Random Multi-Graphs: A semi-supervised learning framework for classification of high dimensional data☆

Qin Zhang[a,b], Jianyuan Sun[a], Guoqiang Zhong[a], Junyu Dong[a,*]

[a]Department of Computer Science and Technology, Ocean University of China, No. 238 Songling Road, Qingdao 266100, China
[b]Department of Science and Information, Agriculture University of Qingdao, No. 700 Changcheng Road, Qingdao 266109, China

ARTICLE INFO

Article history:
Received 14 May 2016
Received in revised form 22 July 2016
Accepted 16 August 2016
Available online 26 August 2016

Keywords:
Semi-supervised learning
Graph
Regularization
Randomness

ABSTRACT

Currently, high dimensional data processing confronts two main difficulties: inefficient similarity measure and high computational complexity in both time and memory space. Common methods to deal with these two difficulties are based on dimensionality reduction and feature selection. In this paper, we present a different way to solve high dimensional data problems by combining the ideas of Random Forests and Anchor Graph semi-supervised learning. We randomly select a subset of features and use the Anchor Graph method to construct a graph. This process is repeated many times to obtain multiple graphs, a process which can be implemented in parallel to ensure runtime efficiency. Then the multiple graphs vote to determine the labels for the unlabeled data. We argue that the randomness can be viewed as a kind of regularization. We evaluate the proposed method on eight real-world data sets by comparing it with two traditional graph-based methods and one state-of-the-art semi-supervised learning method based on Anchor Graph to show its effectiveness. We also apply the proposed method to the subject of face recognition.

1. Introduction

High dimensional data classification problems have been ubiquitous due to significant advances in computing technology, i.e. bag-of-words representation of documents classification with a huge dictionary, gene expression classification, multimedia classification, etc. High dimensionality poses significant mathematical challenges to traditional classification methods because of computational time and space complexity. We analyze two difficulties in high dimensional data processing. One difficulty is the problem of inefficient similarity measure. Actually it makes sense for Euclidean distances in about 2 to 10 dimensional spaces, which is usually used as similarity measure between data points. However the comparability of Euclidean distance between data points does not exist in high dimensional space due to the sparsity of high dimensional data. Its effectiveness as the similarity measure declines along with the increase of the dimensions. The other difficulty is the so-called *Curse of Dimensionality*. Increasing dimensions can bring about an explosive growth in calculation time and memory space.

Two common methods to deal with high dimensional data are *Dimensionality Reduction* and *Feature Selection*. Dimension Reduction techniques try to obtain a low dimensional embedding from high dimensional data, which is essentially divided into two categories: the linear methods and the nonlinear methods. Linear methods such as Principle Component Analysis (PCA), Independent Component Analysis (ICA) and Linear Discriminant Analysis (LDA) and nonlinear methods such as Laplacian Eigenmaps (LE), Local Linear Embedding (LLE), MultiDimensional Scaling (MDS), Isometric Mapping (ISOMAP), and Kernel PCA (KPCA), are all commonly used. See [1] as a tutorial. Feature Selection techniques try to select the more effective features and eliminate the irrelevant ones [2]. Nowadays research on feature selection focuses on search strategy and evaluation criteria. See [3] as a tutorial. Generally speaking the core idea behind the two methods is to use fewer significant and discriminatory features to represent the original data.

However this paper proposes a different way to handle high dimensional data. We consider the semi-supervised setting based on the following points. Firstly the data acquisition is more and more easy due to technology improvements. Thus the amount of data is becoming more and more large scale with much higher dimensions. This is what we called *Big Data*. Secondly it is very common in application domains that labeled data is scarce and expensive but unlabeled data is large and cheap. Supervised learning does not suit this scenario, while semi-supervised learning which uses a

☆ This paper has been recommended for acceptance by Jiwen Lu.
* Corresponding author.
E-mail addresses: zhang_qin1005@163.com (Q. Zhang), sunjianyuan11@163.com (J. Sun), gqzhong@ouc.edu.cn (G. Zhong), dongjunyu@ouc.edu.ch (J. Dong).

large amount of unlabeled data to help improve the classification performance was born for this purpose.

To solve the difficulties of high dimensional data processing mentioned above, we observe two facts. One is the Random Forests [4] method, it can handle high dimensional data without dimensionality reduction or feature selection. It has good generalization performance and is not easy to overfit due to the randomness. We adopt a similar idea to inject randomness in graphs by randomly select a subset of features to create a graph. The size of the feature subset is usually far smaller than the original dimension. Thus in this selected feature space similarity measure based on Euclidean distance can be effective again due to its lower dimensions. Another fact is the subset-based large scale graph construction method Anchor Graph [5]. It uses a small subset of data points called anchors to construct the whole graph of a data set. It scales linearly with the size of the data set and can deal with very large scale data sets. We utilize the Anchor Graph method to construct a graph in the randomly selected feature space and do semi-supervised inference on this graph. So combing the idea of Random Forests with semi-supervised learning based on Anchor Graph, we propose a new semi-supervised framework named Random Multi-Graphs to deal with high dimensional and large scale data problem. We randomly select a subset of features and use Anchor Graph to construct a graph. The above process is repeated to obtain multiple graphs which can be implemented in parallel to ensure the runtime efficiency and then the multiple graphs vote to determine the labels for the unlabeled data.

We evaluate the proposed method on eight real-world data sets; compared with two traditional graph-based methods and one state-of-the-art semi-supervised learning method based on Anchor Graph to show its effectiveness. As an application of the proposed method, we will analyze data for the purpose of face recognition from images.

The main contributions of this paper are as follows:

- We present a new graph-based semi-supervised learning framework to handle high dimensional and large scale data problem by injecting randomness to the graph.
- We show that the randomness can be viewed as a kind of regularization technique to avoid the curse of dimensionality and overfitting.
- Experiments show the performance increase in high dimensional data problem.

The rest of this paper is organized as follows. In Section 2 we give a brief introduction to graph-based semi-supervised learning, Random Forests and Anchor Graph. The proposed framework is described in Section 3. In Section 4, the experiments are presented, and Section 5 concludes this paper.

