# DISTRIBUTED ROBUST SUBSPACE TRACKING

*Yannis Kopsinis*[1] *Symeon Chouvardas*[2] *Sergios Theodoridis*[1]

[1]University of Athens,
Dept. of Informatics and Telecommunications,
Athens 15784, Greece.
Emails: kopsinis@ieee.org, stheodor@di.uoa.gr

[2]Mathematical and Algorithmic Sciences Lab,
Huawei France R&D,
Paris, France.
Email: symeon.chouvardas@huawei.com

## ABSTRACT

In this paper, a distributed, set-theoretic based subspace tracking scheme is presented. In particular, each one of the agents in the network has access to a subset of data, which are not allowed to be shared among them. Moreover, the data vectors lie on a low–rank linear subspace, which is unknown and it might also be time-varying. The agents aim at estimating and tracking the unknown subspace using solely their own data and the tentative subspace estimates of their neighbours. Moreover, some of the the data might be corrupted with outlier noise. Method is evaluated in a synthetic simulation example, where the unknown subspace exhibits abrupt changes.

***Index Terms***— Distributed online learning, Robust Subspace Tracking, APSM.

## 1. INTRODUCTION

The volume of data captured worldwide is growing at an exponential rate posing certain challenges regarding their processing and analysis. Moreover, taking into account that the available data exhibit an increased complexity (new types of data emerge) and they are also involved in advanced architectures according to the *Internet of Things* [1] paradigm, it is clear that further advances in already established machine learning techniques are required in order to cope with the new challenges.

Even though data tend to live in high dimensional spaces, they often exhibit a high degree of redundancy; that is, their useful information can be represented using a number of attributes much lower compared to their original dimensionality. This is a key attribute, which allows the efficient analysis and processing of such data, provided that the low dimensional subspace, which they lie on, has been estimated accurately enough. The most common tool for subspace estimation, dimensionality reduction and data analysis is the celebrated Principal Component Analysis (PCA). This technique, especially when dealing with high dimensional data, has a number of drawbacks mostly related to the fact that it operates in a batch mode. More specifically, all the available data has to be stored leading to increased memory requirements and high computational load. On top of that, in big data applications, the data might not be able to be stored in the first place. In such cases, the algorithm needs to retrieve them from slow memory devices such as hard disks or to access them over a network. This procedure implies excessive delays due to communication costs. Another disadvantage due to batch operation is that the unknown subspace has to be re–computed from scratch whenever a new datum becomes available.

Online/sequential processing offers significant benefits regarding memory requirements and computational complexity compared to the batch mode of operation. Moreover, in many occasions the subspace is subjected to changes as time evolves, [2]. In such scenarios, the employed subspace estimation algorithm needs to track the subspace online; *i.e,* to update its current estimate based on the data which become available sequentially, one per time instance. Online Subspace Tracking (ST), *e.g.,* [3, 4, 5, 6], plays a central role in many applications, such as, tracking of moving objects [7], foreground/background separation [8], beamforming [9], just to name a few.

In order to tackle potential limitations of processing power and/or of storage capacity, one may also consider to split the full data analysis problem into subtasks in order to distribute it among a number of processing units. This philosophy has been followed in several methods in the framework of MapReduce/Hadoop [10, 11] involving the fusion of the subproblem outcomes in a central processor, which need to communicate with all the subunits. An alternative path is to resort to fully distributed/decentralized solutions based on recent advances, which stem from the signal processing and machine learning communities [12, 13, 14, 15]. The latter approach offers certain advantages. First, the existence of a fusion center is avoided and one solely relies on in–network processing in ad-hoc topologies. This leads to increased reliability and robustness of the network system, because it is not affected by possible fusion center failures. Overall, besides the computational and storage ease per processing unit, which can be attained by distributed processing, another important attribute is that of privacy. In particular, the processing of data is performed locally avoiding the need for sensitive information exchange [16]. Finally, in many ST applications (see for example [8, 5]) the data set includes outliers; that is corrupted data that do not adhere to the adopted model.

This work is partly supported by Marie Curie IEF, "SOL", 302898. This work is cofinanced by the European Union (European Social Fund) and Greek national funds through the operational program Education and Lifelong Learning of the National Strategic Reference Framework (NSRF). Research funding program THALES: investing in knowledge society through the European Social Fund.

