# DISTRIBUTED CLASSIFICATION OF MULTIPLE OBSERVATIONS BY CONSENSUS

*Effrosyni Kokiopoulou*

Seminar for Applied Mathematics
ETHZ
Switzerland - 8092, Zurich

*Pascal Frossard*

Signal Processing Laboratory (LTS4)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Switzerland - 1015, Lausanne

## ABSTRACT

We consider the problem of distributed classification of multiple observations of the same object that are collected in an ad-hoc network of vision sensors. Assuming that each sensor captures a different observation of the same object, the problem is to classify this object by distributed processing from the sensors. We present a graph-based problem formulation whose objective function captures the smoothness of candidate labels on the data manifold. We design a distributed average consensus algorithm for estimating the unknown object class by computing the value of the above smoothness objective function for different class hypotheses. It initially estimates the objective function locally, based on the observation of each sensor. All the observations are then progressively taken into account in the estimation of the objective function, along the iterations of the distributed consensus algorithm. We illustrate the performance of the distributed classification algorithm by simulation of multi-view face recognition in an ad-hoc network of vision sensors. When the training set is sufficiently large, the simulation results show that the consensus classification decision is equivalent to the decision of a centralized system that would have access to all observations.

## 1. INTRODUCTION

Over the past few years novel multimedia architectures such as vision sensor networks have started to emerge. Typically, these networks have an ad-hoc organization: there is no central coordinator node and the topology can be arbitrary and dynamic (e.g., due to sensor motion). This presents new challenges to the analysis of multimedia data, which has to be done now distributively and efficiently, while being robust to topology changes. We consider the problem of classifying an observed object, whose multiple observations are collected in a distributed fashion (e.g., [1], [2]). Such a scenario is interesting in the context of (distributed) scene analysis or multiview recognition.

The problem then is to classify distributively the observed object *at all sensors* such that they reach a consensus decision by aggregating partial information provided by each local observation. It is important to note that this problem is different from the well-studied problem of distributed classification in the presence of a fusion center, where the information from all sensors is gathered in order to reach the final classification decision. The ultimate goal of distributed classification is to get close to the performance of a centralized classification solution having all observations in its disposal, despite the fact that the sensors only possess partial information about signals.

We first present a graph-based problem formulation that defines a smoothness criterion of candidate labels on the data manifold. The value of the smoothness objective function is computed by distributed average consensus [3]. In general, the main goal of distributed consensus is to reach a global solution iteratively in ad-hoc networks using only local computation and communication, while staying robust to changes in the network topology.

Our algorithm capitalizes on the fact that the multiple observations belong to the same class. In particular, each sensor captures an observation of the same object and computes its nearest neighbors among the labelled examples. Under a certain class hypothesis, those neighbors contribute to the local computation of a portion of the objective function value.

Those portions are summed distributively by means of average consensus, so that all observations are progressively taken into account and the total value of the objective function is eventually computed at all sensors. This process is repeated for all class hypotheses and eventually the sensors reach a consensus classification decision, by picking the class resulting in the smoothest label assignment.

We illustrate the performance of the proposed distributed algorithm in multi-view face recognition in a simulated ad-hoc network of vision sensors. When the training set is sufficiently large, the simulation results show that the consensus classification decision is equivalent to the decision of a centralized system that would have access to all observations.

## 2. PROBLEM FORMULATION

Let us formally define the problem of distributed classification of multiple observations in an ad-hoc sensor network. We consider a network of $m$ sensors and we model the network topology as an undirected graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ with nodes $\mathcal{V}_s = \{1, \ldots, m\}$ corresponding to sensors. An edge $(i, j) \in \mathcal{E}_s$ is drawn if and only if sensor $i$ can communicate with sensor $j$. We associate a weight $W(i, j)$ with each edge $(i, j) \in \mathcal{E}_s$. The matrix $W$ that gathers the edge weights $W(i, j)$ is called the weight matrix. $W$ is a sparse matrix whose sparsity pattern is driven by the network topology. We denote the set of neighbors for node $i$ as $\mathcal{N}_i = \{j |\ (i, j) \in \mathcal{E}_s\}$.

We assume that each sensor $j$ captures a single (unlabelled) observation $x_j^{(u)}$ of an object $f$. Each observation is of the form

$$x_j^{(u)} \triangleq O_j(f), \quad j = 1, \ldots, m. \tag{1}$$

and it is different from its peers. For instance, it could be a rotation of the object $f$. Hence, there are $m$ observations of the object $f$ that are collected distributively over the sensor network. All observations share the same class label due to the common dependence on $f$.