2. Background

In this section, we briefly review three related topics: 1) graph-based semi-supervised learning framework; 2) Random Forests; and 3) Anchor Graph.

2.1. Graph-based semi-supervised learning framework

Graph-based semi-supervised learning (GSSL) methods have been successfully used in large number of applications [6–14] covering many application domains where labeling data is very expensive and time-consuming while unlabeled data is very cheap and easy to collect. The GSSL methods use an undirected graph to model the data set and the relationships among the data points. Given a data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l} \bigcup \{(\mathbf{x}_j)\}_{j=l+1}^{l+u}$, the vertices of the graph represent the l labeled data $(\mathbf{x}_i)_{i=1}^{l}$ and u unlabeled data points $(\mathbf{x}_j)_{j=l+1}^{l+u}$. The edges with weights represent the similarity between the affiliated nodes, and are usually represented by a weight matrix $\mathbf{W} = \{w_{ij}\}, i,j = 1\cdots n, n = l+u$. Once the graph is built, the label information is injected into the graph and propagated throughout the whole graph to obtain the labels for the unlabeled data. The main idea is that if the weight w_{ij} is large, then the labels of the adjacent vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ are expected to be the same. This is called smoothness assumptions (cluster assumption or manifold assumption). Traditional graph-based semi-supervised methods include MinCut [15], GRHF [16], LLGC [17], and MR [18], etc. See [19–22] for tutorials.

These traditional graph-based semi-supervised learning methods can be formulated as, or are closely related to the following quadratic optimization problem [23],

$$\min_{\mathbf{f}} (\mathbf{f}-\mathbf{y})^T\mathbf{C}(\mathbf{f}-\mathbf{y}) + \mathbf{f}^T\mathbf{L}\mathbf{f}, \tag{1}$$

where the vector $\mathbf{y} = (y_1, \cdots, y_l, 0, \cdots, 0)^T \in \mathbb{R}^n, y_i \in \{+1, -1\}$, $\mathbf{f} \in \mathbb{R}^n$ is the predicted labels, $\mathbf{L} \in \mathbb{R}^{n\times n}$ is the regularization matrix, and $\mathbf{C} \in \mathbb{R}^{n\times n}$ is a diagonal matrix with its ith diagonal elements $c_i = \alpha_l > 0$ for $1 \le i \le l$ and $c_i = \alpha_u \ge 0$ for $l+1 \le i \le l+u$. α_l and α_u are two parameters.

The first term of Eq. (1) is the fitting term which uses quadratic loss to measure the error between the predicted labels $\mathbf{f}$ and the known labels $\mathbf{y}$. The second term of Eq. (1) is the regularization term which uses a regularization matrix $\mathbf{L}$ to evaluate how smooth the prediction $\mathbf{f}$ varies along the data manifold which is usually represented by a graph.

The popular choice for L is the graph Laplacian [24]. It is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{2}$$

where $\mathbf{W}$ is the weight (similarity) matrix of the graph and commonly computed by the Gaussian Kernel as

$$w_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} i \ne j, w_{ii} = 0 \tag{3}$$

where σ is the kernel width parameter to be tuned. The diagonal matrix $\mathbf{D}$ is the row sum of $\mathbf{W}$ with its ith diagonal element $d_i = \sum_{j=1}^{n} w_{ij}$. The normalized graph Laplacian is the normalized version of L which is defined as

$$\widetilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{\frac{1}{2}}. \tag{4}$$

Computing the partial derivative w.r.t $\mathbf{L}$ and letting it equal to 0, a closed-form solution of the problem (1) can be obtained as

$$\mathbf{f} = (\mathbf{L} + \mathbf{C})^{-1}\mathbf{C}\mathbf{y}. \tag{5}$$

Then the predicted labels can be easily obtained from $\mathbf{f}$ by looking at the signs of the predicted values $y_i = sgn(f_i), l+1 \le i \le l+u$.

It is easy to extend the objective function (1) to deal with multi-class classification problems. For a c-class problem, one can use the one-in-c representation to represent the labels. The objective function for multi-class semi-supervised learning is defined as

$$\min_{\mathbf{F}} tr((\mathbf{F}-\mathbf{Y})^T\mathbf{C}(\mathbf{F}-\mathbf{Y})) + tr(\mathbf{F}^T\mathbf{L}\mathbf{F}), \tag{6}$$

where $tr(\cdot)$ is the trace function, $\mathbf{F} \in \mathbb{R}^{n\times c}$ is the prediction matrix to be learned, $\mathbf{Y} = (\mathbf{y}_1, \cdots, \mathbf{y}_l, \mathbf{0}, \cdots \mathbf{0})^T \in \mathbb{R}^{n\times c}$, $\mathbf{y}_i$ is the label vector for

data point $\mathbf{x}_i$ and it is an indicator vector with $y_{ij} = 1$ if $\mathbf{x}_i$ belongs to the jth class and $y_{ij} = 0$ otherwise. $\mathbf{0}$ is the all 0 vector. Then each $\mathbf{x}_i$ can be classified to the jth class if F_{ij} is the largest one in the ith row of $\mathbf{F}$ which can be written as $y_i = \text{argmax}_j \mathbf{F}_{ij}, j = 1 \cdots c$.

Note that problem (1) represents several famous traditional graph-based learning methods summarized in Table 1.

However the traditional graph-based semi-supervised learning methods are not suitable for high dimensional and large scale problems due to the following reasons. Firstly, the accuracy depends a lot on the graph which describes the relationships among data points, and reflects the true structure (manifold) of the data set. However in high dimensional feature space the data points are much more sparse, the distance-based measure loses its effectiveness to measure the (dis)similarity between points. Secondly, they do not scale well for high dimensional and large scale data set because of the high complexity involved. It can be seen that the major processes in graph-based semi-supervised learning methods are graph construction and an n-by-n matrix inversion which needs $O(n^2)$ and $O(n^3)$ respectively. We will solve these two problems in our proposed framework.

