2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Dynamic Adaptive Graph Construction: Application to Graph-based Multi-observation Classification

F. Dornaika[1,2], R. Dahbi[3], A. Bosaghzadeh[1], and Y. Ruichek[3]
[1] University of the Basque Country (UPV/EHU)
[2] IKERBASQUE, Basque Foundation for Science, San Sebastian, Spain
[3] IRTES-SeT, University of Technology Belfort-Montbeliard, Belfort, France

***Abstract*—Most of graph construction techniques assume a transductive setting in which the whole data collection is available at construction time. Addressing graph construction for inductive setting, in which data are coming sequentially, has received much less attention. Constructing the graph from scratch can be very time consuming. In this paper, we propose an efficient dynamic graph construction method that adds new samples (labeled or unlabeled) to a previously constructed graph. We use a Two Phase Weighted Regularized Least Square (TPWRLS) coding scheme to represent new sample(s) with respect to an existing data set. The representative coefficients are then used to update the graph affinity matrix. The proposed method not only appends the new samples to the graph but also updates the whole graph structure by discovering which nodes are affected by the introduction of new samples and by updating their edge weights. The proposed construction framework is applied to the problem of graph-based label propagation using multiple observations in a semi-supervised scenario. Experiments on three public image databases show that, without any significant loss in the accuracy of the final classification, the proposed dynamic graph construction is more efficient than the batch graph construction.**

Keywords: Data self-representativeness, graph construction, graph-based semi-supervised learning, multiple observations

I. Introduction

Graph is a very powerful tool and a central object for any graph-based learning task with a wide range of applications in computer vision, signal processing and pattern recognition [1]. A graph consists of nodes which correspond to data and weighted edges which show how strong two nodes are connected [2], [3], [4], [5]. Recently proposed methods try to merge edge setting and edge weighting into a single process and estimate the edges and their weights simultaneously [6], [7]. With the rapid growth in the use of digital still and video cameras, we are facing a huge amount of data. In many practical cases new samples come continually and we need to analyze the relation between the already available and recently received samples. Among different graph construction methods, KNN graph is the most efficient graph construction technique (despite its less accurate results). For speeding up graph construction process, [8] builds a small graph that shows the relation between the whole database with respect to some sample (anchor) points. In this method, anchor points are firstly selected using the Kmeans clustering algorithm and the relation between the database and these anchor points are then calculated. Although these algorithms have low computational complexities, their main objective is fast graph construction and they assume that the whole data is available which makes them not adequate for updating tasks and incremental learning. If we want to use the above methods for updating tasks, whenever new samples come, we have to reconstruct the whole graph from scratch. This is computationally expensive, especially for online learning and classification. However, to the best of our knowledge, apart from [9], incremental graph construction has not received a lot of attention. In [9], the authors propose two strategies, Forward and Backward. In the Forward strategy, the objective is to start from a set of points and empty set of edges. Then randomly select two nodes and use a local insertion method to add all available samples to the graph to build a complete one. In the local insertion method, an optimal space which potentially can contain the closest points to the query point is defined and the samples in that space are added to the graph. In the Backward strategy, they start from an over-connected graph and try to remove the edges which does not satisfy the considered neighborhood property.

In this paper, we propose an incremental graph construction method which rapidly and accurately estimates the edge and edge weight for the new samples and then updates the whole graph structure. Rather than recalculating the relation between all samples (building the graph in a batch mode), which is computationally very expensive, we only look on the new samples and update the edges and weights of their close neighbor nodes. It should be mentioned that our work is different from the one proposed in [9] in several aspects. The objective in [9] is for edge setting, however, the goal in our method is affinity matrix construction which includes an edge weighting phase. Also the edge setting in [9] is based on a symmetric distance function whereas here we use a non-symmetric similarity function. Moreover, we conduct several experiments on different post-graph learning tasks using variety of image databases to evaluate the proposed method in different scenarios.

The reminder of this paper is organized as follows: Section II describes the TPWRLS coding technique that is recently introduced and used for graph construction. In section III, we explain the motivation for dynamic graph construction and our proposed dynamic graph construction technique. Section IV presents an application of the proposed framework to the online recognition of multiple observations.

