l)/2\sigma_s^2)}. \quad (8)$$

The probability p_{ij} can be obtained using (8) as

$$p_{ij} = \frac{\sum_{j:c_k=j}\exp(-(\mathbf{x}_{1i} - \mathbf{m}_k)/2\sigma_s^2)}{\sum_{l=1}^{P}\exp(-(\mathbf{x}_{1i} - \mathbf{m}_l)/2\sigma_s^2)}. \quad (9)$$

In soft classification approach, the objective of learning is to estimate the prototype positions $\mathbf{m}_k, k = 1, \ldots, P$, by minimizing (6). This can be done using the stochastic gradient descent, where at t-th update the prototypes are calculated as

$$\mathbf{m}_k^{t+1} = \mathbf{m}_k^t - \eta(t) \cdot (e(i, c_k) - \sum_{j=1}^{M_1} p_{ij} e(i,j)) \cdot g_{ik} \frac{(\mathbf{x}_{1i} - \mathbf{m}_k^t)}{\sigma_s^2}. \quad (10)$$

We will refer to this as the Soft Prototype Quantization (SPQ)

Parameter σ_s controls the fuzziness of the distribution. For $\sigma_s = 0$ the assignments become deterministic, and (6) is equivalent to (3). If $\sigma_s \to \infty$ the assignments become uniform, regardless of the distance. One option is that σ_s^2 be treated as a parameter to be optimized such that (6) is minimized. It is not necessarily the best approach since minimizing (6) does not imply minimizing (3). In this work, we are treating σ_s^2 as an annealing parameter that is initially set to a large value and then is decreased towards zero using $\sigma_s^2(t+1) = \sigma_s^2(0) \cdot \sigma_T/(\sigma_T + t)$, where σ_T is the decay parameter. The purpose of annealing is to facilitate convergence toward a good local optimum of (3). We note that this strategy has been used in soft prototype approaches by other researchers (Kohonen, 1990).

Deterministic Annealing (DA) is a generalization of the soft classification approach. Instead of minimizing (6), it minimizes the regularized objective,

$$L = \beta \cdot \sum_{i=1}^{N}\sum_{j=1}^{M_1} p_{ij} e(i,j) - H, \quad (11)$$

where β controls the tradeoff between the objective (6) (first term) and the entropy $H = -\sum_{i=1}^{N}\sum_{j=1}^{M_1} p_{ij}\log p_{ij}$.

In (Rao et al., 1996) authors suggest minimizing L starting at the global minimum for $\beta = 0$ and updating the solution as β increases. As $\beta \to \infty$ (11) becomes equivalent to (6). The role of the entropy term is to further improve the convergence toward a good local optimum. The DA prototype update rule can be obtained by the stochastic gradient descent as

$$\mathbf{m}_k^{t+1} = \mathbf{m}_k^t - \eta(t)\cdot(G_i(c_k) - \sum_{j=1}^{M_1} p_{ij}G_i(j))\cdot g_{ik}\frac{(\mathbf{x}_{1i} - \mathbf{m}_k^t)}{\sigma_s^2}, \quad (12)$$

where $G_i(j) = \beta\cdot e(i,j) + \log p_{ij}$. Parameter σ_s^2 is updated as $\sigma_s^2(t+1) = \sigma_s^2(0)\cdot\sigma_T/(\sigma_T + t)$, where σ_s^2 is reset back to $\sigma_s^2(0)$ after each increase of β. The drawback is that, since the cost function is defined and minimized at each value of β, the model takes quite long to produce and the convergence can be quite sensitive to the annealing schedule of β.

Distortion Sensitive Learning Vector Quantization. To apply the stochastic gradient descent in (10), one should specify the learning rate $\eta(t)$ and the annealing rate for σ_s^2, while for the DA version in (12) the annealing schedule for β is required too. In this subsection, we show how the update rule (10) can be simplified such that it does not require use of the parameter σ_s^2. The resulting algorithm resembles LVQ2.