2.2. Random Forests

Breiman gave the definition of Random Forests in [4] that a Random Forests is a classifier consisting of a collection of tree-structured classifiers $\{h(x, \theta_k), k = 1, \ldots\}$ where the $\{\theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x. It can be seen that Random Forests is an ensemble approach whose main principle is that a group of weak learners can come together to form a strong learner. It starts with a standard machine learning technique called decision tree which, in ensemble terms, corresponds to the weak learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets bucketed into smaller and smaller sets.

Here is how such a Random Forests is trained [25]. Given a data set with N samples, and for some given number of trees T:

- Sample N cases at random with replacement to create a subset of the data. The subset should be about 66% of the total set.
- At each node: For some number M, select M features at random from all the features. Use the M features that provide the best split, according to some objective function, to do a binary split on that node. The value of M is held constant during the forest growing. At the next node, choose another M features at random and do the same until a tree has been created. Note that there is no pruning.
- Repeat the above process until T trees have been created.

When a new input is entered into the Random Forests, it runs down all of the trees. The result may either be an average or weighted average of all of the terminal nodes that are reached, or, in the case of categorical variables, a voting majority.

Some benefits of Random Forests are listed below.

- High accuracy;
- Runs efficiently on large data sets;
- Handles thousands of input features without feature deletion and selection;
- Gives estimates of what features are important in the classification;
- Easy for distributed processing;
- etc.

The drawback of Random Forests is that with imbalanced training sets the classification results tend to the categories which have more samples, so the number of training samples of all categories must be the same. Chen et al. [26] considered the problem and gave different weights on the judgment of the decision trees according to the proportion of samples, but the experiment results showed that this method was still not good enough at solving the imbalance problem. Another drawback of Random Forests is that the more trees a Random Forests has, the more stable the performance is. If the number of trees is not high enough, the performance may be not stable.

Table 1
Choices of graph Laplacian and parameters in three famous graph-based methods.

Methods	Choice of $\mathbf{L}$	α_l	α_u
GRHF	Graph Laplacian	∞	0
LapRLS	Graph Laplacian	A finite number	0
LLGC	Normalized graph Laplacian	$0 < \alpha_l = \alpha_u < \infty$	

2.3. Anchor Graph

The idea of Anchor Graph is from [27,28]. They worked with large scale data and made the label prediction function to be a weighed average of the labels on a subset of anchor (landmark) samples. As such, the label prediction function f can be represented by a subset $\mathbf{A} = \{a_j\}_{j=1}^m \subset \mathbb{R}^d$ in which each a_j is an anchor point,

$$f(x_i) = \sum_{j=1}^{m} \mathbf{P}_{ij} f(a_j), \tag{7}$$

where P_{ij} is the data-adaptive weight. If we define two vectors $\mathbf{f} = [f(x_1), \ldots, f(x_n)]^T$ and $\mathbf{f}_a = [f(a_1), \ldots, f(a_m)]^T$, then Eq. (7) can be rewritten as

$$\mathbf{f} = \mathbf{P}\mathbf{f}_a, \mathbf{P} \in \mathbb{R}^{n \times m}, m \ll n. \tag{8}$$

This formula reduces the solution space of unknown labels from the larger space $\mathbf{f}$ to a smaller space $\mathbf{f}_a$. The problem here is how to choose the anchor points. Liu et al. [5] suggested the use of K-means clustering centers as anchors instead of randomly sampled points because K-means clustering centers have a stronger representation power to adequately cover the full data set.

Another problem here is how to design the matrix $\mathbf{P}$. Liu et al. [5] proposed a method called *Local Anchor embedding* (LAE) to reconstruct any data point x_i as a convex combination of its closest anchors. The data-anchor mapping problem can be formulated as follows:

$$\begin{aligned} &\min_{\mathbf{P} \in \mathbb{R}^{n \times m}} J(\mathbf{P}) = \tfrac{1}{2} \| \mathbf{X} - \mathbf{P}\mathbf{A} \|^2 \\ &s.t. \quad \mathbf{P}_{ij} \geq 0, \mathbf{P}_i \mathbf{1} = 1, \end{aligned} \tag{9}$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ stands for the data matrix in which every row is a data sample, $\mathbf{A} \in \mathbb{R}^{m \times d}$ is the anchor matrix in which every row is an anchor, $\mathbf{P} \in \mathbb{R}^{n \times m}$ is the data-anchor mapping matrix which is to be learned.

Using matrix $\mathbf{P}$, the adjacency matrix can be designed as $\mathbf{W} = \mathbf{P}\mathbf{\Lambda}^{-1}\mathbf{P}^T$ in which the diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{m \times m}$ is defined as $\mathbf{\Lambda}_{kk} = \sum_{i=1}^{n} \mathbf{P}_{ik}$. This is where the name Anchor Graph comes from, because a graph can be fully represented by its adjacency matrix.

3. Random Multi-Graphs

Fig. 1 shows the proposed method. The details are presented below.

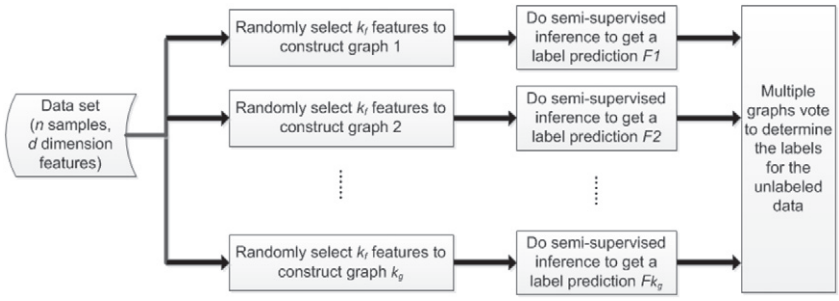


Fig. 1. *Random Multi-Graphs (RMG)* algorithm is summarized in this figure. First randomly select a subset of k_f features and use Anchor Graph to construct a graph. The above process is repeated to obtain k_g graphs which can be implemented in parallel to ensure the runtime efficiency. And then the multiple graphs vote to determine the labels for the unlabeled data.