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

II. Review of the TPWRLS graph construction

In this section, we briefly present our proposed graph construction method coined TPWRLS [10]. Assume we have a dataset represented by a data matrix $\mathbf{X} = [\mathbf{x}_1\mathbf{x}_2...\mathbf{x}_N]$ ($\mathbf{X} \in \mathbb{R}^{D\times N}$). The aim of any graph construction method is to estimate the graph affinity matrix associated with the data. Let $\mathbf{W} \in \mathbb{R}^{N\times N}$ denote this matrix. In TPWRLS, each row in the affinity matrix $\mathbf{W}$ is estimated using a two phase coding scheme. A row $\mathbf{W}(i,:)$ contains the weights of all edges that links the sample $\mathbf{x}_i$ to the rest of all samples. By definition we have $W_{ii} = 0$. Let the vector $\mathbf{b} \in \mathbb{R}^{N-1}$ denote the row vector $\mathbf{W}(i,:)$ from which W_{ii} is removed. This unknown vector is estimated in two consecutive phases that use a coding that is based on the following criterion. The unknown vector $\mathbf{b}$ can be calculated by minimizing the following criterion (WRLS):

$$\mathbf{b} = \arg\min_{\mathbf{b}} \frac{1}{2}\left(\|\mathbf{x}_i - \mathbf{X}_i\mathbf{b}\|_2^2 + \sigma \sum_{j=1}^{N-1} p_j^2\, b_j^2 \right) \quad (1)$$

where $\mathbf{X}_i$ is the data matrix from which the sample $\mathbf{x}_i$ is removed, p_j is a positive weight associated with example $\mathbf{x}_j$ (or equivalently b_j). In Eq. (1), the criterion has two terms: the reconstruction error and the weighted regularized term. σ is a small positive scalar that balances the two terms effect. The weight p_j can be set to the distance between sample $\mathbf{x}_i$ and $\mathbf{x}_j$. The solution to Eq. (1) has a closed form solution that is given by:

$$\mathbf{b} = \left(\mathbf{X}_i^T\mathbf{X}_i + \sigma\,\mathbf{P}\right)^{-1} \mathbf{X}_i^T\,\mathbf{x}_i \quad (2)$$

$\mathbf{P}$ is a diagonal matrix with elements $\mathbf{P}_{jj} = p_j$.

The process for the TPWRLS graph building is shown in Algorithm 1. This algorithm estimates the i^{th} row of the affinity matrix by coding the sample $\mathbf{x}_i$ w.r.t. to the set $\mathbf{X}_i$. This estimation (the inner loop) also gives the TPWRLS coding of a sample w.r.t to a given dataset. The obtained graph is a directed graph, i.e., the weight matrix $\mathbf{W}$ is asymmetric.

Data: A dataset $\mathbf{X}$
Result: A weight matrix $\mathbf{W}$ of its graph
for $i = 1$ **to** N **do**
  * Select the sample $\mathbf{x}_i$ and form $\mathbf{X}_i = \mathbf{X}$ - $\{\mathbf{x}_i\}$
  * From the $N-1 \times N-1$ diagonal matrix $\mathbf{P}$
  * Estimate the vector $\mathbf{b}$ using Eq. (2)
  * Calculate a threshold for $\mathbf{x}_i$ as $TH(\mathbf{x}_i) = \frac{1}{N-1}\sum_{j=1}^{N-1}|b_j|$
  * Select the examples $\mathbf{X}_s$ (whose $|b_j| > TH(\mathbf{x}_i)$)
  * Set the new weight matrix $\mathbf{P}'$ using $p'_j = \frac{1}{|b_j|}$
  * Estimate the vector $\mathbf{b}'$ as $\mathbf{b}' = \left(\mathbf{X}_s^T\mathbf{X}_s + \sigma\,\mathbf{P}_s\right)^{-1} \mathbf{X}_s^T\,\mathbf{x}_i$
  * Set the sparse vector $\mathbf{b}$ from $\mathbf{b}'$
  for $j = 1$ **to** N **do**
    if $i < j$ **then**
      Set $W_{ij} = |b_j|$
    else
      Set $W_{ij} = |b_{j-1}|$
    end if
  end for
end for

Algorithm 1: TPWRLS graph construction

III. Proposed incremental graph construction.

A. Problem statement

Consider the database $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ which is a $D \times N$ matrix containing N samples with dimension D. The affinity graph is represented by $G = (V; E; \mathbf{W})$ where V is the set of nodes ($|V| = N$), E is the set of edges and $\mathbf{W}$ is the $N \times N$ edge weight matrix which is also called the affinity matrix. The weight of the edge (i, j) is given by W_{ij} which quantifies the similarity between the incident nodes i and j. We have new sample matrix represented by $\mathbf{X}' = \{\mathbf{x}'_1, \mathbf{x}'_2, ..., \mathbf{x}'_r\}$ where r is the number of new samples. Note that the r new samples can be labeled, unlabeled or partially labeled. The goal is to construct an affinity graph $\widetilde{G}$ that is associated to the union of the initial database and the new samples (i.e., $\widetilde{\mathbf{X}} = \mathbf{X} \cup \mathbf{X}'$). The arrival of new samples may happen many times.