Objective (6) is a good approximation of (3) for small values of σ_s^2. In this case, assignment probabilities p_{ij} of all but the closest prototypes are near zero. As a result, we approximate (10) by using only the two closest prototypes. Given $\mathbf{x}_{1i}$, we denote the closest prototype as $(\mathbf{m}_A, c_A)$ and the second closest as $(\mathbf{m}_B, c_B)$.

Let us consider three major scenarios. First, if $\mathbf{m}_A$ and $\mathbf{m}_B$ are in similar proximity to $\mathbf{x}_{1i}$, their assignment probabilities will be approximately the same, $g_{iA} = g_{jB} = 0.5$. The prototype update rule from (10) could then be expressed as

$$\mathbf{m}_A^{t+1} = \mathbf{m}_A^t - \eta(t)(e(i,c_A) - e(i,c_B))(\mathbf{x}_{1i} - \mathbf{m}_A^t),$$

$$\mathbf{m}_B^{t+1} = \mathbf{m}_B^t + \eta(t)(e(i,c_A) - e(i,c_B))(\mathbf{x}_{1i} - \mathbf{m}_B^t), \quad (13)$$

where σ_s^2 is incorporated in the learning rate parameter η. The difference $(e(i,c_A) - e(i,c_B))$ determines the amount of prototype displacement; when the difference is small the prototype updates are less extreme. The sign of the difference determines the direction of updates; if $e(i,c_A)$ is larger than $e(i,c_B)$, prototype $\mathbf{m}_A$ is moved away from data point $\mathbf{x}_{1i}$ and $\mathbf{m}_B$ prototype is moved towards it.

The second scenario is when the closest prototype is much closer than the second closest, which makes $g_{iB} \sim 0$. Following (10), none of the prototypes are updated. Taken together, the first two scenarios are equivalent to the LVQ2 window rule.

In the third scenario, the two closest prototypes belong to the same codeword and, as a consequence of $e(i,c_A) = e(i,c_B)$, the prototype positions are not updated.

The three scenarios establish the new algorithm (DSLVQ2): Given $\mathbf{x}_{1i}$, if the two closest prototypes, $\mathbf{m}_A$ and $\mathbf{m}_B$, have different class labels and $\min(d_B/d_A, d_A/d_B) > w$, update their positions (13), otherwise preserve their current positions.

To avoid settling of prototypes in a bad local minima, SPQ algorithm uses annealing, while DA uses double annealing. The proposed DSLVQ2 uses a simple and much faster procedure. Harmful prototypes, stuck in local minima, are identified after every $n_s(=10)$ quantizer design iterations as the ones whose current label c_k is different from label c_k^* which introduces the least prediction error amount to its Voronoi cell

$$c_k^* = \arg\min_j\left\{\sum_{i:\mathbf{x}_{1i}\in v_k} e(i,j)\right\}, \quad (14)$$

where v_k is Voronoi cell of the k-th prototype. In this case, DSLVQ2 switches the label from c_k to c_k^*.

Relationship with LVQ2. When target variable y is categorical instead of real-valued, we could use 0–1 error, defined as $e(i,c_k) = \delta(y_i, h(c_k, \alpha_2(\mathbf{x}_{2i})))$, where $\delta(\cdot)$ is the Dirac delta function, instead of the squared error. Now, (13) reduces to (5). If, in addition, we update prototypes only if $e(i,c_A) = 1$ and $e(i,c_B) = 0$, DSLVQ2 reduces to LVQ2.

Prototype Initialization and Refinement. Regardless of whether (5), (10), (12), or (13) are used for prototype update, the algorithm starts by randomly selecting P points from D and assigning equal number of prototypes to each codeword. Unlike DSLVQ which uses a label replacement mechanism (14), and DA and SPQ which use annealing, the regular LVQ algorithm is highly sensitive to initial choice of prototypes. If the initialization is not done in a proper way good results might never be achieved. Repeating random initialization or using K-means algorithm (MacQueen, 1967) to initialize the prototypes in case of regular LVQ helps in certain cases.