3.1. Notations

Given a data set $\mathbf{X} = \mathbf{X}_l \bigcup \mathbf{X}_u \in \mathbb{R}^d$, d is the dimension of the feature space. $\mathbf{X}_l = \{\mathbf{x}_1, \cdots, \mathbf{x}_l\}$ is a labeled set with the labels $y_i \in \{1, \cdots, c\}$, $\mathbf{X}_u = \{\mathbf{x}_{l+1}, \cdots, \mathbf{x}_{l+u}\}$ is an unlabeled set, and u is the number of unlabeled data. $n = l + u$ is the size of the total data set.

3.2. Algorithm

The whole framework of Random Multi-Graphs is described as below.

- Step 1: Randomly select k_f features from all the d features;
- Step 2: Create a graph: Choose m anchor points to cover the data manifold denoted by an anchors matrix $\mathbf{A}$; compute the mapping matrix $\mathbf{P}$ to represent the rest of the data points via anchors;
- Step 3: Run semi-supervised inference on this graph utilizing graph Laplacian Regularization;
- Step 4: Repeat the above steps to get k_g graphs;
- Step 5: k_g graphs vote to determine the labels for the unlabeled data points.

3.3. Graph construction

Graph construction is essential in graph-based semi-supervised learning. The most common method is to build a full-connected graph using Gaussian kernel described in Eq. (4). Other sparse graphs such as k-NN and ϵ-NN are also commonly used. Even more accurate graphs can be learned via the metric learning method [29–32] which can effectively use label information in graph construction.

In this paper we adopt the Anchor Graph method to construct the graph because of its better efficiency. While the optimization of mapping matrix described in Eq. (9) is slow [33]. Here we use a predefined way to design the mapping matrix $\mathbf{P}$ by Nadaraya-Watson kernel regression [5,34] that defines P_{ik} as

$$\mathbf{P}_{ik} = \frac{K_\sigma(x_i, a_k)}{\sum_{k' \in \mathbf{A}_{<i>}} K_\sigma(x_i, u_{k'})} \quad \forall a_k \in \mathbf{A}_{<i>}, \tag{10}$$

where K_σ is a kernel function with a bandwidth σ which we use Gaussian kernel function as common, and $\mathbf{A}_{<i>}$ is the r nearest anchors of $\mathbf{x}_i$. Intuitively we set $\mathbf{P}_{ik} = 0 \; \forall \mathbf{a}_k \notin \mathbf{A}_{<i>}$. The assumption here is that $\mathbf{P}_{ik}$ should be larger if $\mathbf{x}_i$ is close to $\mathbf{a}_k$, and vice versa.

Using matrix $\mathbf{P}$, the graph can be represented by its adjacency matrix $\mathbf{W} = \mathbf{P}\mathbf{\Lambda}^{-1}\mathbf{P}^T$ in which the diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{m \times m}$ is defined as $\mathbf{\Lambda}_{kk} = \sum_{i=1}^{n} P_{ik}$.

3.4. Semi-supervised inference

We adopt the traditional semi-supervised learning objective function described in Eq. (6), and use the same semi-supervised learning framework as Liu et al. [5]. Based on the anchor-based label prediction model described in Eq. (8), we only need to solve the labels for the anchors denoted by a one-in-c label matrix $\mathbf{F}_A = \{\mathbf{a}_1, \cdots, \mathbf{a}_m\}^T \in \mathbb{R}^{m \times c}$. The labels of other data points can be represented by $\mathbf{F} = \mathbf{P}\mathbf{F}_A$. Then the formulated objective function is as follows

$$\begin{aligned} Q(\mathbf{F}_A) &= tr\left((\mathbf{F}_l - \mathbf{Y}_l)^T(\mathbf{F}_l - \mathbf{Y}_l)\right) + \gamma tr(\mathbf{F}^T\mathbf{L}\mathbf{F}) \\ &= \| \mathbf{P}_l\mathbf{F}_A - \mathbf{Y}_l \|_F^2 + \gamma tr\left((\mathbf{P}\mathbf{F}_A)^T\mathbf{L}(\mathbf{P}\mathbf{F}_A)\right) \\ &= \| \mathbf{P}_l\mathbf{F}_A - \mathbf{Y}_l \|_F^2 + \gamma tr\left(\mathbf{F}_A^T\mathbf{L}_r\mathbf{F}_A\right) \end{aligned} \tag{11}$$

where $\mathbf{P}_l$ is the sub-matrix of $\mathbf{P}$ according to the labeled partition, $\mathbf{Y}_l$ is the labels of the labeled data, $\|\cdot\|_F$ is the Frobenius norm of the matrix, and γ is a regularization parameter. We define $\mathbf{L}_r = \mathbf{P}^T\mathbf{L}\mathbf{P}$, $\mathbf{L}$ is the graph Laplacian matrix defined as $\mathbf{L} = \mathbf{D} - \mathbf{W} = \mathbf{I} - \mathbf{P}\mathbf{\Lambda}^{-1}\mathbf{P}^T$. Then the computation of $\mathbf{L}_r$ is as follows

$$\mathbf{L}_r = \mathbf{P}^T\mathbf{L}\mathbf{P} = \mathbf{P}^T(\mathbf{I} - \mathbf{P}\mathbf{\Lambda}^{-1}\mathbf{P}^T)P = \mathbf{P}^T\mathbf{P} - (\mathbf{P}^T\mathbf{P})\mathbf{\Lambda}^{-1}(\mathbf{P}^T\mathbf{P}). \tag{12}$$

Obviously a global optima for the objective function described in Eq. (11) can be easily obtained by setting the partial derivative w.r.t $\mathbf{F}_A$ to zero:

$$\mathbf{F}_A^* = \left(\mathbf{F}_l^T\mathbf{F}_l + \gamma\mathbf{L}_r\right)^{-1}\mathbf{F}_l^T\mathbf{Y}. \tag{13}$$

Then the labels of unlabeled data can be determined by

$$y_i = \arg\max_{j \in 1, \cdots, c} \; \mathbf{P}_{i.}\mathbf{F}_{Aj.}, i = l+1, \cdots, n, \tag{14}$$

where $\mathbf{P}_{i.}$ denotes the ith row of $\mathbf{P}$, $\mathbf{F}_{Aj.}$ denotes the jth column of $\mathbf{F}_A$.