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

For example, when the data are exchanged within nodes in a wireless sensor network outliers may occur due to malfunctioning nodes. If the presence of outliers is not taken into consideration, then the performance of the ST algorithms can be seriously degraded, *e.g.,* [8]. Hence, robustness against outliers is a matter of paramount importance in subspace tracking. Outliers are also present in a number of big data applications.

Recently, an algorithm for robust subspace tracking was proposed [17]. The presented scheme belongs to the family of the Adaptive Projected Subgradient Method (APSM) algorithms, *e.g.,* [18, 19, 20, 15]. In this paper, a distributed version of the aforemention algorithm is presented. According to the APSM rationale, at each node and in any particular time instance, a loss function is defined around the most recently obtained data vector. This loss function scores a zero for a non–empty (property) set of points/possible solutions. All these points are candidate solutions. Such a philosophy is in line with arguments related to robust statistics costs. The goal of the method is to find a point (solution), which belongs to the intersection of all these property sets associated with the received data.

The nodes of the network cooperate with each other by exchanging information and fusing it with respect to the diffusion rationale, adopted in distributed adaptive filtering task, e.g., [21]. Furthermore, our proposed algorithm identifies the time instances at which the data contain outlier noise. In this case, a sparsity–promoting greedy technique, namely the Compressed Sampling Orthogonal Matching Pursuit (CoSAMP), [22], is mobilized to estimate the sparse outlier vector, and removes it from the data vector (in principle any sparsity promoting algorithm can be adopted). Finally, our algorithm attacks the missing entries scenario, by employing a technique, which attempts to predict the missing entries.

*Notation:* The set of real numbers and the set of non–negative integers are denoted by $\mathbb{R}$ and $\mathbb{N}$ respectively. Matrices are denoted by uppercase boldface letters and vectors by boldface letters. $(\cdot)^T$ stands for vector/matrix transposition. Moreover, $\|\cdot\|$ stands for the Euclidean norm and $\|\cdot\|_F$ for the Frobenius norm. Finally, the $m \times r$ zero matrix and the $m \times 1$ zero vector are denoted by $\boldsymbol{O}_{m\times r}$ and $\boldsymbol{0}_m$ respectively.

## 2. PROBLEM STATEMENT

Consider a collection of agents, which are capable of performing computations locally. Moreover, each agent has a number of neighbors; its neighborhood comprises all agents, which it is connected to and it can exchange information with. Such a network of agents can be modeled as an undirected graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, \ldots, K\}$ stands for the set of all the nodes (each node representing an agent) and $\mathcal{E}$ is the set of pairs of nodes, which are neighbors. Of special interest are the strongly connected networks, in which there exists at least one (possibly multihop) path connecting every two nodes of the network. Such a network is illustrated in Fig. 1.

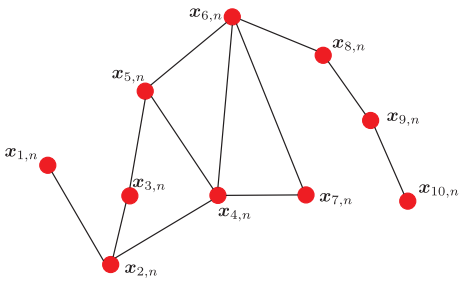


**Fig. 1**. Illustration of an ad–hoc network.

Each node, $k \in \mathcal{N}$, has access to a set of observation vectors $(\boldsymbol{x}_{k,n}), \boldsymbol{x}_{k,n} \in \mathbb{R}^m$, where $n$ denotes successive time instances, $n = 1, \ldots, N$. The observed vectors are generated via the following model:

$$\boldsymbol{x}_{k,n} = \boldsymbol{U}_*^{(n)} \boldsymbol{w}_{k,n} + \boldsymbol{v}_{k,n} + \boldsymbol{o}_{k,n}, \ \forall n \in \mathbb{N} \qquad (1)$$

where $\boldsymbol{U}_*^{(n)}$ is an $m \times r$, $m > r$ orthonormal matrix, $\boldsymbol{w}_{k,n} \in \mathbb{R}^r$, $\boldsymbol{v}_{k,n} \in \mathbb{R}^m$ are vectors corresponding to additive noise and $\boldsymbol{o}_{k,n}$ is either $\boldsymbol{0}_m$ or corresponds to an outlier, *i.e.,* $\boldsymbol{o}_{k,n} = \boldsymbol{s}_{k,n} \in \mathbb{R}^m$. Following a similar rationale as in [8, 5], we assume that the outlier vectors are sparse, that is, $\|\boldsymbol{s}_{k,n}\|_0 \ll m$, where with $\|\cdot\|_0$ we denote the $\ell_0$ pseudo-norm.