4. Experiments

In order to compare quantizer design using DSLVQ2 algorithm with those using LVQ2, DA, SPQ, and Regression Tree algorithms, we performed three sets of experiments with synthetic data and one experiment with real-world AOD data. The series of experiments were of increasing complexity, meaning that the number of sources and source features grew. When compared to the DSLVQ2 algorithm, the DA and SPQ algorithms showed a performance decrease in out of sample prediction with increase in the number of sources and source types.

Setup. The algorithms were compared on different budget sizes P, representing the number of prototypes and Regression Tree nodes. In synthetic data experiments we used two training data sizes of $N = 10{,}000$ and $N = 1000$ points and a test set of size 10,000. The experiments were repeated 10 times and the average test set MSE are reported. Training of encoders and decoder was terminated when the training set MSE (1) stopped decreasing by more than 10^{-5}. Experiments were performed on Intel Core Duo 2.6 GHz processor machines with 2 GB of RAM. We also report the total time needed to design a decentralized system from training data using different encoder design algorithms.

Parameter Selection. We have empirically observed that the sensitivity of prototype-based algorithms to parameters varies significantly - they are fairly robust to some and highly sensitive to others. The learning rate η is universal for all algorithms. We initially set it to $\eta_0 = 0.03$ and update it using $\eta(t) = \eta_0\cdot\eta_T/(\eta_T + t)$, where $\eta_T = 8N$. The window parameter w, used in LVQ2 and DSLVQ2, is easy to adjust and was fixed to a value of $w = 0.7$. A common practice (Rao et al., 1996) for annealing the DA parameter β is to initially set it to a small value (e.g. 0.01) and update it using $\beta_{t+1} = \beta_T\cdot\beta^t$, where $\beta_T = 1.11$. Parameter σ_s^2 for sensor S is often (Seo et al., 2003) initialized as the sensor data variance and updated using the annealing schedule with $\sigma_T = 8N$. However, our preliminary experiments revealed that DA and SPQ are very sensitive to these parameters. As a result, parameters $\sigma_s^2, \sigma_T, \beta_T, \beta_0$ have to be adjusted from case to case depending on K, P and N.

Table 1
Parameter search grid.

Parameter	Start value	Search step	End value
σ_s^2	0.3	+0.3	sensor s data variance + 0.3
σ_T	$5N$	×2	$40N$
β_0	0.01	×2	0.08
β_T	1.05	+ 0.05	1.2

Table 2
1-sensor synthetic data performance comparison, Decoder: Lookup Table, $M = 4$.

N	Encoder	Number of training sessions	P					
			100		40		20	
			MSE	time (s)	MSE	time (s)	MSE	time (s)
N = 10,000	DA	256	**0.956**	485 K	**0.968**	254 K	**1.12**	155 K
	SPQ	16	0.975	18 K	0.981	10 K	1.17	8 K
	Reg.Tree	1	1.19	340	1.26	295	1.44	254
	LVQ2	1	1.76	1.6 K	1.94	1 K	2.48	732
	DSLVQ2	1	0.979	226	0.987	239	1.25	222
N = 1000	DA	256	**1.22**	71 K	**1.24**	30 K	1.40	22 K
	SPQ	16	1.24	1.8 K	1.32	1 K	**1.39**	645
	Reg.Tree	1	1.47	13.5	1.59	8.7	1.63	6.5
	LVQ2	1	1.86	451	2.07	322	2.81	192
	DSLVQ2	1	**1.22**	42.2	1.26	37.8	1.49	36.2

Table 3
2-sensor synthetic data performance comparison, Decoder: Lookup Table, $M = 4$.