3.5. Inductive extension

After solving the labels of anchors, a very natural inductive extension is that if a new data $\mathbf{x}_{new}$ comes, we first compute the mapping weights by Eq. (10), then using Eq. (14), we can get the label of $\mathbf{x}_{new}$. Although, for high dimensional data, to avoid the curse of dimensionality, it still needs to go throughout k_g graphs and vote to determine the labels for $\mathbf{x}_{new}$.

3.6. Randomness as regularization

The core idea of our proposed framework is injecting randomness into the graphs. More particularly, we select a random subset of features to create a graph. We illustrate two advantages of this approach. One is that it can deal with the high dimensional data very well, the other is that it can also help to avoid overfitting.

For the former, we adopt the idea of *divide and conquer* to only use a small subset of features to participate the construction of a graph, and use multiple (i.e. 100) graphs to make up the representation deficiency caused by fewer features. Thus every graph is constructed in a relatively low feature space. Using this method we can avoid two of the problems caused by high dimensional data sets: the effectiveness of Euclidean distance and high complexity .

For the latter, we show that it can be regarded as a kind of regularization. We all know that if we have too many features the learned model may fit the training set very well, but fail to generalize to new examples. This is called *overfitting*. The most common solution to this problem is *regularization*. That is, to keep all the features but reduce the magnitude of the parameters. This works well when there are a lot of features, each of which contributes a bit to train the model. This means we minimize the empirical error and we add a series of penalties to the parameters of the model. Another interpretation to the penalties is to introduce some priors to the model parameters. In this sense our approach can also be regarded as a kind of regularization because we select a small subset of features to construct a graph by penalizing the weights for the other features to become zero. Also there are some similar statements from previous works by other authors. In [4] it has been stated that the two kind of randomness can help to avoid overfitting. In [35,36] it has also been confirmed that the randomness introduced in the model can be viewed as a kind of regularization.

4. Experiments

We evaluate the proposed method on eight real-world data sets to show its effectiveness.

4.1. Data sets

We use eight real-world data sets in our experiments, which are USPS data set[1], 20Newsgroups data set[2], YaleB data set[3,4], and five data sets from the UCI machine learning data sets[5] [37]. These data sets come from various application domains; including image processing, text classification, face recognition, handwritten recognition, medical diagnosis and protein analysis. The feature dimensions of these data sets vary from tens, to hundreds, to thousands. Table 2 shows the properties of these eight data sets.

4.2. Experimental results

We ran the experiments in the computing environment: Intel(R) Xeon(R)CPU E5506@2.13 GHz, 72 GB memory, Windows Server 2012 64 bit operation system, and Matlab version 8.4.0.

We compared the proposed method Random Multi-Graphs (RMG for short) to the state-of-the-art method AnchorGraphReg [5] and two traditional graph-based semi-supervised learning methods: Gaussian Field and Harmonic Function (GFHF) [16] and Learning Local and Global Consistency (LLGC) [17] on both classification accuracy and runtime. The experimental results are shown in Tables 3 and 4. For every data set, we ran each method 20 times and for each run we randomly chose 10% of the data as labeled. The following tables report the average performance, the boldface items in these tables represent the best performance.

Table 3 shows the average classification accuracy. It can be seen from Table 3 that our proposed method reports the best average accuracy on these eight data sets. And the traditional graph-based semi-supervised methods are not perform well especially on high dimensional data sets.

Table 4 shows the average runtime. We didn't get the result of 20Newsgroups data set for GFHF and LLGC because of too long runtime. It can be seen from Table 4 that our proposed method does not perform well in runtime because we create multiple graphs (100 in our experiments) to vote. Actually these multiple trees are independent of each other. Thus it can be implemented in parallel. We use one GeForce GTX TITAN X GPU to accelerate the proposed method and the results are listed in the last column of Table 4. It can be seen that runtime of small data sets like Protein and DBWorld emails are increasing, mainly because the expenditure of data transmission exceeds the computational acceleration. It can achieve 2–5 times of speed-up ratio for the rest of the data sets. It is not enough of course. We will try some other methods to solve this problem in the future.

Table 2
Eight data sets used in the experiments.

Data set	# of data points	# of dimensions	# of classes
Protein	116	20	6
Image segmentation	2310	19	7
Dermatology	366	34	6
Waveform40	5000	40	3
YaleB	2432	241	38
20Newsgroups	16,242	100	4
USPS	9298	256	10
DBWorld emails	64	4702	2

4.3. Effect of parameters

There are two parameters in our framework: k_g and k_f representing the number of graphs and the number of randomly selected features, respectively.

With respect to k_g, Random Forests is very robust when handling different numbers of trees. We use four data sets to verify whether the proposed method holds the same characteristics as Random Forests or not. The four data sets used in this experiment include Protein, YaleB and DBWorld emails, whose properties can be found in Table 2, and an unbalanced data set Glass with 223 samples of 6 classes and 9 feature dimensions. We choose the values of k_g from a set of $\{5, 10, 50, 100, 500\}$. For every data set and every value of k_g we run the proposed method ten times to get the results. The parameter k_f is fixed as $2sqrt(d)$ in this experiment. Fig. 2 shows the results. We can see from the figure that when k_g is larger than 50 the accuracy is almost the same. Thus the proposed method is also robust as to the number of graphs. So we fix the number of graphs k_g as 100 in our experiments.

Table 3
Average classification accuracy.

Data set	GFHF	LLGC	AnchorGraphReg	RMG
Protein	0.2398	0.2972	0.5130	**0.6167**
Image Segmentation	0.1722	0.1570	0.9006	**0.9158**
Dermatology	0.2406	0.3414	0.9319	**0.9456**
Waveform40	0.3633	0.3770	0.7241	**0.7988**
YaleB	0.1493	0.0692	0.7410	**0.8792**
20Newsgroups	–	–	0.8214	**0.8938**
USPS	0.1480	0.1717	0.9257	**0.9551**
DBWorld emails	0.5415	0.5932	0.8602	**0.8831**

[1] http://statweb.stanford.edu/tibs/ElemStatLearn/data.html.
[2] http://qwone.com/jason/20Newsgroups/.
[3] http://vision.ucsd.edu/leekc/ExtYaleDatabase/YaleFaceDatabase.htm.
[4] http://vision.ucsd.edu/leekc/ExtYaleDatabase/ExtYaleB.html.
[5] http://archive.ics.uci.edu/ml/.