Assume for the moment that matrix $\boldsymbol{U}_*^{(n)}$ remains fixed, $\forall n$. Then, in the absence of noise and outliers, it is clear that the observed vectors, $\boldsymbol{x}_{k,n}, \forall k, n$, lie on the $r$-dimensional subspace, which is spanned by the column vectors of matrix $\boldsymbol{U}_*^{(n)}$. All the nodes of the network share a common goal; that of estimating the subspace where the noise–free observations lie on. In particular, the nodes are requested to estimate a *common* matrix $\hat{\boldsymbol{U}}_*^{(n)} \in \mathbb{R}^{m\times r}$, whose column space is the unknown subspace. When $\boldsymbol{U}_*^{(n)}$, changes with time, $n$, then the common objective of all nodes is to track these changes effectively. Even though, each agent has not access to the whole data set, cooperation among the agents can be beneficial in terms of improved performance. Intuitively, each node is benefited via cooperating with its neighbours, which in turn have already been benefited via their cooperation with their own neighbours. It turns out that in certain applications, such as distributed sparse regression [14], the solution which would be obtained coincides to the one reached in the distributed setting, i.e., solely relying on local measurements and on in-network processing. The theoretical aspects in the subspace tracking scenario, which is studied in this paper, are much harder to tackle due to the fact that each node aims at solving a non–convex problem. We are currently working on a proof regarding consensus in a local minima.

### 2.1. Projection Approximation Subspace Tracking

Many popular subspace tracking techniques are based on the Projection Approximation Subspace Tracking (PAST) algo-

rithm [3]. Let us, for simplicity, switch to the centralized processing approach for the moment denoting the observations as $\boldsymbol{x}_n$ rather than $\boldsymbol{x}_{k,n}$. Motivated by the exponentially weighted approach, central in the Recursive Least Squares (RLS) algorithm, online processing and tracking abilities are assigned to the PAST algorithm, which aims at minimizing the following loss function

$$J^{(n)}(\boldsymbol{U}) = \sum_{i=1}^{n} \beta^{n-i} \|\boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{x}_i\|^2, \tag{2}$$

$0 < \beta \leq 1$ is the so called forgetting factor, used in non–stationary environments, where the subspace undergoes changes. Moreover, a further simplification is adopted aiming to convexify (2). In particular:

$$J^{(n)}(\boldsymbol{U}) = \sum_{i=1}^{n} \beta^{n-i} \|\boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{y}_i\|^2, \tag{3}$$

where

$$\boldsymbol{y}_i = \boldsymbol{U}_{i-1}^T \boldsymbol{x}_i, \tag{4}$$

*i.e.,* one of the unknown parameter matrices, $\boldsymbol{U}$, is replaced by the respective tentative estimates, $\boldsymbol{U}_{i-1}^T$.

The cost function given in (3) is the exponential Least Squares criterion, which has been extensively studied in adaptive filtering, *e.g.,* [9]. The matrix $\boldsymbol{U}_{LS}^{(n)}$, that minimizes the cost (3) at time instance $n$, is given by [3]:

$$\boldsymbol{U}_{LS}^{(n)} = \boldsymbol{C}_{xy}(n)\boldsymbol{C}_{yy}^{-1}(n), \tag{5}$$

where

$$\boldsymbol{C}_{xy}(n) = \beta \boldsymbol{C}_{xy}(n-1) + \boldsymbol{x}_n \boldsymbol{y}_n^T, \tag{6}$$

and

$$\boldsymbol{C}_{yy}(n) = \beta \boldsymbol{C}_{yy}(n-1) + \boldsymbol{y}_n \boldsymbol{y}_n^T. \tag{7}$$

Having access to the quantities $(\boldsymbol{C}_{xy}(n), \boldsymbol{C}_{yy}(n))_{n\in\mathbb{N}}$, the classical PAST algorithm employs the Recursive Least Squares (RLS) Method for the estimation of $\boldsymbol{U}_{LS}^{(n)}$, by computing efficiently the matrix $\boldsymbol{C}_{yy}^{-1}(n)$.