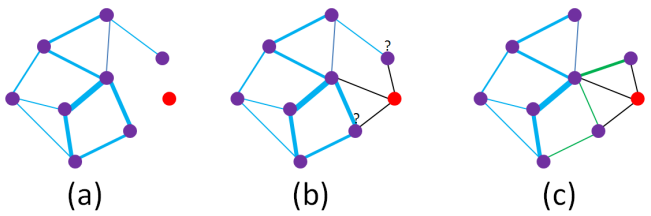

Fig. 1. Basic idea of our proposed method. (a) 8 available samples which are connected (colored in violet) and one new sample (colored in red). (b) The new received sample is inserted in the already available graph. (c) The edge connections of samples which are very close to the new one are reevaluated and the edges and edge weights are changed accordingly (colored in green).

B. Basic idea

Our proposed incremental graph construction method is based on the assumption that adding a new sample to the database will only affect its nearby samples, i.e., the edges of the graph of only a few close samples should be updated. Hence, after receiving new samples, it is not necessary to rebuild the graph from scratch. Fig. 1 illustrates the basic idea of our proposed method. In Fig. 1(a), we have 8 available samples connected to each other and colored in violet and a new sample in red. The segment between a pair of nodes shows the edge and its thickness represents the edge weight, the higher the thicker. At this stage, the graph relating the violet nodes is available. Fig. 1(b) shows the connection between the new received sample and the available ones. The two nodes with question marks on top, are the nodes which are very close/similar to the new sample. Hence, their current relation with the samples (edges) have to be reevaluated for possible updates. In Fig. 1(c), we update the edge and edge weights

of samples which are close to the new one. The modified edges and weights are colored in green. As we can see, one edge is removed and some of the weights have been changed (weakened or strengthened). This is due to the fact that the insertion of a new sample will affect the edges and edge weights of its nearby samples. At this point, the graph is built dynamically without invoking a full construction from scratch.

We propose a dynamic graph construction method which adds the recently arrived samples to a previously available graph (with their weighted edges) and then update the whole graph structure locally due to the introduction of the new sample. This scheme will have lower computational complexity compared to the batch graph construction. Due to lower computational complexity of the proposed method, it is faster than batch graph construction.

C. Proposed dynamic graph construction method

In the previous section, we presented the basic idea. In this section, we present our proposed dynamic graph construction method in more details. As explained in the previous section, one solution is to construct a new graph using all data together. However, this solution is computationally expensive when a lot of test samples should be inserted into the graph and/or the graph construction has to be repeated several times as it is the case for online learning and classification. Consider the initial data matrix $\mathbf{X}$ with its corresponding affinity matrix $\mathbf{W}$ and the new sample matrix as $\mathbf{X}'$. We want to construct a new affinity graph $\widetilde{\mathbf{W}}$ ($(N+r)\times(N+r)$ matrix) which shows the relation between the whole data. We build a data matrix containing the union of database and new samples (i.e., $\widetilde{\mathbf{X}} = \{\mathbf{X} \cup \mathbf{X}'\}$). The new unknown affinity matrix has the following form:

$$\widetilde{\mathbf{W}} \quad = \quad \left[\begin{array}{cc} \widetilde{\mathbf{W}}_{dd} & \widetilde{\mathbf{W}}_{dn} \\ \widetilde{\mathbf{W}}_{nd} & \widetilde{\mathbf{W}}_{nn} \end{array} \right] \tag{3}$$

where $\widetilde{\mathbf{W}}_{dd}$ shows the similarity between the database samples with each other, $\widetilde{\mathbf{W}}_{dn}$ represents the similarity between the database samples and new samples, $\widetilde{\mathbf{W}}_{nd}$ shows the similarity between the new samples and database samples and $\widetilde{\mathbf{W}}_{nn}$ is the sub-matrix that demonstrates the similarity between new samples. We estimate the new affinity matrix $\widetilde{\mathbf{W}}$ row by row. In the first step (inserting the new samples), for each new sample $\mathbf{x}'$, we use TPWRLS algorithm to code it with respect to the whole dataset (i.e., $\widetilde{\mathbf{X}}$) and obtain its reconstruction coefficients, $\mathbf{b}'$, which is a vector of dimension $1\times(N+r)$. We put these coefficients, which show the similarity between the new sample and the whole data matrix, directly in the bottom of final graph (i.e., $[\widetilde{\mathbf{W}}_{nd}\widetilde{\mathbf{W}}_{nn}]$).