We assume further that there is a training set; hence, the whole data set can be organized in two parts $X = \{X^{(l)}, X^{(u)}\}$, with $X^{(l)} = \{x_1, x_2, \ldots, x_l\} = \{x_1^{(l)}, x_2^{(l)}, \ldots, x_l^{(l)}\} \subset \mathbb{R}^d$ and

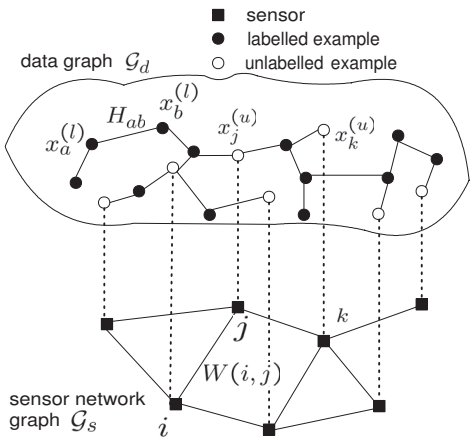



**Fig. 1**. Conceptual distinction between the two graphs of the problem. $\mathcal{G}_s$ (resp. $\mathcal{G}_d$) denotes the graph of the sensor network topology (resp. the data graph). In $\mathcal{G}_d$, the filled (resp. empty) circles correspond to labelled (resp. unlabelled) examples.

$X^{(u)} = \{x_{l+1}, \ldots, x_n\} = \{x_1^{(u)}, \ldots, x_m^{(u)}\} \subset \mathbb{R}^d$, where $n = l + m$. Let also $\mathcal{L} = \{1, \ldots, c\}$ denote the label set. The $l$ examples in $X^{(l)}$ are labelled $\{y_1, y_2, \ldots, y_l\}$, $y_i \in \mathcal{L}$ and common to all sensors. The $m$ examples in $X^{(u)}$ are associated to the set of multiple observations in (1), which are unlabelled and distributed. When each sensor knows its neighbors and the weights of its links to them, the problem of distributed classification can be formulated as follows.

**Problem 1.** *Each node seeks to predict the correct class $c^*$ of the object of interest $f$, by aggregating information from all available observations over the network and own observations $x_j^{(u)}$, such that all sensors reach a consensus classification decision.*

## 3. DISTRIBUTED CLASSIFICATION WITH CONSENSUS

We present a graph-based formulation of Problem 1, which is inspired by Label Propagation [4]. We solve it in distributed settings by consensus.

We make use of the *smoothness assumption*, which states that if data samples $x_1$ and $x_2$ are similar, then their corresponding labels $y_1$ and $y_2$ should be close. We represent the data labels with a 1-of-$c$ encoding, which allows to form a binary label matrix of size $n \times c$, whose $i$th row encodes the class label of the $i$th example. The class label is basically encoded in the position of the nonzero element. Denote by $\mathcal{M}$ the set of matrices with nonnegative entries, of size $n \times c$. Notice that any matrix $M \in \mathcal{M}$ provides a labelling of the data set by applying the following rule: $y_i = \max_{j=1,\ldots,c} M_{ij}$. We denote the initial label matrix as $Y \in \mathcal{M}$ where $Y_{ij} = 1$ if $x_i$ belongs to class $j$ and 0 otherwise.

We further form the $k$ nearest neighbor ($k$-NN) graph denoted as $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$, where the vertices $\mathcal{V}_d$ correspond to the data samples $X$. Typically, an edge $e_{ij} \in \mathcal{E}_d$ is drawn if and only if $x_j$ is among the $k$ nearest neighbors of $x_i$. However, due to the distributed settings, the nearest neighbors of each example can be chosen only among the labelled ones (as each sensor does not have access to the unlabelled examples, apart from its own observation). It is common practice to assign weights on the edge set of $\mathcal{G}_d$, gathered in a weight matrix $H \in R^{n \times n}$. The similarity matrix $S \in R^{n \times n}$ is further defined as $S = D^{-1/2} H D^{-1/2}$, where $D$ is a diagonal matrix with entries $D_{ii} = \sum_{j=1}^{n} H_{ij}$. Our framework is illustrated in Figure 1 that represents both the sensor graph and the data graph.

