DISTRIBUTED SVM APPLIED TO IMAGE CLASSIFICATION

Effrosyni Kokiopoulou and Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute - ITS
CH- 1015 Lausanne, Switzerland
{effrosyni.kokiopoulou,pascal.frossard}@epfl.ch

ABSTRACT

This paper proposes an algorithm for distributed classification, based on a SVM scheme. The contribution of each support vector is approximated by low complexity distributed thresholding over sub-dictionaries, whose union forms a redundant dictionary of atoms that spans the space of the observed signal. Redundant dictionaries allow for sparse representation of the observed signal, hence a good approximation of the support vector contributions, which is moreover robust to noise. The algorithm is applied to distributed image classification, in the context of handwritten digit recognition in a sensor network. The experimental results indicate that the proposed method is capable of achieving the same classification performance as the standard (non distributed) SVM, with an increased resiliency to noise.

1. INTRODUCTION

Advances in processor and radio technology have motivated a lot of research efforts on sensor networks, which is an emerging and promising field in the signal processing community. Distributed algorithms are getting increasing attention, as they allow to shift the computational complexity towards the receiving end, possibly without loss in performance. In this paper, we study the distributed classification problem where the observed signal x which is to be classified, is typically apart from the classifying unit.

In particular, we investigate a distributed image classification scenario, in a network of inexpensive general purpose vision sensors with severe limitations on memory and power capabilities. The sensors take measurements of the observed image x by projecting it on a redundant dictionary of visual primitives [1] and then by keeping a few of the largest components. This can be viewed as a feature extraction process. These features are sent to a central processing unit, usually called Fusion Center (FC). The proposed distributed classification algorithm makes use of Support Vector Machines (SVM) [2], which is among the state-of-the-art classification algorithms. The choice of SVMs as a classification method is also motivated by the nature of the features extracted at the sensors, which form a sparse approximation of the observed signal.

To the best of our knowledge, there is not much work done on distributed image classification. A somewhat related framework is given in [3], where the authors propose the use of rotation invariant features for distributed image retrieval. However, the primary target of our algorithm is generic classification problems, like handwritten digit recognition and face recognition applications, which are different than image retrieval. We show that the distributed classification scheme allows to reach performance that is very similar to a classical (non-distributed) SVM algorithm, with an improved resilience to noise.

2. SUPPORT VECTOR MACHINES OVERVIEW

This section presents a brief overview of SVM algorithms, and the motivations behind their choice for distributed classification. Denote by $X = [x_1, \ldots, x_n] \in R^{d \times n}$ the training samples and by $Y \in R^{n \times C}$ their associated class labels, where C is the number of classes. Consider first the binary classification problem and the case of linear discriminant functions. Each test sample $x \in R^d$ is classified to class 1 or class 2 according to the sign of a linear discriminant function of the form $\langle w, x \rangle + w_0$, where w, w_0 have been obtained by training and w_0 is usually called *bias*.

In SVMs the goal is to determine w such that the margin among the training samples of different classes is maximized [2, ch.4]. This involves the solution of a quadratic problem (QP) whose dual form is

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j, \qquad (1)$$

where $\alpha_i, i = 1, \ldots, n; \alpha_i \leq 0$ are the Lagrange multipliers. It turns out that the vector w has a sparse form over the training data samples i.e., w is a linear combination of a few training samples, usually called the support vectors. Those vectors

This work has been partly supported by the Swiss National Science Foundation, under grants PP-002-68737, and NCCR IM2.

correspond to the training data whose Lagrange multiplier α_i is nonzero. Thus, it holds that

$$w = \sum_{i \in SV} \alpha_i y_i s_i, \tag{2}$$

where s_i is the i-th support vector. The signal x is classified according to the sign of the discriminant function, which reads, in the linear SVM case:

$$g(x) = \sum_{i \in SV} \alpha_i y_i \langle s_i, x \rangle + w_0. \tag{3}$$

Consider now the nonlinear SVM case. In this case we employ a nonlinear mapping $\phi : R^d \rightarrow \Phi$ which embeds the data in a higher dimensional space (perhaps of infinite dimension) and we perform implicitly linear classification in that space. For the binary classification problem we seek again a discriminant function of the form $g(x) = \langle w, \phi(x) \rangle + w_0$. The dual form of the SVM QP now becomes

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(x_i)^\top \phi(x_j). \tag{4}$$