In the second step (updating the local structure of the graph), we update the edges and edge weights of the samples which seem to be very related to the new samples. We use reconstruction coefficients (the ones in $\widetilde{\mathbf{W}}_{nd}$) as similarity criterion to find these close samples.

Two scenarios can be adopted to select close samples, adaptive and fixed selection. In adaptive selection, a threshold is used in order to select close samples. The threshold can be the result of applying any statistical function on the coefficients as:

$$TH(\mathbf{y}) = \text{STAT}(|b'_1|, \ldots, |b'_N|), \tag{4}$$

where $\text{STAT}(|b'_1|, \ldots, |b'_N|)$ is a statistical function that can select the most relevant or similar samples to the recently added sample $\mathbf{x}'$. $\mathbf{b}'$ is the TPWRLS code associated with the new sample. One possible choice for the threshold can be the average of the obtained coefficients:

$$TH(\mathbf{x}') = \frac{1}{N}\sum_{j=1}^{N} |b'_j|. \tag{5}$$

In the fixed selection scenario, a percentage of the samples having the largest coefficients will be chosen to have their edges updated. In fixed selection, a predefined number of samples are selected as the most similar ones. By any of the above mentioned selection schemes, we construct a new data subset represented by $S = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_Q\}$. This subset of samples is the union of all close samples found for each new sample.

This matrix contains Q closed selected samples that their edge and edge weights may have been affected. As a consequence their graph coefficients should be updated. In the updating phase, we code each sample of S with respect to the whole database (i.e., $\widetilde{\mathbf{X}}$) using TPWRLS coding algorithm. The obtained coefficients of each sample will be inserted into the corresponding row of the new affinity graph (i.e., $[\widetilde{\mathbf{W}}_{dd}\widetilde{\mathbf{W}}_{dn}]$). For the rest of the samples which has not been selected, we directly copy their coefficients from similarity matrix $\mathbf{W}$ into the sub-matrix $\widetilde{\mathbf{W}}_{dd}$ of the new affinity graph. We set corresponding columns of the sub-matrix $\widetilde{\mathbf{W}}_{dn}$ to zero. This is justified by the fact that these samples are far from new sample(s) and they do not interact with them. Hence, there is no edge between them and the corresponding weights in the affinity matrix are zero.

Algorithm 2 summarizes the proposed dynamic graph construction method.

IV. Performance evaluation

A. Targeted task: Label propagation for multiple observations recognition

In this section, we study the performance of the proposed dynamic graph construction method when applied to graph-based label propagation for label inference of multiple observations having the same unknown label. This constitutes a test bed for the proposed dynamic graph construction method since the propagation is invoked whenever unknown observations should be classified using a fixed labeled data subset together with their corresponding graph. Let C denote the total number of classes. Let $\mathbf{X}_u$ (a $D \times r$ matrix) denote the r unknown images/observations. Let $\mathbf{X}_l$ (a $D\times N$ matrix) be the N known observations (i.e., the training samples). The concatenation of these two data sets yields the data matrix $\mathbf{X} = (\mathbf{X}_l, \mathbf{X}_u)$. $\mathbf{X}_l$ is the data matrix associated with labeled samples. Let $\mathbf{Y} = (\mathbf{Y}_l, \mathbf{Y}_u)$ (a $C\times(N+r)$ matrix) denote the corresponding