## 3. DISTRIBUTED SUBSPACE TRACKING VIA THE APSM ALGORITHM

Recently, an algorithm for robust subspace tracking was proposed [17]. This algorithm was built upon the convexification rational of PAST in order to formulate a set-theoretic based estimation scheme in accordance to the family of the Adaptive Projected Subgradient Method (APSM) algorithms, *e.g.,* [18, 19, 20]. Moreover, an efficient outlier detection and cleansing technique was proposed as well. Next, a distributed extension of this algorithm is presented:

The PAST algorithm computes, at each step, the matrix $\boldsymbol{U}_{LS}^{(n)}$ by solving (5). The LS solution, which is sought via the time averaged covariance matrices $(\boldsymbol{C}_{xy}(n), \boldsymbol{C}_{yy}(n))$, is likely to deviate from the true solution, *i.e.,* from the true subspace, due to a number of reasons such as: additive noise, measurement and model inaccuracies, as well as calibration errors. In order to accommodate such deviations, following set theoretic arguments, we seek the unknown subspace within an "extended" set of possible solutions, which guarantee to include the true one, or at least, to include it with high probability. To be more specific, given a certain tolerance $\epsilon > 0$, for each node, we define the following loss function

$$\Theta_{k,n} : \mathbb{R}^{m\times r} \to [0, +\infty) : \boldsymbol{U} \mapsto \max\left\{0, \frac{1}{2}\|\boldsymbol{C}_{xy}(k,n) - \boldsymbol{U}\boldsymbol{C}_{yy}(k,n)\|_F^2 - \epsilon\right\}, \tag{8}$$

where

$$\boldsymbol{C}_{xy}(k,n) = \beta \boldsymbol{C}_{xy}(k,n-1) + \boldsymbol{x}_{k,n}\boldsymbol{y}_{k,n}^T, \tag{9}$$

$$\boldsymbol{C}_{yy}(k,n) = \beta \boldsymbol{C}_{yy}(k,n-1) + \boldsymbol{y}_{k,n}\boldsymbol{y}_{k,n}^T. \tag{10}$$

The quantity $\boldsymbol{y}_{k,n}$ can be estimated, either as in the PAST algorithm, i.e., (4) or via the pseudo–inverse of the matrix $\boldsymbol{U}_{k,n-1}$, i.e., $\boldsymbol{y}_{k,n} = (\boldsymbol{U}_{k,n-1}^T\boldsymbol{U}_{k,n-1})^{-1}\boldsymbol{U}_{k,n-1}^T\boldsymbol{x}_{k,n}$. This latter option has been proposed in [6] and we can confirm, based on an extensive set of simulations, that it results in enhanced performance.

For each one of the loss functions, we consider the corresponding level set, defined as: $\mathrm{lev}_{\leq 0}\Theta_{k,n} := \{\boldsymbol{U} \in \mathbb{R}^{m\times r} : \Theta_{k,n}(\boldsymbol{U}) \leq 0\}$. Notice that $\boldsymbol{U}_{LS}^{(n)} \in \mathrm{lev}_{\leq 0}\Theta_{k,n}$, hence the level–set at each time instance is an "enlarged" set of candidate solutions, since it contains every matrix, which scores a zero loss, instead of containing a single point, which is the case in the Least Squares cost in the PAST algorithm.

The proposed algorithm, is based on the set theoretic estimation approach; the goal is to compute a point in the intersection of all the previously defined level sets. This can be effectively achieved via the Adaptive Projected Subgradient Method formula, *e.g.,* [19, 23, 18, 20], given next:

$$\tilde{\boldsymbol{U}}_{k,n} = \begin{cases} \boldsymbol{U}_{k,n-1} - \lambda_{k,n} & \frac{\Theta_{k,n}(\boldsymbol{U}_{k,n-1})}{\|\Theta'_{k,n}(\boldsymbol{U}_{k,n-1})\|_F^2}\Theta'_{k,n}(\boldsymbol{U}_{k,n-1}), \\ & \Theta'_{k,n}(\boldsymbol{U}_{k,n-1}) \neq \boldsymbol{O}_{m\times r} \\ \boldsymbol{U}_{k,n-1}, & \Theta'_{k,n}(\boldsymbol{U}_{k,n-1}) = \boldsymbol{O}_{m\times r} \end{cases} \tag{11}$$