We now exploit the special structure of the problem, namely that the multiple observations belong to the same class. If we define a binary class label vector $\lambda = [\lambda_1, \ldots, \lambda_c] \in \mathbb{R}^c$, the optimal classification of Problem 1 should have only one non-zero entry, with the form $\lambda = [0, \ldots, \underbrace{1}_{c^*}, \ldots, 0]$. Intuitively, we seek for one of the $c$ vectors $\lambda$ with only one non-zero entry, which best reflects our smoothness assumption. This optimal vector results in similar class label assignments for pairs that are similar. For each candidate vector $\lambda$, each sensor $j$ locally computes a smoothness criterion as a weighted summation over the labelled examples that reads

$$r(j) = \sum_{i=1}^{l} S_{ji} \|Y_i - \lambda\|^2 \tag{2}$$

where $Y_i$ denotes the $i$th row of the label matrix $Y$. The global smoothness function $Q_d$ then aggregates the local criteria as

$$Q_d(\lambda) \quad = \quad \sum_{j=l+1}^{n} r(j) \tag{3}$$

where the index $j$ runs over the unlabelled examples (observations). Notice that when an unlabelled example $x_j$ ($j > l$) is similar to a labelled example $x_i$ (i.e., the weight $S_{ji}$ is large), then minimizing the above objective function will result in $\lambda$ being smooth across similar examples. Hence, we need to solve the following optimization problem.

Optimization problem: **OPT**
$\min_{[\lambda_1,\ldots,\lambda_c]} Q_d([\lambda_1, \ldots, \lambda_c])$
subject to
$\lambda_p \in \{0, 1\}, p = 1, \ldots, c,$
$\sum_{p=1}^{c} \lambda_p = 1.$

We now describe how one can compute distributively the sum of local functions with consensus algorithms. Distributed consensus [3] has recently become an important computational tool for multimedia data analysis (see e.g., distributed pose estimation applied to face recognition in [2]) and various aggregation tasks in ad-hoc sensor networks. We consider distributed linear iterations of the following form

$$z_{t+1} = W z_t. \tag{4}$$

where $z_t$ represents the values computed by sensors at iteration $t$. When the weight matrix $W$ is properly designed, these distributed iterations can be shown to converge to the average of the values initially measured at all sensors [5].

We finally have all the ingredients to present the distributed algorithm. First, each sensor $j$ computes the nearest neighbors of its observation $x_j^{(u)}$ among the labelled examples as well as the associated similarity weights $S_{ji}$. Next, it computes the value of the objective function $Q_d(\lambda)$ (see eq. (3)) for each candidate class $p$. This involves first a local computation step and then a distributed computation step. In particular, for a certain class $p$, the neighbors of $x_j^{(u)}$ contribute to the calculation of a portion $r(j)$ of the objective function value, which involves only local computation. Next, those portions are averaged distributively, by means of average consensus, so that all observations are taken into account. The total value of the

**Algorithm 1** The distributed MASC algorithm

1: **Input to each sensor**:
$l$: number of labelled data.
$X^{(l)} \in \mathbb{R}^{d\times l}, Y^{(l)}$: labelled examples.
$x^{(u)} \in \mathbb{R}^{d\times 1}$: unlabelled example (observation).
2: **Output at each sensor**:
$\hat{p}$: estimated unknown class.
3: **Initialization at each sensor**:
4: Form the $k$-NN graph $\mathcal{G}_d$ of the data set $\{X^{(l)} \bigcup x^{(u)}\}$.
5: Compute the weight matrix $H \in \mathbb{R}^{(l+1)\times(l+1)}$ of $\mathcal{G}_d$.
6: Compute the diagonal matrix $D$, where $D_{i,i} = \sum_{j=1}^{l+1} H_{ij}$.
7: Compute $S = D^{-1/2} H D^{-1/2}$.
8: **for** $p = 1 : c$ **do**
9: Each sensor sets $\lambda = [0, \ldots, \underbrace{1}_{p}, \ldots, 0]$.
10: Each sensor $j$ computes $r(j) = \sum_{i=1}^{l} S_{l+1,i} \|Y_i - \lambda\|^2$.
11: $q(p) = \sum_{j=1}^{m} r(j)$ :=average_consensus($r$).
12: **end for**
13: $\hat{p} = \arg\min_p q(p)$

objective function is thus computed *at all sensors*, according to eq. (3). The evaluation of $Q_d(\lambda)$ is repeated for all candidate classes and eventually the sensors reach a consensus classification decision, by picking the class that results in the minimum value of the objective function.