Data: Database matrix $\mathbf{X}$, its affinity matrix $\mathbf{W}$ and new sample matrix $\mathbf{X}'$ having r samples
Result: New affinity matrix $\widetilde{\mathbf{W}}$ associated with $\widetilde{\mathbf{X}} = \mathbf{X} \cup \mathbf{X}'$

for $i = 1, \ldots, r$ **do**
  New sample coding phase;
  Code $\mathbf{x}'_i$ respect to the $\widetilde{\mathbf{X}}$;
  Copy the obtained coefficients to the corresponding row of $[\widetilde{\mathbf{W}}_{nd}\widetilde{\mathbf{W}}_{nn}]$;
  Updating phase ;
  According to the obtained coefficients, select the set of samples S whose edges should be updated;
  for $j = 1, \ldots, Q$ **do**
    Code $\mathbf{x}_{s_j}$ respect to the $\widetilde{\mathbf{X}}$;
    Copy the obtained coefficients to the corresponding row of $[\widetilde{\mathbf{W}}_{dd}\widetilde{\mathbf{W}}_{dn}]$;
  end
  for *Each non-selected sample of* $\mathbf{X}$ **do**
    Copy the corresponding row from $\mathbf{W}$ to $\widetilde{\mathbf{W}}_{dd}$;
    Set the corresponding rows in $\widetilde{\mathbf{W}}_{dn}$ to zero;
  end
end

Algorithm 2: Proposed dynamic graph construction.

label matrix. Each column vector $\mathbf{y}_i$ of $\mathbf{Y}$ is a vector encoding the probabilities of the datum $\mathbf{x}_i$ belonging to different classes, namely,

$$y_i(c) = p(c|\mathbf{x}_i); c = 1, 2, \ldots, C$$

where $p(c|\mathbf{x}_i)$ is the posterior probability of the class c for the datum $\mathbf{x}_i$. For a labeled datum $\mathbf{x}_i$, $y_i(c) = 1$ if $\mathbf{x}_i$ belongs to the c^{th} class; $\mathbf{y}_i(c) = 0$, otherwise. Label propagation tries to estimate the label matrix $\mathbf{Y}_u$ by using the whole data and the known label matrix $\mathbf{Y}_l$. For instance, this can be carried out by minimizing the following label smoothness criterion (See [11]):

$$E(\mathbf{Y}) = \sum_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij} = trace(\mathbf{Y}\,\mathbf{L}\,\mathbf{Y}^T) \quad (6)$$

where $\mathbf{W}$ is the affinity matrix of the graph associated with whole data $\mathbf{X} = (\mathbf{X}_l, \mathbf{X}_u)$, $\mathbf{D}_{row}$ is a diagonal matrix whose diagonal elements are the row sums of the corresponding rows of $\mathbf{W}$, and $\mathbf{D}_{col}$ is a diagonal matrix whose diagonal elements are the column sums of the corresponding columns of $\mathbf{W}$. $\mathbf{D}_{row} - \mathbf{W}$ and $\mathbf{D}_{col} - \mathbf{W}^T$ are the row and column Graph Laplacian matrices, respectively. Note that the matrix $\mathbf{L} = \mathbf{D}_{row} + \mathbf{D}_{col} - (\mathbf{W} + \mathbf{W}^T)$ is symmetric. Since the r images/observations have the same unknown label (r is the number of unlabeled samples in $\mathbf{X}_u$), the unknown label matrix $\mathbf{Y}_u$ will possess C different configurations $(\mathbf{Y}_u(1), \ldots, \mathbf{Y}_u(C))$ where $\mathbf{Y}_u(c)$ has only the c^{th} row equal to ones and the the rest of the rows are zeros. Thus, $\mathbf{Y} = (\mathbf{Y}_l, \mathbf{Y}_u)$ can be written as $\mathbf{Y} = (\mathbf{Y}_l, \mathbf{Y}_u(c))$ where $\mathbf{Y}_l$ is constant matrix. To estimate the unique label of the unknown observations $\mathbf{X}_u$, we use the following:

$$c^\star = \arg\min_c E(\mathbf{Y}_c) \quad (7)$$

where $\mathbf{Y}(c) = (\mathbf{Y}_l, \mathbf{Y}_u(c))$. Therefore, the optimal label is estimated by carrying out C evaluations of the criterion $E(\mathbf{Y}_c)$. The process of multi-observation recognition based on the dynamic TPWRLS graph is shown in Algorithm 3. In this algorithm the unlabeled samples are considered as new samples that should be added to the existing graph associated with the labeled samples $\mathbf{X}_l$.

Data: A set of multiple images/observations $\mathbf{X}_u$, a training set $\mathbf{X}_l$ and their labels $\mathbf{Y}_l$
Result: The label of the unknown observations c*

Construct the dynamic TPWRLS graph, $\mathbf{W}$, over the data $(\mathbf{X}_l, \mathbf{X}_u)$ using Algorithm 2
Infer the label $c^\star$ using Eq. (7)

Algorithm 3: Multi-observation recognition via dynamic graph based label propagation.