The solution w now has the form

$$w = \sum_{i \in SV} \alpha_i y_i \phi(s_i), \tag{5}$$

and the classification of a new data sample x involves evaluating the sign of $g(x) = \sum_{i \in SV} \alpha_i y_i \langle \phi(s_i), \phi(x) \rangle + w_0$. Using Mercer kernels it holds that $K(x, y) = \langle \phi(x), \phi(y) \rangle$, and the inner product of $\phi(x)$ and $\phi(y)$ can be computed as a function of x and y. Therefore, the discriminant function now becomes:

$$g(x) = \sum_{i \in SV} \alpha_i y_i K(s_i, x) + w_0. \tag{6}$$

In the case of multi-class classification with C classes, there is no straightforward way to extend the SVM maximum margin principle. Usually, one combines several binary SVM classifiers. Two well known methods are the one-versus-all and all-to-all. In the first method, one trains C classifiers such that when the i-th classifier is trained, the training samples of the i-th class are assigned in one class and all the remaining training samples are assigned to the other class. In this case a new test sample is assigned according to the following decision rule

$$g(x) = \arg\max_k \{ \sum_{i \in SV(k)} \alpha_i^{(k)} y_i K(s_i^{(k)}, x) + w_0^{(k)} \}, \tag{7}$$

where k runs among all C classifiers. The other alternative is the all-to-all method. In this case, we train $\frac{C(C-1)}{2}$ binary classifiers, one for each pair of classes and a decision rule similar to (7) is employed. In this paper, we use the first one-versus-all method due to its simplicity and its low computational complexity. It is important to observe that, in order to classify a new test sample x, we need only to compute the kernel products $K(s_i, x)$ of x with the support vectors. When K is a Mercer kernel then the product $K(s_i, x)$ corresponds to a function of the inner product $\langle s_i, x \rangle$ [2, ch.4]. We show in the next section how we can compute the products $\langle s_i, x \rangle$ (and therefore $K(s_i, x)$) in a sensor network using the framework of redundant dictionaries.

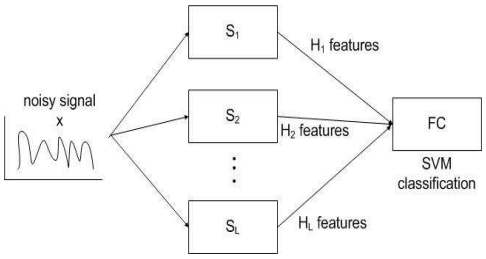

Fig. 1. Sensor network architecture.

3. DISTRIBUTED CLASSIFICATION

3.1. Distributed Thresholding

Consider now the classification problem using SVM in the context of a sensor network as illustrated in Figure 1, where S_i denotes the i-th sensor and FC the Fusion Center. Motivated by the fact that the decision rule of a test sample x can be applied when the components $\langle s_i, x \rangle$ are known, we propose the use of the redundant dictionaries framework for feature extraction at the sensors. We assume the existence of a redundant dictionary $\mathcal{D}$ which spans the Hilbert space $\mathcal{H}$ of all images. Often, $\mathcal{D}$ is constructed in a structured way by applying transformations γ to a mother function ϕ, $\mathcal{D} = \{\phi_\gamma,\ \phi_\gamma = U(\gamma)\phi,\ \gamma \in \Gamma\}$, where Γ is an index set on the parameter space. The components ϕ_γ are usually called *atoms*. The number of different parameters directly drives the redundancy of the dictionary.

Denote by L the number of sensors. We partition the full dictionary $\mathcal{D}$ into L subsets $C_i,\ i = 1, \ldots, L$. The i-th sensor S_i is assigned the atoms of subset C_i. The sensors observe a possibly noisy signal x that is to be classified. Each sensor projects x on its own part of the dictionary and thresholds the resulting components of the following form

$$\langle x, \phi_{\gamma_j} \rangle, \gamma_j \in C_i. \tag{8}$$

In this case, by thresholding, we mean that the sensor keeps only the H_i largest components (in magnitude). This process can be interpreted as feature extraction and the components from (8) are the features used by our distributed algorithm. In the sequel, each sensor sends the H_i components (features) that survived the thresholding part to the fusion center.

3.2. Dictionary construction and distribution

Redundant dictionaries have been successfully used for sparse signal representations, particularly for non-linear image expansions [1]. The intuition for using redundancy in our problem is that thresholding is able to capture the most prominent primitives of the signal which are helpful for discrimination purposes. These primitives represent geometric features in the case of image expansion.