where $\Theta'_{k,n}$ stands for any subgradient, which belongs to the subdifferential $\partial\Theta_{k,n}$, defined as [24]:

$$\partial\Theta_{k,n}(\boldsymbol{U}) = \begin{cases} \boldsymbol{O}_{m\times r}, & \frac{1}{2}\|\boldsymbol{C}_{xy}(k,n) - \boldsymbol{U}\boldsymbol{C}_{yy}(k,n)\|_F^2 < \epsilon, \\ \gamma(\boldsymbol{U}\boldsymbol{C}_{yy}(k,n) - \boldsymbol{C}_{xy})\boldsymbol{C}_{yy}^T(k,n),\ \gamma \in [0,1] & \\ & \frac{1}{2}\|\boldsymbol{C}_{xy}(k,n) - \boldsymbol{U}\boldsymbol{C}_{yy}(k,n)\|_F^2 = \epsilon, \\ (\boldsymbol{U}\boldsymbol{C}_{yy}(k,n) - \boldsymbol{C}_{xy})\boldsymbol{C}_{yy}^T(k,n), & \\ & \frac{1}{2}\|\boldsymbol{C}_{xy}(k,n) - \boldsymbol{U}\boldsymbol{C}_{yy}(k,n)\|_F^2 > \epsilon, \end{cases} \tag{12}$$

and $\lambda_{k,n} \in (0,2)$.

It should also be stressed that at each time step the APSM recursion, given in (11), is allowed to be repeated multiple times (namely $M$ times). Obviously, this increases the complexity of the algorithm, albeit it leads to enhanced convergence speed and tracking agility.

Our aim, now, is to effectively combine the tentative estimates in each node, $\tilde{\boldsymbol{U}}_{k,n},\ \forall k$ in a way that fuses the local information of the neighboring nodes in order to lead to improved estimates. According to the diffusion distributed learning rationale [21], this can be achieved via the following convex combination

$$\boldsymbol{U}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{l,k} \tilde{\boldsymbol{U}}_{l,n}, \tag{13}$$

where $\mathcal{N}_k$ is the neighborhood of node $k$, i.e., the set of nodes with which it can exchange information and $a_{l,k}$ are combination coefficients computed via several adequate rule, such as the Metropolis rule:

$$a_{l,k} = \begin{cases} \frac{1}{\max\{|\mathcal{N}_k|,|\mathcal{N}_l|\}}, & \text{if } l \in \mathcal{N}_k \text{ and } l \neq k, \\ 1 - \sum_{l \in \mathcal{N}_k \setminus k} a_{l,k}, & \text{if } l = k, \\ 0, & \text{otherwise.} \end{cases}$$

Note that $a_{l,k} > 0$, if $l \in \mathcal{N}_k$, $a_{l,k} = 0$, if $l \notin \mathcal{N}_k$ and $\sum_{l \in \mathcal{N}_k} a_{l,k} = 1, \forall k \in \mathcal{N}$. Furthermore, it is assumed that each node is a neighbor of itself, *i.e.,* $a_{k,k} > 0,\ \forall k \in \mathcal{N}$.

It should be stressed that the combination step (13), encourages all neighbours to find the same matrix $\boldsymbol{U}$. This request, however, might be considered quite restrictive since our primary goal is not the nodes to estimate the same matrix but the same subspace. We are currently working towards a modification of the algorithm proposed here, where this constrain is relaxed and each node is allowed to estimate a different $\boldsymbol{U}$ matrix, whereas all these matrices are encouraged to span the same subspace.