B. Datasets

Extended Yale[1]**:** We use the cropped version which contains 1774 face images of 28 individuals. The images of the cropped version contain illumination variations and facial expression variations. The images size is 192×168 pixels with 256-bit grey scale. The images are rescaled to 32×32 pixels in our experiments.

Honda Video (HVDB) database has been acquired for the purpose of face tracking and recognition [12]. It depicts persons sitting in front of a camera in a totally uncontrolled environment and performing unconstrained in-plane and out-of-plane head motion. The used dataset contains 2317 images organized in 22 classes, with an average of 100 images per class.

USPS Handwritten Digits[2]: This dataset contains 11000 grayscale images of handwritten digits. This dataset has ten classes "0" through "9". Each class has 1100 images. The image size is 16 × 16.

C. Experimental setup

Each dataset is randomly split into two parts: a training part T_{train} and a test part T_{test}. From the latter, and for a given number of multiple observations, r, we generate a sequence of subsets called T_m where each subset has r samples that belong to the same class. It is clear that we have $card(T_m) = round(card(T_{test})/r)$. For recognition using the dynamic TPWRLS graph, the unknown label of each element $T_m(i), i = 1, ..., card(T_m)$, a subset of r observations, is inferred separately using the following two steps. Firstly, the dynamic TPWRLS graph is built from the union $T_{train} \cup T_m(i)$

[1] $http://vision.ucsd.edu/\sim iskwak/ExtYaleDatabase/ExtYaleB.html$

[2] $www.cs.nyu.edu/\sim roweis/data.html$

using Algorithm 2 for which the new data samples $\mathbf{X}'$ are given by the r samples in $T_m(i)$. this process is depicted in Figure **??**. Secondly, the label of the subset $T_m(i)$ is then inferred using Algorithm 3. The recognition rate is finally evaluated for the whole set T_m using the ground-truth labels. This process is repeated over ten different random splits. The reported recognition rates correspond to an average over these ten splits. For each dataset, the feature vectors (image descriptors) are first normalized using the zero-mean and unit-variance normalization, then the dimensionality of the obtained data is reduced using Principal Component Analysis for which the number of principal components is set to $min(D, N-1)$ where N is the number of training samples and D is the sample dimension. In the sequel, unless it is stated otherwise, the evaluation will use PCA dimensionality reduction and ten random splits.

D. Experimental results

1) Classification accuracy: Tables I, II, and III show the recognition rates obtained with Extended Yale, HVDB, and USPS datasets for several descriptors and for several numbers of multiple observations. For descriptors, we considered image rawbrightness, Local Binary Patterns (LBP) [13], [14], and Histograms of Oriented Gradients (HOG) [15]. All results are obtained using the dynamic TPWRLS graphs. The Train-Test percentage is 10%-90% for Extended Yale and USPS datasets. Two percentages are used with HVDB dataset: 5%-95% and 10%-90%. From Table I, we can observe that the performance of label propagation increases as the number of observations increases. We can notice that by adding two more observations (r=3) the recognition performance has been significantly improved with respect to the single observation case (r=1). In general, the use of LBP images give the best performance for all used numbers. This can be explained by the fact that LBP images considerably overcome the illumination variation affecting the Extended Yale dataset. Table II illustrates the recognition rates obtained with Honda Video dataset for several numbers of multiple observations (image rawbrightness). The first row corresponds to the Train-Test percentage of 5%-95%, and the second row to 10%-90%. We can observe that by using three observations, the average performance have increased by more than 20% with respect to the use of one single observation (See first and second rows). This can be explained by the fact that the observations correspond to faces undergoing arbitrary 3D motions so ambiguity affecting one single image can be overcome by exploiting several images of the same face. Table III illustrates the recognition rates obtained with the handwritten digit images (USPS dataset) for several numbers of multiple observations. We depict the performance obtained with normalized raw images as well as with several configurations of the HOG descriptor. The number in parenthesis depicts the number of sub-images to which the HOG descriptor is applied. From the table, we can easily see that the use of HOG gave better results than raw images. This is very consistent with the fact that handwritten digits are mainly described by their shape and gradients. We can also observe that good results are obtained with multi-block HOG with overlap. For small number of observations, the use of hybrid descriptors has improved the performance with respect to the use of raw images alone. However, it is still worse than that obtained with HOG descriptor alone.