N	Encoder	Number of training sessions	P					
			100		40		20	
			MSE	time (s)	MSE	time (s)	MSE	time (s)
N = 10,000	DA	256	**1.82**	782 K	**1.87**	384 K	2.19	245 K
	SPQ	16	2.03	36 K	1.93	18 K	**2.03**	14 K
	Reg.Tree	1	1.99	526	2.09	434	2.32	347
	LVQ2	1	2.99	2.8 K	3.21	1.8 K	3.76	1.2 K
	DSLVQ2	1	1.98	382	1.92	407	2.30	285
N = 1000	DA	256	2.51	123 K	2.47	54 K	2.68	36 K
	SPQ	16	2.46	3.2 K	2.33	2.1 K	**2.53**	1 K
	Reg.Tree	1	2.52	38.5	2.78	32.4	2.81	31.1
	LVQ2	1	3.32	1.5 K	3.58	1 K	3.93	735
	DSLVQ2	1	**2.44**	177	**2.33**	178	2.72	171

Parameter search for DA and SPQ algorithms was based on the grid displayed in Table 1. The best parameters were chosen as the ones that achieve the lowest MSE on the validation set, formed by randomly selecting 25% of the training data. The same validation set was used for Regression Tree pruning.

We note that if the sensors differ in type because they measure different quantities, the choice of σ_s^2 becomes more involved as appropriate σ_s^2 would differ for each type of sensor. If the best σ_s^2 for a single sensor type s is found in o_s training steps, the best combination of σ_s^2 values for T sensor types is found in $o_1 \cdot o_2 \cdot \ldots \cdot o_T$ training steps.

Experiments with a single two-dimensional sensor. In the first experiment we simulated a system with a single two-dimensional noisy source that generates a vector $\mathbf{x}_1 = [\mathbf{x}_{11}, \mathbf{x}_{12}]$. This scenario corresponds to the General Vector Quantization. Target variable y was generated as $y = \mathbf{x}_{11}^2 + \mathbf{x}_{12}^2 + \epsilon$, where $\mathbf{x}_{ij} \sim \mathcal{N}(0, 1.25)$ and $\epsilon \sim \mathcal{N}(0, 0.25)$. We used a quantizer with $M_1 = 4$ codewords. At the fusion site, target y was estimated using a one-dimensional lookup table h with M_1 elements.

Table 2 compares prototype-based and Regression Tree algorithms with different budget and training data sizes. It shows MSE on test data and total training time in seconds. The total training time measures the complete effort needed to build the decentralized estimation system. The largest computational effort in case of DA and SPQ goes on the grid search, as the whole training procedure is repeated numerous times in search for the best combination of parameters (the number of training sessions due to grid search is shown in the first column of the table).

The results in Table 2 show that DSLVQ2, SPQ and DA algorithms have similar accuracy, with DA being the most accurate overall. Regression Tree is significantly less accurate, while LVQ2 is the least accurate. Training time for DA is almost two orders of magnitude longer than SPQ and more than 3 orders of magnitude longer than the proposed DSLVQ2. The main reason is a significantly higher effort needed for DA to explore the parameter space. A minor difference is that a single training session for the fixed parameter choice is up to 5 times slower for DA than for DSLVQ2. Interestingly, LVQ2 is very slow, despite its simplicity. This is explained by the very slow convergence of this algorithm. Regression Tree is the fastest overall, but is comparable to DSLVQ2. The efficiency of regression tree comes at the price of significantly reduced accuracy. Overall, the proposed DSLVQ strikes a nice balance between accuracy and computational time, as it is nearly as accurate as DA and over 3 orders of magnitude faster to train.