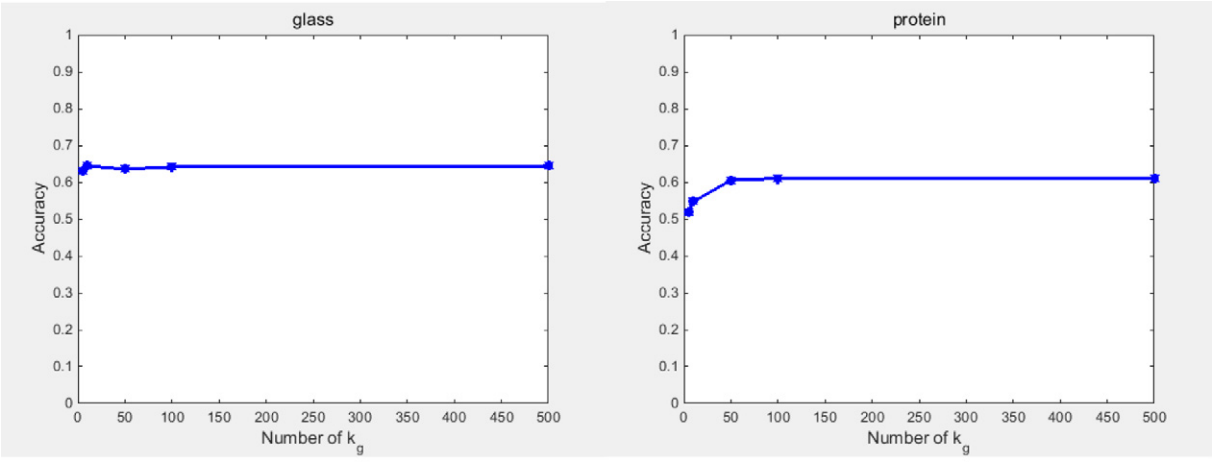
Table 4
Average runtime.

Data set	GFHF	LLGC	AnchorGraphReg	RMG	RMG_GPU
Protein	0.0630	0.0585	**0.0300**	1.5344	4.0456
Image segmentation	17.8723	49.7439	**2.7811**	180.6050	79.9740
Dermatology	0.7001	0.8904	**0.0682**	1.0589	4.3412
Waveform40	129.6673	346.1910	**14.6079**	968.9694	338.9526
YaleB	65.2389	96.1705	**5.6422**	311.6759	135.3131
20Newsgroups	–	–	**597.6964**	6230.7799	2266.1678
USPS	1147.8508	2439.3654	**127.6280**	5993.5319	1390.3187
DBWorld emails	0.2543	0.4963	**0.0950**	1.6619	2.1047

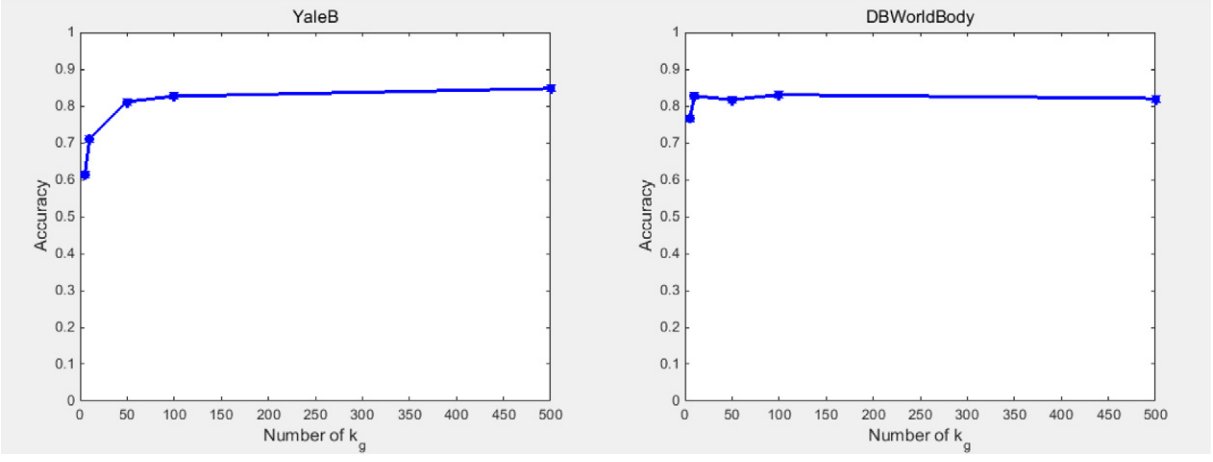
As for k_f, Brieman suggests three possible values for k_f: $\frac{1}{2}\sqrt{m}$, $\sqrt{m}$, and $2\sqrt{m}$ [4]. We try to find the relationship between k_f and the performance on the same four data sets mentioned above. For every data set we run our algorithm ten times with $k_f = \left\{\frac{1}{2}\sqrt{m}, \sqrt{m}, 2\sqrt{m}\right\}$. The number of graphs k_g is fixed as 100. Fig. 3 shows the average accuracy. We can see that the number of features randomly selected for every graph does not affect the performance too much except the YaleB data set. The main reasons are that the number of classes of the YaleB data set is 38 which is much more than the other three data sets, and its feature dimension is 241 which is also a little high. Thus fewer features can not provide enough information to distinguish these many classes. We fixed k_f as $2\sqrt{m}$ in our experiments. But a larger number of randomly selected features is strongly recommended for high dimensional and many classes data sets, i.e. $4\sqrt{m}$ or $8\sqrt{m}$.