TABLE I
RECOGNITION RATES FOR EXTENDED YALE DATASET OBTAINED WITH LABEL PROPAGATION OVER TPRWLS GRAPH. THE TRAIN-TEST PERCENTAGE IS 10%-90%.

Descriptor	*r=1*	*r=3*	*r=5*	*r=7*	*r=9*	*r=11*
Raw images	86.8	94.5	96.0	97.5	98.7	97.7
Constr. images	**91.4**	95.3	**97.6**	97.1	97.8	**98.9**
LBP	91.2	97.5	**97.6**	**97.9**	**98.8**	98.5
HOG	88.6	94.3	94.2	95.3	95.4	96.3
LBP + HOG	90.6	95.5	95.9	95.4	97.0	97.6

TABLE II
RECOGNITION RATES FOR HONDA VIDEO DATASET OBTAINED WITH LABEL PROPAGATION OVER TPRWLS GRAPH. THEY CORRESPOND TO AN AVERAGE OVER TEN RANDOM SPLITS.

Raw images	*r=1*	*r=3*	*r=5*	*r=7*	*r=9*	*r=11*
Train-Test 5-95%	53.0	74.6	81.6	85.8	88.2	89.4
Train-Test 10-90%	66.5	87.9	93.2	95.4	96.8	97.6

TABLE III
RECOGNITION RATES FOR USPS DATASET OBTAINED WITH LABEL PROPAGATION OVER TPRWLS GRAPH. THE TRAIN-TEST PERCENTAGE IS 10%-90%.

Descriptor	*r=1*	*r=3*	*r=5*	*r=7*	*r=9*
Raw images	72.5	92.8	97.4	99.1	99.6
HOG (2x2)	83.3	96.7	98.3	99.2	99.6
HOG (3x3)	81.0	96.8	99.2	99.7	99.7
HOG (4x4)	82.4	97.2	**99.4**	**99.7**	**99.9**
HOG (overlap 3x3)	**85.4**	**97.7**	99.3	99.7	99.8
Raw+HOG (2x2)	74.9	93.9	98.1	99.5	99.8
Raw+HOG (3x3)	76.4	94.7	98.4	99.5	99.8
Raw+HOG (4x4)	77.0	94.8	98.2	99.3	99.7
Raw+HOG (overl. 3x3)	77.2	94.9	98.4	99.6	99.7

2) Computational time: Figure 2 compares the CPU time needed for constructing the graph in batch mode (red curves) and in a dynamic mode (blue curves) as a function of the number of updates (how many times new data are inserted into the graph). Here the batch mode refers to the the classic TPWRLS construction method outlined in Algorithm 1. From top to bottom, the results correspond to Extended Yale, HVDB, and USPS datasets. In all plots, the Train-Test percentage is 10%-90%. The computation times are plotted for two different sizes of the new samples. As can be seen, the dynamic graph scheme is much more efficient than the batch graph construction. All tests are conducted using MATLAB that runs

on a PC equipped with an Intel Core i5 CPU at 2.3 Ghz and 4 GB of RAM.

Table IV illustrates the label propagation accuracy obtained by the batch and dynamic graph on the three datasets. The Train-Test percentage is 10%-90%. The results correspond to one split used by both graphs. As can be seen, the dynamic graph have provided almost the same accuracy as the batch graph while it is much more efficient. In same cases, the performance of the dynamic graph is better than that of the batch graph. This can be explained by the fact that by adding novel samples to the graph and by keeping the edges of many nodes unchanged may make the resulting dynamic graph more informative than the batch graph.

TABLE IV
GRAPH-BASED LABEL PROPAGATION ACCURACY (%).