Experiments with 2 two-dimensional sensors. In the second set of experiments, we simulated a system with 2 two-dimensional sources that generate vectors $\mathbf{x}_1 = [\mathbf{x}_{11}, \mathbf{x}_{12}]$ and $\mathbf{x}_2 = [\mathbf{x}_{21}, \mathbf{x}_{22}]$. Target variable y was generated as $y = \mathbf{x}_{11} + 2\mathbf{x}_{12} + \mathbf{x}_{21}^2 + \mathbf{x}_{22}^2 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.25)$ and the j-th attribute had Gaussian Distribution $\mathbf{x}_j \sim \mathcal{N}(\mu_j, \Sigma_j), \mu_j = (0, 0)$ and $\Sigma_j = (1.25, 0.5; 0.5, 1.25)$. The quantizers had $M_1 = M_2 = 4$ codewords. At the fusion center, the y was estimated using a 4×4 lookup table.

Table 3 summarizes total training times and test set MSE of different algorithms. It can be seen that DSLVQ2 performed slightly worse than the computationally costly DA and SPQ when training data were abundant ($N = 10,000$). However, when $N = 1000$ and $P = 100, 40$ it was more accurate than DA and SPQ and for $P = 20$ it had similar accuracy.

This is due to locally optimal solutions of DA and SPQ caused by harmful prototypes, which are moved to low density regions of training data in attempt to reduce their influence. However, these low density regions can be populated with a significant number of test data points and reflected in higher MSE. DSLVQ2 resolves this issue by the label switching strategy from (14). DSLVQ2 consis-

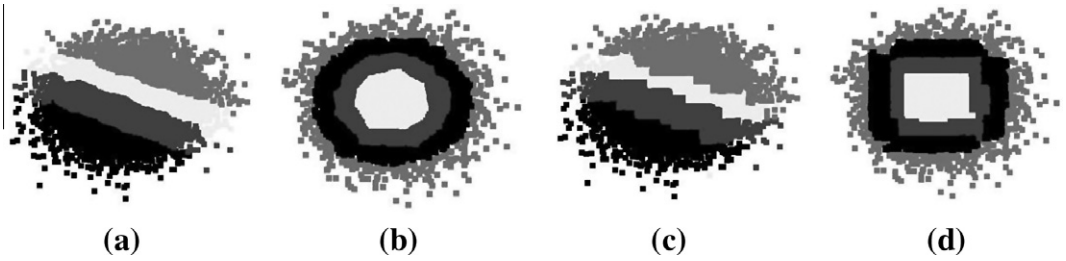

Fig. 2. 2-sensor synthetic data quantization results ($P = 40, N = 10,000$) (a) sensor 1 DSLVQ2; (b) sensor 2 DSLVQ2; (c) sensor 1 Regression Tree; (d) sensor 2 Regression Tree.

Table 4
4-sensor synthetic data performance comparison.

N	Encoder	Number of training sessions	P					
			120		80		40	
			MSE	time (s)	MSE	time (s)	MSE	time (s)
Lookup table	DA	2,160	2.42	8 M	2.65	7 M	2.82	4 M
	SPQ	108	3.13	355 K	2.57	189 K	**2.70**	126 K
	Reg.Tree	1	3.09	952	3.14	867	2.97	657
	LVQ2	1	3.33	1.6 K	3.43	1 K	4.01	732
	DSLVQ2	1	**2.38**	653	**2.55**	814	**2.70**	526
Direct sum	DA	2,160	2.06	8 M	2.10	6.4 M	2.29	3.8 M
	SPQ	108	2.77	336 K	2.33	174 K	**2.22**	122 K
	Reg.Tree	1	2.27	857	2.29	781	2.68	549
	LVQ2	1	2.61	1.5 K	2.78	960	3.22	721
	DSLVQ2	1	**2.01**	573	**2.06**	782	2.24	552

tently outperformed Regression Tree algorithm, often by significant margins, and was superior to the hard classification LVQ2 approach. The training time comparisons in Table 3 are consistent to the 1-sensor experiments reported in Table 3. DA and SPQ training times were orders of magnitude higher than DSLVQ2 and were consistent with those reported in Table 2.