### 3.1. Robustness against outliers

The algorithm, as it has been proposed up to now, does not account for outliers, $\boldsymbol{s}_{k,n}$, that may contaminate the observed vectors at the nodes. In [17], a two stage approach was proposed. First, a mechanism for deciding whether an outlier is present or not is employed; this is based on the error residuals $||\boldsymbol{x}_{k,n} - P_{U_{n-1}}\boldsymbol{x}_n||^2 = ||\boldsymbol{r}_{k,n}||^2$, where $P_{\boldsymbol{U}}\boldsymbol{x}$ denotes the projection of $\boldsymbol{x}$ onto the column space of $\boldsymbol{U}$ and $\boldsymbol{r}_{k,n} = \boldsymbol{x}_{k,n} - (\boldsymbol{U}_{k,n-1}^T\boldsymbol{U}_{k,n-1})^{-1}\boldsymbol{U}_{k,n-1}^T\boldsymbol{x}_{k,n}$. In particular, a a threshold need to be set and if it is exceeded by the current error residual then the presence of an outlier is identified. Second, in case of an outlier presence, the CoSaMP, [22], sparsity promoting algorithm is employed, which provides estimates of the outlier vectors, $\hat{\boldsymbol{s}}_{k,n}$. The same method will be used in the distributed learning scenario discussed here, by applying it to each node separately and it is not repeated here due to space limitations. As long as estimates of the outlier vectors are available, the observed data vectors $\boldsymbol{x}_{k,n}$ can be cleansed from the outlier's effects by subtracting $\hat{\boldsymbol{s}}_{k,n}$ from $\boldsymbol{x}_{k,n}$ before its use for the computation of $\boldsymbol{y}_{k,n}$ and $\boldsymbol{C}_{xy}(k,n)$.

The steps of the proposed algorithm, which is hereafter referred to as Distributed Subspace Tracking based on Adaptive Projection Subgradient Method (DSTAPSM), are given in detail in Table 1, where $N$ is the number of data vectors per node. Note that the threshold, $r_{k,n}$, which determines whether an outlier is present or not, is parameterised as $\delta\bar{r}_{k,n}$, where is the average of the $q$ most recent error residuals, $q$ is a user–defined parameter and $\delta$ is a user–defined multiplication factor. Apparently, the larger the $\delta$ is the less sensitive the algorithm becomes in the detection of outlier vectors. Moreover, in the case that we know that every data vector is corrupted by an outlier vector, then $\delta$ should be set equal to zero. The reason why the threshold is computed via an average over the $q$ most recent error residuals, instead of the whole history, is in order "forget" past values allowing operation in non-stationary environments Note that, one could adopt more sophisticated outlier detection mechanisms, *e.g.,* [8], or taking into account the statistics of the residual or the apriori information that the outlier vector is sparse. However, even the simple outlier detection approach discussed above appears to perform well enough. So, in this paper, we adhere to this scenario in order to give emphasis to other, more important aspects of the proposed methodology.

**Table 1**.

**Distributed Subspace Tracking Adaptive Projected Subgradient Method**

**Initialization:** An $m \times r$ random matrix $\boldsymbol{U}_{k,0}$, $\epsilon > 0$, window length $q$, $\delta > 0$.

**FOR** $n = 1 : N$ **DO**
**FOR** $k = 1 : K$ **DO**
1: $\boldsymbol{r}_{k,n} = \boldsymbol{x}_{k,n} - \boldsymbol{U}_{k,n-1}(\boldsymbol{U}_{k,n-1}^T\boldsymbol{U}_{k,n-1})^{-1}\boldsymbol{U}_{k,n-1}^T\boldsymbol{x}_{k,n}$
**IF** $\|\boldsymbol{r}_{k,n}\|^2 =: r_{k,n} \leq \delta\bar{r}_{k,n-1}$
2: $\hat{\boldsymbol{s}}_{k,n} = \boldsymbol{0}_m$
**ELSE**
3: Estimate $\hat{\boldsymbol{s}}_{k,n}$ as in [17]
**ENDIF**
4: $\boldsymbol{y}_{k,n} = (\boldsymbol{U}_{k,n-1}^T\boldsymbol{U}_{k,n-1})^{-1}\boldsymbol{U}_{k,n-1}^T(\boldsymbol{x}_{k,n} - \hat{\boldsymbol{s}}_{k,n})$
5: $\boldsymbol{C}_{xy}(k,n) = \beta\boldsymbol{C}_{xy}(k,n-1) + (\boldsymbol{x}_{k,n} - \hat{\boldsymbol{s}}_{k,n})\boldsymbol{y}_{k,n}^T$
6: $\boldsymbol{C}_{yy}(k,n) = \beta\boldsymbol{C}_{yy}(k,n-1) + \boldsymbol{y}_{k,n}\boldsymbol{y}_{k,n}^T$
7: $\tilde{\boldsymbol{U}}_{k,n} = \begin{cases} \boldsymbol{U}_{k,n-1} - \lambda_{k,n}\frac{\Theta_{k,n}(\boldsymbol{U}_{k,n-1})}{\|\Theta_n'(\boldsymbol{U}_{k,n-1})\|_F^2}\Theta_n'(\boldsymbol{U}_{k,n-1}), & \Theta_n'(\boldsymbol{U}_{k,n-1}) \neq \boldsymbol{O}_{m\times r} \\ \boldsymbol{U}_{k,n-1}, & \Theta_n'(\boldsymbol{U}_{k,n-1}) = \boldsymbol{O}_{m\times r} \end{cases}$
8: $q' = \max\{0, n-q\}$, $\bar{r}_{k,n} = \frac{1}{n-q'}\sum_{j=q'}^{n} r_j$
**END**
**FOR** $k = 1 : K$ **DO**
9: $\boldsymbol{U}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{k,l}\tilde{\boldsymbol{U}}_{l,n}$
**END**
**END**