4.4. Face recognition application

We also apply the proposed method on the application of face recognition, which is a classical application domain in computer vision. Many researches have been done in this field. Unlike those feature extraction based methods [2,38] or learning accurate features

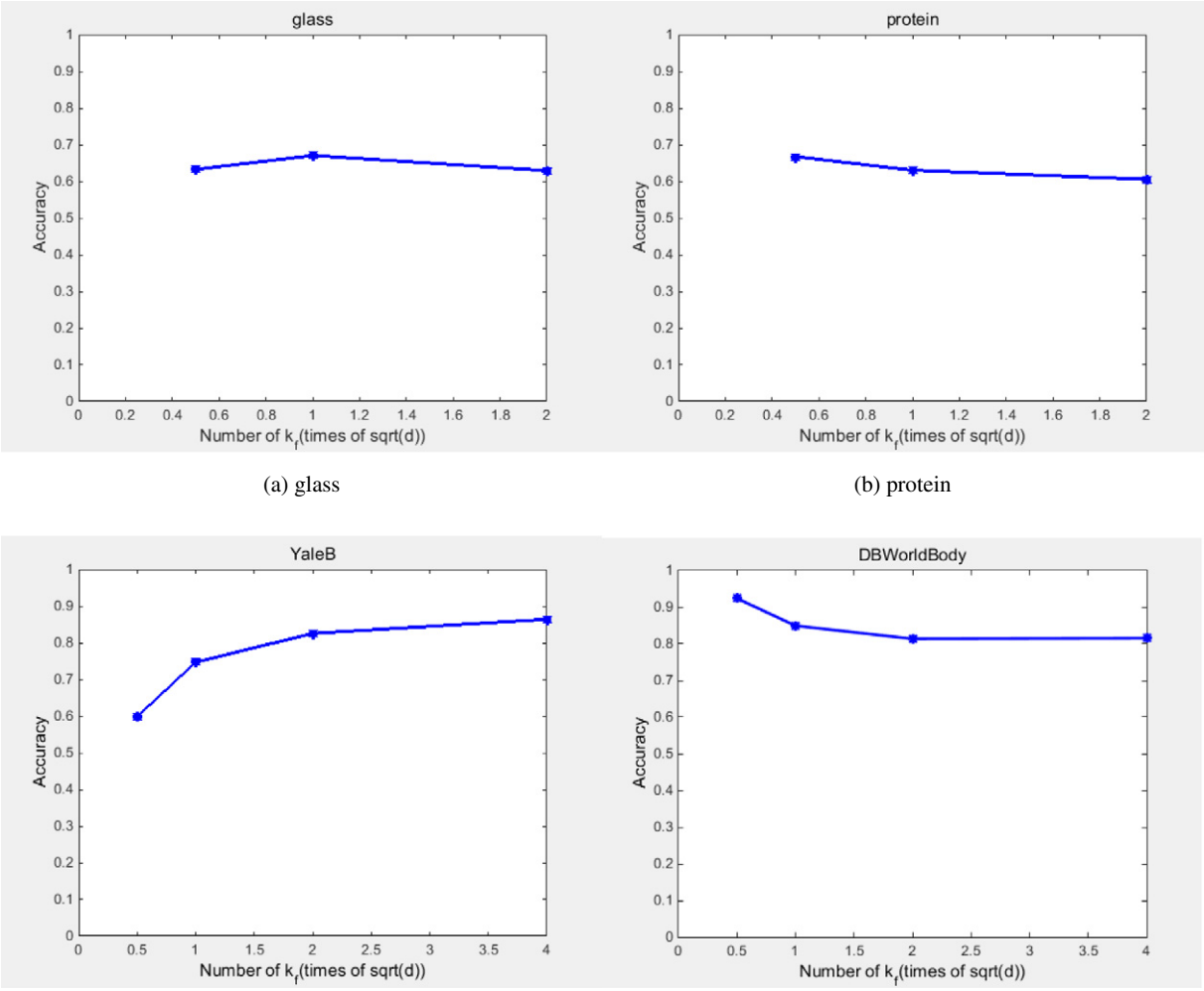


(a) glass (b) protein



(c) YaleB (d) DBWorld emails

Fig. 2. Relationship between the number of graphs and the performance evaluated on four data sets.



(a) glass (b) protein

(c) YaleB (d) DBWorld emails

Fig. 3. Relationship between the number of features and the performance evaluated on four data sets.

for face recognition [39–41], we use the raw pixels of 4096 dimensions directly with 64×64 resolution of every face image in ORL data set.[6] There are ten different images of each of 40 distinct persons in this face data set. For some persons, the images were taken at different times, varying the lighting, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses). We randomly choose 10%, 20%, 30%, and 40% per class (person) as labeled data and the rest is unlabeled. Table 5 shows the average accuracy and average runtime of ten times running.

5. Conclusion and future work

We focused on the graph-based semi-supervised learning of high dimensional data. Combining the ideas of Random Forests and Anchor Graph we propose a new framework to deal with high dimensional data, which can effectively avoid the curse of dimensionality and efficiently obtains better classification accuracy. We randomly choose a subset of the features to create a graph based on anchors, repeat the above process to obtain multiple graphs, then vote to determine the labels of unlabeled data.

We tested the proposed method with traditional and state-of-the-art methods including GFHF, LLGC, and AnchorGraphReg. The experiments show that the proposed method can give an obvious improvement in the classification accuracy with 10% labeled data.

However there is still some work to be done in the future. One thing is the bottleneck of the runtime. We consider to use more GPUs to accelerate the proposed method. Or we can try to learn a "strong" graph with multiple "week" graphs thus to avoid the bottleneck of runtime. Another thing is that we estimate there should exist some certain relationship among the number of randomly selected features k_f, the number of graphs k_g, the feature dimensions d, and the number of classes of the data set c. Further study will be needed to determine if this relationship exists, and its' rules. The next step

Table 5
ORL face recognition results.

ORL (percentage of labeled data)	10%	20%	30%	40%
Ave. accuracy	0.5956	0.7047	0.7411	0.7708
Ave. runtime(s)	7.0307	7.0766	6.9363	7.1025

[6] http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html.

will be to obtain more experiments' results and to try to train a regression model to predict the values of k_f and k_g for a given data set.

Acknowledgments

This work was supported by the National Natural Science Foundation Of China (NSFC) (Nos. 61271405, 61403353), the International Science & Technology Cooperation Program of China (ISTCP) (No. 2014DFA10410), the Open Project Program of the National Laboratory of Pattern Recognition (NLPR) and the Fundamental Research Funds for the Central Universities of China. The Titan X GPU used for this research was donated by the NVIDIA Corporation. We thank Leon Bullock very much for his great assistance in revising the manuscript.