Dataset	r=1	r=3	r=5	r=7	r=9
Ext. Yale					
Batch graph	91.2	94.5	98.4	98.6	97.6
Dynamic graph	91.0	94.9	98.0	99.1	97.0
HVDB					
Batch graph	66.3	88.7	94.3	95.4	98.2
Dynamic graph	66.4	88.3	93.8	94.7	97.7
USPS					
Batch graph	72.6	92.6	97.3	99.6	98.2
Dynamic graph	72.5	92.8	97.4	99.1	99.6

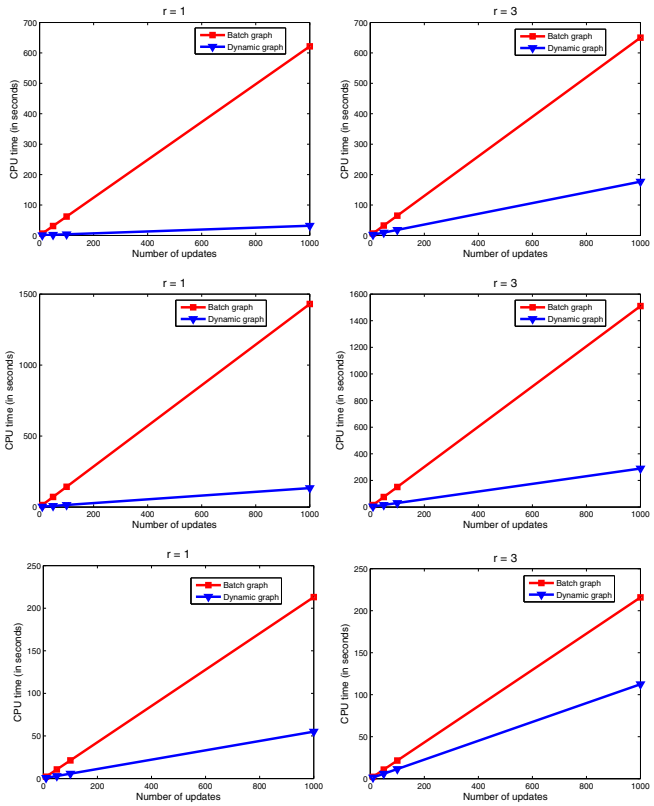

Fig. 2. CPU time needed for batch graph construction (red curve) and dynamic graph (blue curve) as a function of the number of updates. The datasets are Extended Yale, HVDB, and USPS.

V. CONCLUSION

We proposed a new dynamic graph construction method for inductive semi-supervised learning. The proposed method, after receiving new samples, updates the affinity graph dynamically without the need to construct the graph from scratch. Although we adopted TPWRLS coding, any other similarity coding like Gaussian kernel, LLE coding or sparse ℓ_1 coding can be used. The conducted experiments show that, although the proposed method only updates few nodes of the graph, the dynamic construction is computationally much cheaper than the batch graph construction. Furthermore, the dynamic graphs are as informative as the batch ones, in the sense that the performance of label propagation is the almost the same in both cases.

REFERENCES

[1] D. B. West, *Introduction to Graph Theory*, 2nd ed. Prentice Hall, September 2001.

[2] L. Berton and A. De Andrade Lopes, "Graph construction based on labeled instances for semi-supervised learning," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, Aug 2014, pp. 2477–2482.

[3] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. Huang, "Learning with l1-graph for image analysis," *Trans. Img. Proc.*, vol. 19, no. 4, pp. 858–866, Apr. 2010.

[4] T. Jebara, J. Wang, and S. Chang, "Graph construction and b-matching for semi-supervised learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 441–448.

[5] S. Daitch, J. Kelner, and D. Spielman, "Fitting a graph to vector data," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. ACM, 2009, pp. 201–208.

[6] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[7] F. Dornaika and A. Bosaghzadeh, "Adaptive graph construction using data self-representativeness for pattern classification," *Information Sciences*, vol. 325, pp. 118–139, 2015.

[8] W. Liu, J. He, and S. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, June 2010, pp. 679–686.

[9] H. Hacid and T. Yoshida, "Incremental neighborhood graphs construction for multidimensional databases indexing," in *Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, Z. Kobti and D. Wu, Eds., 2007, vol. 4509.

[10] F. Dornaika, A. Bosaghzadeh, H. Salmane, and Y. Ruichek, "Locality constrained encoding graph construction and application to outdoor object classification," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, Aug 2014, pp. 2483–2488.

[11] S. Yan and H. Wang, "Semi-supervised learning by sparse representation," in *SDM*, 2009, pp. 792–801.

[12] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman, "Visual tracking and recognition using probabilistic appearance manifolds," *Computer Vision and Image Understanding*, vol. 99, pp. 303–331, 2005.

[13] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[14] M. Bereta, P. Karczmarek, W. Pedrycz, and M. Reformat, "Local descriptors in application to the aging problem in face recognition," *Pattern Recognition*, vol. 46, pp. 2634–2646, 2013.

[15] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection." in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.