We call the proposed distributed algorithm MASC i.e., MAnifold Smoothing under Constraints, and we discuss it below in details. For notational ease, we drop the subscript $j$ from $x_j^{(u)}$ when it is clear from the context that we refer to sensor $j$. The main steps are shown in Algorithm 1. First, each sensor computes the $k$-NN graph of its own data set $\{X^{(l)} \bigcup x^{(u)}\}$ and forms the corresponding $S$ matrix of size $(l+1)\times(l+1)$ (Lines 4-7). Next, each class hypothesis is tested (loop 8-12). For each class hypothesis $p$, each sensor $j$ first computes a scalar number $r(j)$ that involves local computation only; namely a weighted sum of the nonzero entries of the last row of $S$ (i.e., $(l+1)$th row). This corresponds to a portion of the value of the objective function, which captures the smoothness of the label assignment under the current class hypothesis. In order to compute the value of the objective function $q(p)$, the partial sums $r(j)$ need to be added up together and this involves distributed computation. This step is performed by *distributed average consensus* (Line 11), where the summation of all $r$'s is computed *at each sensor*. Note that this results in a scaled version of $q(p)$, due to presence of $1/m$ in the average. However, this has no influence on the classification decision, which is taken in Line 13 by all sensors, after all hypotheses have been tested.

Finally, notice that all observations contribute to the final classification decision, thanks to the use of average consensus. At the end of the algorithm, all sensors reach a consensus decision.

## 4. SIMULATION RESULTS

We show the feasibility of the distributed MASC algorithm in the context of distributed multi-view face recognition. We consider the case of a vision sensor network, where the face of a subject is captured by different cameras organized in an ad-hoc network. Each observation in this case represents a facial image captured under different viewing angles. Observe again that all observations belong to the same class and that the problem resides in estimating the unknown class i.e., recognizing the subject.

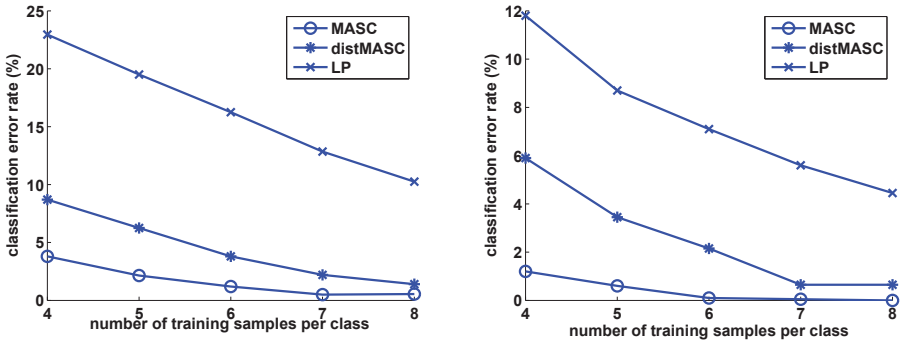
We compare our distributed algorithm with two centralized algorithms for the classification of multiple observations, namely, a centralized version of the distributed MASC algorithm [6] and a Label Propagation method [4]. In the centralized scenario, each algorithm has access to all observations $X^{(u)}$ and can further form a full similarity matrix $S \in \mathbb{R}^{n\times n}$. We use the UMIST database [7] in our simulations. The UMIST database contains 20 people under different poses. The number of different views per subject varies from 19 to 48. We used a cropped version of the UMIST database that is publicly available[1]. Then, the sensor network is constructed with a random geographic graph model [8]. According to this model, we randomly distribute $m$ nodes on a 2-dimensional unit area. Two nodes are adjacent if their Euclidean distance is smaller than $\epsilon = \sqrt{\frac{\log m}{m}}$, which ensures connectedness with high probability. Finally, in all algorithms we use Gaussian weights in the data graph. which are defined as

$$H_{ij} = \begin{cases} \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}) & \text{when } (i,j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

We investigate the classification performance of distributed MASC with respect to that of centralized MASC and centralized Label Propagation (LP). We assume that the distributed average consensus in Line 11 of Algorithm 1 has converged to the asymptotic solution. In other words, we assume that the distributed summation is exact. The purpose of this experiment is to investigate whether the distributed algorithm suffers any loss in performance due to partial information. We set the number of nearest neighbors $k$ to 3 in all methods, and the regularization parameter in LP is set to $\mu = 0.1$. We investigate the behavior of all methods, when the number of multiple observations $m$ varies. For each particular value of $m$, we measure the classification error rate for different sizes of training set. In particular, we increase gradually the number of training examples per class and measure the average classification error rate over 100 random experiments (i.e., 100 random splits of the data into training (labelled) and test (unlabelled) sets).