The dictionary is split into disjoint sub-dictionaries C_i, $i = 1, \ldots, L$, which are distributed to the different sensors. The motivation for having sub-dictionaries as disjoint as possible lies in the fact that the thresholded components (i.e. largest ones) are usually located around high correlation peaks. Notice that, having many sub-dictionaries which are as much independent as possible, yields components from many peaks, hence more meaningful ones. In our algorithm, we partition the full dictionary into smaller parts, using Spectral Clustering [4]. In particular, we build a weighted graph $G = (N, E)$ where each node corresponds to a different atom in the dictionary and the edges are $e_{ij} = \langle \phi_i, \phi_j \rangle$. Then, clustering is performed by computing the eigenvectors of the Laplacian matrix of G.

3.3. Classification

The FC collects all the features ($M = \sum_i H_i$ in total) from the sensors and classifies the signal x by evaluating the discriminant function given by equation (3) for linear SVM and by equation (6) for nonlinear SVM. We assume that the FC has trained an SVM for the particular classification problem we are interested in. Thus, it has computed the support vectors, the Lagrange multipliers a_i and the bias w_0. So, the missing part is the components $\langle x, s_i \rangle$. If the FC knew these components, then it would feed them into equation (3) or equation (6) and would classify the observed signal x. However, these components involve x which is observed only at the sensors and the FC can only approximate them via the features provided by the sensors.

Let us see now how one can approximate those components using the features collected from the sensors. Denote by $\mathcal{S}$ the union of all index sets of the atoms of each sensor, which survived the thresholding operation and made it to the FC. Denote also $\Phi_S \in R^{d \times M}$ the matrix whose columns contain the atoms of set $\mathcal{S}$. Recall that the FC has the full dictionary and therefore it can form the matrix Φ_S. Then, in order to compute $\langle x, s_i \rangle$, we approximate each support vector s_i from the column span of Φ_S. This is done by solving a small least squares (LS) system of the form

$$\beta^* = \arg\min_{\beta} \|s_i - \Phi_S \beta\|_2 = \Phi_S^{\dagger} s_i, \qquad (9)$$

where $\Phi_S^{\dagger}$ denotes the Moore-Penrose pseudo-inverse of Φ_S [5]. The LS solution is optimal with respect to mean squared error. Note that the above LS system must be solved for each support vector. This can be performed efficiently by solving a LS system with multiple right hand sides, as follows

$$B^* = \arg\min_{B} \|S - \Phi_S B\|_2 = \Phi_S^{\dagger} S. \qquad (10)$$

In the above formula we introduced S, whose columns are the support vectors that must be approximated, and B, whose columns are the coefficients to be computed. This involves the computation of the pseudo-inverse $\Phi_S^{\dagger}$ only once. Additionally, assume that the collected features are stacked into a vector $r = [\langle x, \phi_{\gamma_1} \rangle, \ldots, \langle x, \phi_{\gamma_M} \rangle]^{\top}$. If β^* are the approximation coefficients for s_i, then observe that the components $\langle s_i, x \rangle$ can be approximated as follows

$$\langle s_i, x \rangle \quad \approx \quad \langle \sum_{j \in \mathcal{S}} \beta_j^* \phi_{\gamma_j}, x \rangle = \sum_{j \in \mathcal{S}} \beta_j^* \langle \phi_{\gamma_j}, x \rangle = \langle \beta^*, r \rangle.$$

Therefore, once the support vectors have been approximated by solving Eq. (10), the computation of $\langle s_i, x \rangle$'s simplifies to an inexpensive inner product. For the nonlinear SVM case, we use the $\langle s_i, x \rangle$'s to further compute the kernel product $K(s_i, x)$.

Finally, note that one possible distributed approach is to simply distribute the support vectors to the sensors. Then each sensor projects x on its own support vectors and the FC collects the partial results. However, this results in very specialized sensors, which will be able to work only for specific instances of the problem. In many cases, the SVM has to be re-trained (for instance by incremental learning, boosting etc) and the sensors have then to be updated. On the contrary, our algorithm uses general purpose sensors that are independent of the support vectors at hand.

4. EXPERIMENTAL RESULTS

We provide experimental results that demonstrate the validity of the proposed algorithm for distributed image classification. For the SVM training, we use SPIDER[1] an object-oriented machine learning library for MATLAB. We consider the multi-class classification problem in hand-written digit recognition. We use the digit collection that is publicly available at S. Roweis web page[2]. This collection contains 20×16 bit binary images of "0" through "9", and each class contains 39 samples.