To gain a further insight into the resulting encoders, Fig. 2 compares the resulting partitions of the DSLVQ2 and Regression Tree encoders, for $P = 40$ and $N = 10,000$. The partitions for sensor 1 resemble parallel lines with slope near $-1/2$, reflecting the twice larger sensitivity of target to $\mathbf{x}_{21}$ than to $\mathbf{x}_{11}$. The partitions for sensor 2 resemble concentric rings, reflecting the quadratic dependence of target to the observations at this sensor. As it can be seen, the axis-parallel partitions of the regression tree are not appropriate for representing either partition.

Experiments with 4 sensors. We simulated a system with 4 sensors which measured different quantities as follows $\mathbf{x}_1 = [t_1, p_1, h_1], \mathbf{x}_2 = [t_2], \mathbf{x}_3 = [p_2, h_2]$ and $\mathbf{x}_4 = [t_3, p_3]$ [1].

We assigned different numbers of codewords to different quantizers, $M_1 = M_4 = 8, M_2 = 2, M_3 = 4$. The target variable y was generated as $y = \sum_{i=1}^{3}(t_i^2/3) + \sum_{i=1}^{2}(h_i^2/3) + \sum_{i=1}^{3} 2p_i + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.25)$. Estimation of y was made using the fusion function h for which we considered: 1) the lookup table method where h is 4-dimensional with 512 cells and 2) the Direct Sum method (Gubner, 1993) with coefficients $a_i, b_j, c_k, d_l, i = 1, \ldots, 8, j = 1, 2, k = 1, \ldots, 4$ and $l = 1, \ldots, 8$.

As we can observe from Table 4, DSLVQ2 is consistently and significantly more accurate than LVQ2 and Regression Tree algorithm. It is more accurate than SPQ and DA when the budget is large and as accurate as SPQ and DA when the budget is small. The decrease in SPQ and DA performance is due to the fact that sensors are of different type, meaning that they require different σ_s^2 parameters. This leads to an increase in the number of parameters. In addition, DA and SPQ training comes at the price of very large computational effort, due to extensive parameter search as the sensors measure different quantities. Training of DSLVQ2 was the fastest overall. Finally, the Direct Sum decoder was superior to the lookup table decoder.

Experiments with AOD data. Let us first describe the instruments that were used for data collection, mainly AERONET and MODIS.

AERONET is a global network of highly accurate ground-based instruments that observe aerosols. They are densely situated in industrialized areas and sparsely located elsewhere. For our evaluation purposes we considered 37 AERONET instruments located in North America (Holben et al., 1998).

MODIS, aboard NASA's Terra and Aqua satellites, is an instrument for satellite-based AOD retrieval (Kaufman et al., 1992) that provides global coverage with a moderately accurate AOD retrieval estimated from multispectral images collected by MODIS.

The characteristics of MODIS- and AERONET-based AOD retrieval are quite different. AERONET retrievals consist of providing a single continuous measurement (AOD retrieval) many times a day but only at instrument location. On the other hand MODIS achieves an almost complete global coverage daily, providing multivariate observations extracted from multispectral images (e.g. reflectance, azimuth, etc.) in form of a feature vector that serves as an input to NASA's currently operational MODIS retrieval algorithm (C005) (R, 2006). C005 provides AOD retrieval predictions that are considered to be of moderate accuracy. It is typically outperformed by more sophisticated methods such as neural networks (Radosavljevic et al., 2010; Ristovski et al., 2012).

Data collected using AERONET served as our ground truth y measurement, while MODIS measurements at distributed locations collocated with the corresponding AERONET site served as our $\mathbf{x}$ measurements. The collocation of the AERONET and the MODIS data involved aggregating MODIS observations into blocks of 10 km $\times$ 10 km around each AERONET site. In our setup, 9

[1] $t_i \sim \mathcal{N}(\mu_t, \Sigma_t), h_i \sim \mathcal{N}(\mu_h, \Sigma_h), p_i \sim \mathcal{N}(\mu_p, \Sigma_p), \mu_t = \mu_p = (0;0;0), \mu_h = (0;0), \Sigma_t = (1.03, 0, 0; 0, 1.03, 0; 0, 0, 1.03), \Sigma_h = (1.25, 0.625; 0.625, 1.25), \Sigma_p = (1.6, 1.2, 1.2; 1.2, 1.6, 1.2; 1.2, 1.2, 1.6)$.