Figs 2(a) and 2(b) show the obtained results for different number $m$ of observations, when the number of training examples per class increases from 4 to 8 with step 1. Notice that there is a small loss in performance of distributed MASC with respect to its centralized counterpart. To see why this happens, it is important to observe that the $k$-NN graph in the distributed case is different than that in the centralized case. This is due to the fact that the multiple observations are collected distributively. Hence, the neighbors of an observation $x^{(u)}$ can only be selected among the labelled examples, whereas in the centralized case they may be selected among all labelled and unlabelled examples. When the training set becomes larger, this phenomenon decays and the difference between MASC and its distributed variant becomes negligent. Notice finally, that even with this small loss in performance, distributed MASC is still superior to (centralized) LP, which does not exploit the fact that all observations belong to the same class. Note however that it is exactly this difference in the construction of the $k$-NN graph that allows the distributed MASC algorithm have much lower computational cost ($O(l+c)$) than that of centralized MASC ($O((l+m)^2+c)$). Essentially, this is the main characteristic that makes it efficient and feasible in distributed settings. However, this comes at the cost of a small performance loss, which becomes even smaller when the training set is sufficiently large.

[1] http://www.cs.toronto.edu/~roweis/data.html




(a) $m = 6$ (b) $m = 10$

**Fig. 2**. Difference in performance between MASC and its distributed version versus the number of training samples (per class).

In the previous experiment, we assumed that the distributed summation in Line 11 of Algorithm 1 is exact, for the sake of simplicity. In this experiment we drop this assumption and we investigate the effect of employing distributed consensus for the computation of this sum. Note that our goal in this particular experiment is to study the effect of consensus on the classification performances. For this reason, we use the same $k$-NN graph of distributed MASC in its centralized counterpart. This way, the performance difference of the two algorithms will only be due to the summation part. First, we split randomly the data set into training and test set, by including two examples per class in the labelled set $X^{(l)}$ and the rest is assigned to the test set. We form $m = 10$ multiple observations, which are drawn randomly from the test set, and we use $k = 1$ in the construction of the $k$-NN graph.

Fig. 3 shows the average classification error rate (over 500 random experiments) measured on a certain sensor, say the first one, when the number of iterations in distributed consensus varies from 1 to 100 with step 5. We use the Metropolis weight matrices [5], which are known to lead iteration $z_{t+1} = W z_t$ to asymptotic convergence to the average $\bar{z}_0 = \frac{1}{m}\sum_{i=1}^{m} z_0(i)$. Observe that fairly few iterations, namely between 30 and 40, provide sufficient accuracy in the computation of the distributed sum, such that it reaches similar performance as the centralized MASC algorithm.

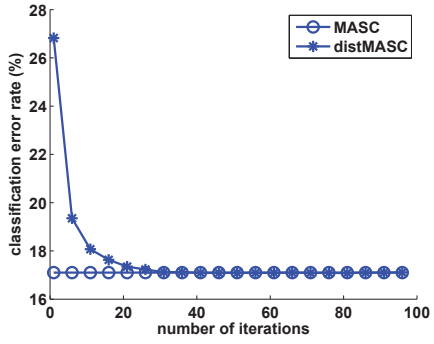



**Fig. 3**. Average classification error rate vs consensus iterations, for the Metropolis weight matrices.

## 5. CONCLUSIONS

We studied the problem of classification of multiple observations in the scenario where the observations are collected distributively. We showed that distributed classification in ad-hoc sensor networks can be effectively performed using distributed consensus. In particular, we proposed a distributed graph-based algorithm that aggregates information from all observations across the network and results in a consensus classification decision among the sensors. We have illustrated its performance in the context of distributed multi-view face recognition. The simulation results have shown that, when the training set is sufficiently large, the classification decision of the distributed algorithm is equivalent to that of the centralized algorithm.

## 6. REFERENCES

[1] K. Flouri, B. Beferull-Lozano, and P. Tsakalides. Distributed consensus algorithms for SVM training in wireless sensor networks. *16th European Signal Processing Conference (EUSIPCO)*, 2008.

[2] R. Tron and R. Vidal. Distributed face recognition via consensus on SE(3). *8th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*, 2008.

[3] D. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[4] T. Navin Lal J. Weston D. Zhou, O. Bousquet and B. Scholkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[5] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, (53):65–78, February 2004.

[6] E. Kokiopoulou, S. Pirillos and P. Frossard. Graph-based Classification for Multiple Observations of Transformed Patterns. *Proceedings of ICPR*, 2008.

[7] D. B Graham and N. M Allinson. Characterizing virtual eigensignatures for general purpose face recognition. *Face Recognition: From Theory to Applications*, 163:446–456, 1998.

[8] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. on Information Theory*, 46(2):388–404, March 2000.