For the dictionary construction, we use the Anisotropic Refinement (AR) atoms which have been successfully used in image coding [1]. These are edge-like atoms which are obtained by the partial second derivative of the Gaussian function with respect to one of its coordinates. In particular, the mother function is $\phi = \frac{2}{\sqrt{3\pi}}(4x^2 - 2)\exp(-(x^2 + y^2))$. In this case, a geometric transformation $\gamma = (\vec{t}, \vec{a}, \theta)$ used for the

[1] http://www.kyb.tuebingen.mpg.de/bs/people/spider/

[2] http://www.cs.toronto.edu/ roweis/data/binaryalphadigs.mat

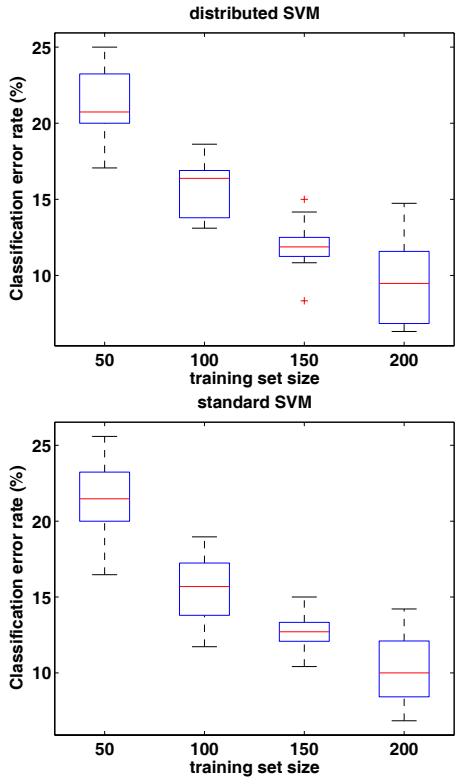

Fig. 2. Classification error rates for both distributed and standard SVM.

dictionary construction, consists of five parameters: translation $\vec{t}$, anisotropic dilations $\vec{a}$ and rotation θ. We use 10 rotation angles in $[0, \pi]$, and 10 scales which are logarithmically equi-distributed in the interval $[1, N]$, where N is the size of the image. Finally, the clustering of the atoms into subdictionaries is performed on the non-translated atoms (i.e., $\vec{t} = \vec{0}$). Translation is eventually applied on the atoms (after clustering) and spans all possible pixel locations.

Classification performance. We now explore the multiclass classification performance of both distributed SVM and standard SVM across various sizes of training set. We use linear SVM and the one-versus-all scheme for multi-class training. We compute the classification error rate (in %) for training set sizes of $\{5, 10, 15, 20\}$ samples per class. In order to make sure that the results are not biased by a specific instance of the training set, we perform 10 random realizations of the training set. In our experiment, we use $L = 30$ subdictionaries and keep $H = 2$ components per dictionary. Thus, 60 features in total are collected by the FC. Figure 2 illustrates the results in boxplot notation. Observe that the distributed SVM algorithm is competitive with the standard (non - distributed) SVM algorithm.

Resilience to noise. We also study the behavior of both distributed and standard SVM algorithm with respect to noise on the observed signal x. We use a large training set of 20 samples per class (200 samples in total) and a small test set (19 samples) consisting only of digits "0". We add additive white Gaussian noise on the test set and we measure the classification error rate for SNR $\in [-5 : 5 : 10]$. We report in Table 1 the average results over 10 random realizations of the training/test set. Observe that the proposed algorithm is more resilient to noise than standard SVM for low values of SNR. For higher values of SNR both algorithms exhibit similar behavior, as expected.

SNR	-5	0	5	10
distributed SVM	22.11	9.47	5.79	3.16
standard SVM	27.37	12.11	6.32	3.16

Table 1. Classification error rate (%) for various values of SNR.

5. CONCLUSIONS

This paper has presented a distributed classification algorithm, with an application to image classification in sensor networks. The proposed algorithm is based on low-complexity thresholding over redundant dictionaries at the sensors. The extracted features are inner product values, which are subsequently used at the fusion center for classification using an SVM. Experimental results suggest that the proposed algorithm is competitive to the standard SVM and it is also more resilient to noise.

6. REFERENCES

[1] Figueras i Ventura R, Vandergheynst P, and Frossard P, "Low rate and flexible image coding with redundant representations.," *IEEE Transactions on Image Processing*, February 2006.

[2] A. Webb, *Statistical Pattern Recognition*, Wiley, 2nd edition, 2002.

[3] B. Beferull-Lozano, H. Xie, and A. Ortega, "Rotation-invariant features based on steerable transforms with an application to distributed image classification," in *Proc. IEEE Int. Conference on Image Processing*, 2003, pp. 521–524.

[4] Andrew Y. Ng, Michael Jordan, and Yair Weiss, "On spectral clustering: analysis and an algorithm," in *NIPS 14*, 2002.

[5] G. H. Golub and C. Van Loan, *Matrix Computations, 3rd edn*, The John Hopkins University Press, Baltimore, 1996.