## 4. NUMERICAL EXAMPLES

In this section, we examine the performance of the Robust DSTAPSM in a non–stationary scenario. In particular we show that cooperation leads to performance enhancement in a network which comprises 5 nodes and 6 connections among the nodes. The adopted performance metric, is the angle between the true and the estimated subspace in logarithmic scale. We adopt the model described in (1), with $m = 100$ and $r = 20$. We assume that $\boldsymbol{U}_*^{(n)} = \boldsymbol{U}_*,\ \forall n \in \mathbb{N}$, where the columns of the $m \times r$ matrix $\boldsymbol{U}_*$ are realizations of an i.i.d. $\mathcal{N}(\boldsymbol{0}_m, \boldsymbol{I}_m)$ and then, they are getting orthornomalized. The coefficients of the vector $\boldsymbol{w}_{k,n}$ and the noise $\boldsymbol{v}_{k,n}$ are drawn from the Gaussian distribution with zero mean and variance equal to $\sigma_w^2 = 1$ and $\sigma_v^2 = 10^{-3}$ respectively. In order to examine the tracking ability of the proposed scheme at time

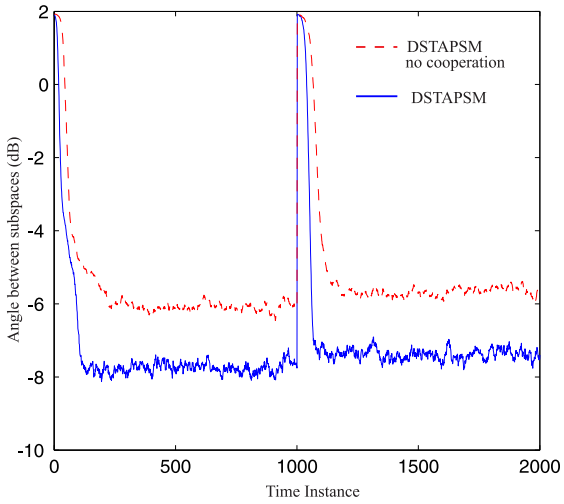


**Fig. 2**. Angle between the subspaces: the non–stationary case

instance 1000 the true subspace is changed abruptly.

Finally, we assume that $10\%$ of the data vectors are contaminated with outliers. For the sparse outlier vector, we have that $\|\boldsymbol{s}_n\|_0 = 5$ and its non–zero coefficients follow the Gaussian distribution with zero mean and standard deviation equal to 5. The positions of the non–zero coefficients of $\boldsymbol{s}_n$ and the time steps, on which the outliers occur, are selected randomly.