Table 5
3×3 grid AOD data performance comparison, Decoder: Direct Sum, $M = 4$.

Data type	Algorithm	Number of training sessions	MSE	R^2
Original data	Neural network	10	**.0058**	**.7952**
	C005	1	.0094	.6698
Encoded data	DA	256	.0098	.6564
	SPQ	16	.0087	.6949
	Reg.Tree	1	.0111	.6097
	LVQ2	1	.0187	.3443
	DSLVQ2	1	**.0075**	**.7370**

closest blocks to each AERONET site were considered, i.e. 9 21-dimensional feature vectors, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_9$. The features constructed using MODIS observations are said to be temporally collocated with the corresponding AERONET AOD retrievals if there is a valid AERONET AOD retrieval within a one-hour window centered at the satellite overpass time. The data collocated in this way are obtained from the official MODIS Web site of NASA (R, 2006).

The goal was to design 9 encoders with $M = 4$ codewords aboard satellite, and a decoder to be used at any ground location. Once the encoders and the decoder are trained, the satellite no longer needs to transmit 9×21 continuous measurements for AOD prediction at specific ground locations. Instead, only 9 discrete variables with cardinality $M = 4$ are communicated.

For our study, we collected 2,210 collocated observations distributed over 37 North America AERONET sites during year 2005. A total of 1,694 examples at randomly selected 28 AERONET sites were used as training data, while 516 examples at remaining 9 AERONET sites were used as test data. This procedure ensured that the resulting model was tested at AERONET locations unobserved during training, which gave us some insight into model performance at any North America ground location.

Table 5 compares prototype-based and Regression Tree algorithms in terms of *MSE* and R^2, defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - h(\alpha_1(\mathbf{x}_{1i}), \alpha_2(\mathbf{x}_{2i})))^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \quad (15)$$

where $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$.

Budget was set to $P = 100$, and estimation of y was made using the Direct Sum fusion function h. Alternative fusion function look-up table method was highly inefficient due to large number of table cells. The decentralized estimation results were in addition compared to C005 and Neural Network algorithms that used original features in absence of communication constraints. A Neural Networks with 5 hidden layers was trained using resilient propagation optimization algorithm for 300 epochs.

It can be observed that DSLVQ2 outperformed the competing methods. Using additional parameters in SPQ and DA was not beneficial, probably due to large number of sources. Very low values of AOD retrievals might also have effected the accuracy of SPQ and DA (y had a mean value of 0.14 and ranged from 0.006 to 1).

In addition, DSLVQ2 had better accuracy than NASA's C005 algorithm and performed fairly close to NN, while achieving significant communication cost savings. Each satellite message for a single ground location requires 9×2 bits with DSLVQ2, requires 9×504 bits with NN, which is a significant difference, especially when we consider that these measurements are taken all year round.

5. Discussion

Overall, the most appealing features of DSLVQ2 are ease of implementation, training speed, and insensitivity to parameter selection. DSLVQ2 training is much faster than DA and SPQ because it does not require annealing. On the accuracy side, DSLVQ2 is superior to LVQ2 and Regression Trees, it is comparable to more expensive and difficult to tune SPQ and DA algorithms in scenarios with small number of sensors, while it becomes superior to SPQ and DA when the number of sensors and sensor measurements increases.