For the proposed algorithm, we set $\epsilon = 2 \times \sigma_v^2$, $\beta = 0.9$, $q = 20$, $M = 10$, $\delta = 3$ and $\lambda_n = 1$. Regarding the step–size $\lambda_{k,n}$, extensive experimentation, indicated that the larger the $\lambda_{k,n}$ the faster the convergence, at the expense of a higher error floor. Choosing $\lambda_{k,n} = 1$ leads to a good trade–off between convergence speed and steady–state error floor. Increasing the parameter $M$ results in accelerated convergence speed, which comes at the cost of a higher complexity. It is assumed that we have an estimate of the number of non–zero coefficients of the vector $\boldsymbol{s}_n$, employed in the CoSaMP algorithm. Finally, the number of iterations utilized in the CoSaMP for the outlier vector estimation, equals to 30.

The performance results are depicted in Fig. 2 where the no cooperation refers to the average of the performances of all the nodes when each one of them estimates the unknown subspace solely relying on its own data. It is apparent that cooperation leads to significantly enhanced performance.

## REFERENCES


[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] Tülay Adali and Simon Haykin, *Adaptive signal processing: next generation solutions*, vol. 55, John Wiley & Sons, 2010.

[3] Bin Yang, "Projection approximation subspace tracking," *Signal Processing, IEEE Transactions on*, vol. 43, no. 1, pp. 95–107, 1995.

[4] Morteza Mardani, Gonzalo Mateos, and Georgios B Giannakis, "Rank minimization for subspace tracking from incomplete data," in *In Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 5681–5685.

[5] Jun He, Laura Balzano, and John Lui, "Online robust subspace tracking from partial information," *arXiv preprint arXiv:1109.3827*, 2011.

[6] Yuejie Chi, Yonina C Eldar, and Robert Calderbank, "PETRELS: Subspace estimation and tracking from partial observations," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 3301–3304.

[7] Shahram Shahbazpanahi, Shahrokh Valaee, and Mohammad Hasan Bastani, "Distributed source localization using esprit algorithm," *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2169–2178, 2001.

[8] Gonzalo Mateos and Georgios B Giannakis, "Robust PCA as bilinear decomposition with outlier-sparsity regularization," *IEEE Transactions on Signal Processing*, vol. 60, pp. 5176–5190, 2012.

[9] A.H. Sayed, *Fundamentals of adaptive filtering*, John Wiley & Sons, New Jersey, 2003.

[10] Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun, "Map-reduce for machine learning on multicore," in *NIPS*, 2006, vol. 6, pp. 281–288.

[11] Jimmy Lin, "Mapreduce is good enough? if all you have is a hammer, throw away everything that's not a nail!," *Big Data*, vol. 1, no. 1, pp. 28–37, 2013.

[12] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[13] Ali H Sayed, "Diffusion adaptation over networks," *arXiv preprint arXiv:1205.4220*, 2012.

[14] G. Mateos, J.A. Bazerque, and G.B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, 2010.

[15] Sergios Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, Academic Press.

[16] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations*, vol. 4, no. 2, pp. 28–34, 2002.

[17] S. Chouvardas, Y. Kopsinis, and S. Theodoridis, "An adaptive projected subgradient based algorithm for robust subspace tracking," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 5497–5501.

[18] Konstantinos Slavakis and Isao Yamada, "The adaptive projected subgradient method constrained by families of quasi-nonexpansive mappings and its application to online learning," *SIAM Journal on Optimization*, vol. 23, no. 1, pp. 126–152, 2013.

[19] I. Yamada and N. Ogura, "Adaptive projected subgradient method for asymptotic minimization of sequence of nonnegative convex functions," *Numerical functional analysis and optimization*, vol. 25, no. 7&8, pp. 593–617, 2004.

[20] Sergios Theodoridis, Konstantinos Slavakis, and Isao Yamada, "Adaptive learning in a world of projections," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 97–123, 2011.

[21] Ali H Sayed, "Diffusion adaptation over networks," *Academic Press Library in Signal Processing*, vol. 3, pp. 323–454, 2013.

[22] Deanna Needell and Joel A Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301–321, 2009.

[23] K. Slavakis, I. Yamada, and N. Ogura, "The adaptive projected subgradient method over the fixed point set of strongly attracting nonexpansive mappings," *Numerical Functional Analysis and Optimization, 27*, vol. 7, no. 8, pp. 905–930, 2006.

[24] Konstantinos Slavakis, Pantelis Bouboulis, and Sergios Theodoridis, "Adaptive multiregression in reproducing kernel hilbert spaces: The multiaccess mimo channel case," *IEEE Transactions on Neural Networks Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 260–276, 2012.