6. Conclusion

In this paper we addressed the problem of data-driven quantizer design in decentralized estimation. We proposed DSLVQ2, a simple algorithm that was shown to be successful in multi-type, multi-sensor environments. DSLVQ2 exhibits better performance than previously proposed LVQ2 and Regression Tree algorithm. It also outperforms the state of the art DA algorithm in applications with large number of sensors and sensor measurements, while being less sensitive to parameter selection and orders of magnitude cheaper to train.

References

Fang, J., Li, H., 2010. Distributed estimation of GaussMarkov random fields with one-bit quantized data. IEEE Signal Process. Lett. 17 (5), 449–452.

Grbovic, M., Vucetic, S., 2009. Decentralized estimation using learning vector quantization. Data Compress. Conf., 446.

Gubner, J.A., 1993. Distributed estimation and quantization. IEEE Trans. Inf. Theory 39 (3), 1456–1459.

Holben, B.N., Eck, T.F., Slutsker, I., Tanr, D., Buis, J.P., Setzer, A., Vermote, E., Reagan, J.A., Kaufman, Y.J., Nakajima, T., Lavenu, F., Jankowiak, I., Smirnov, A., 1998. AERONET: A federated instrument network and data archive for aerosol characterization. Remote Sensing Environ. 66 (1), 1–16.

Kaufman, Y., Menzel, W.P., Tanre, D., 1992. Remote sensing of cloud, aerosol, and water vapor properties from the Moderate Resolution Imaging Spectrometer (MODIS). IEEE Trans. Geosci. Remote Sensing 30 (1), 2–27.

Kohonen, T., 1990. The Self-organizing Map. Proc. IEEE 78, 1464–1480.

Lam, W.M., Reibman, A.R., 1993. Design of quantizers for decentralized estimation systems. IEEE Trans. Comm. 41 (11), 1602–1605.

Li, H., 2007. Distributed adaptive quantization and estimation for wireless sensor networks. Signal Process. Lett. 14 (10).

MacQueen, J.B., 1967. Some methods for classification and analysis of multi- variate observations. Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, 281–297.

Megalooikonomou, V., Yesha, Y., 2000. Quantizer design for distributed estimation with communication constraints and unknown observation statistics. IEEE Trans. Comm. 48 (2).

Nguyen, X.L., Wainwright, M.J., Jordan, M.I., 2005. Nonparametric decentralized detection using kernel methods. IEEE Trans. Signal Process. 53 (11), 4054–4066.

Remer, L.A., Tanr, D., Kaufman, Y.J., Levy, R., Mattoo, S. Algorithm for remote sensing of tropospheric aerosol from MODIS: Collection 005. MODIS Algorithm Theoretical Basis Document ATBDMOD-04, 2006.

Radosavljevic, V., Vucetic, S., Obradovic, Z., 2010. A data-mining technique for aerosol retrieval across multiple accuracy measures. IEEE Geosci. Remote Sensing Lett. 7 (2), 411–415.

Rao, A., Miller, D., Rose, K., Gersho, A., 1996. A generalized VQ method for combined compression and estimation. Internat. Conf. Acoust. Speech Signal Process, 2032–2035.

Ristovski, K., Vucetic, S., Obradovic, Z., 2012. Uncertainty analysis of neural network-based aerosol retrieval. IEEE Trans. Geosci. Remote Sensing 50 (2), 409–414.

Seo, S., Bode, M., Obermayer, K., 2003. Soft nearest prototype classification. Trans. Neural Networks 14, 390–398.

Wang, T.Y., Chang, L.Y., Chen, P.Y., 2009. Collaborative sensor-fault detection scheme for robust distributed estimation in sensor networks. Trans. Comm. 57 (10), 3045–3058.

Xiao, J.J., Luo, Z.Q., 2005. Universal decentralized detection in a bandwidth-constrained sensor network. IEEE Trans. Signal Process. 53 (8), 2617–2624.

Xiao, J.J., Cui, S., Luo, Z.Q., Goldsmith, A.J., 2008. Linear coherent decentralized estimation. IEEE Trans. Signal Process. 56 (2), 757–770.