References

[1] I.K. Fodor, A survey of dimension reduction techniques, Neoplasia 7 (5) (2008) 475–485.
[2] G. Zhong, C.-L. Liu, Error-correcting output codes based ensemble feature extraction, Pattern Recogn. 46 (4) (2013) 1091–1100. http://dx.doi.org/10.1016/j.patcog.2012.10.015.
[3] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Comput. Electr. Eng. 40 (1) (2014) 16–28. http://dx.doi.org/10.1016/j.compeleceng.2013.11.024.
[4] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32. http://dx.doi.org/10.1023/A:1010933404324.
[5] W. Liu, J. He, S.-F. Chang, Large graph construction for scalable semi-supervised learning, International Conference on Machine Learning (ICML), 2010. pp. 679–686.
[6] M. Abbasi, H.R. Rabiee, C. Gagné, Monocular 3D human pose estimation with a semi-supervised graph-based method, International Conference on 3D Vision, 2015. pp. 518–526.
[7] A.B. Goldberg, X. Zhu, Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization, Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing, 2006. pp. 45–52.
[8] X. Zeng, D.F. Wong, L.S. Chao, L.F. Han, Chinese named entity recognition with graph-based semi-supervised learning model, ACL-IJCNLP 2015. pp. 15.
[9] R. Liu, J. Zhou, M. Liu, Graph-based semi-supervised learning algorithm for web page classification, International Conference on Intelligent Systems Design and Applications, 2006. pp. 1–5.
[10] B.B. Liu, Z.M. Lu, Image colourisation using graph-based semi-supervised learning, IET Image Process. 3 (3) (2009) 115–120. http://dx.doi.org/10.1049/iet-ipr.2008.0112.
[11] L. Ma, A. Ma, C. Ju, X. Li, Graph-based semi-supervised learning for spectral-spatial hyperspectral image classification, Pattern Recogn. Lett. (2016) http://dx.doi.org/10.1016/j.patrec.2016.01.022.
[12] M. Stikic, D. Larlus, B. Schiele, Multi-graph based semi-supervised learning for activity recognition, 2009 International Symposium on Wearable Computers, IEEE, 2009. pp. 85–92.
[13] J. Tang, H. Li, G.J. Qi, T.S. Chua, Image annotation by graph-based inference with integrated multiple/single instance representations, IEEE Trans. Multimedia 12 (2) (2010) 131–141. http://dx.doi.org/10.1109/TMM.2009.2037373.
[14] Y. Zhao, R. Ball, J. Mosesian, J.F. De Palma, Graph-based semi-supervised learning for fault detection and classification in solar photovoltaic arrays, IEEE Trans. Power Electron. 30 (5) (2013) 1628–1634. http://dx.doi.org/10.1109/TPEL.2014.2364203.
[15] A. Blum, S. Chawla, Learning from labeled and unlabeled data using graph mincuts, Eighteenth International Conference on Machine Learning, 2001. pp. 19–26.
[16] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, International Conference on Machine Learning (ICML), vol. 3, 2003. pp. 912–919.
[17] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, Adv. Neural Inf. Proces. Syst. 16 (16) (2004) 321–328.
[18] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[19] M. Seeger, Learning with labeled and unlabeled data, Epfl. 2000.
[20] O. Chapelle, B. Schölkopf, A. Zien, Semi-supervised Learning, MIT press Cambridge. 2006. http://dx.doi.org/10.1109/TNN.2009.2015974.
[21] X. Zhu, Semi-supervised learning tutorial, International Conference on Machine Learning(ICML), 2007. pp. 1–135.
[22] X. Zhu, A.B. Goldberg, Introduction to semi-supervised learning, Synth. Lect. Artif. Intell. Mach. Learn. 3 (1) (2009) 1–130. http://dx.doi.org/10.2200/S00196ED1V01Y200906AIM006.
[23] M. Wu, B. Schölkopf, Transductive Classification via local learning regularization, International Conference on Artificial Intelligence and Statistics, 2007. pp. 628–635.
[24] F.R. Chung, Spectral Graph Theory, CBMS Regional Conference Series in Mathematics, vol. 92, 1997,
[25] J. Sun, G. Zhong, J. Dong, Banzhaf random forests, Technical Report, Ocean University of China, Department of Computer Science and Technology 2016.
[26] C. Chen, A. Liaw, L. Breiman, Using random forest to learn imbalanced data, Technical Report, University of California, Berkeley 2004.
[27] O. Delalleau, Y. Bengio, N. Le Roux, Nonparametric function induction in semi–supervised learning, In Proc. Artificial Intelligence and Statistics, 2005. pp. 96–103.
[28] X. Zhu, J. Lafferty, Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning, International Conference on Machine Learning(ICML), 2005. pp. 1052–1059.
[29] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (1) (2009) 207–244.
[30] Y. Ying, K. Huang, C. Campbell, Sparse metric learning via smooth optimization, Advances in Neural Information Processing Systems 22: Conference on Neural Information Processing Systems a Meeting Held 7-10 December 2009, Vancouver, British Columbia, Canada.
[31] J.V. Davis, B. Kulis, P. Jain, S. Sra, I.S. Dhillon, Information-theoretic metric learning, NIPS 2006 Workshop on Learning to Compare Examples, 2015. pp. 209–216.
[32] G. Zhong, K. Huang, C.-L. Liu, Low rank metric learning with manifold regularization, IEEE 11Th International Conference on Data Mining (ICDM), 2011. pp. 1266–1271.
[33] Q. Zhang, G. Zhong, J. Dong, Speeding up the spectral clustering via anchors, Technical Report, Ocean University of China, Department of Computer Science and Technology. 2016.
[34] T. Hastie, R. Tibshirani, J. Friedman, J. Franklin, The Elements of Statistical Learning: Data Mining, Inference and Prediction, 2nd ed., Springer. 2009. http://dx.doi.org/10.1111/j.1467-985X.2010.00646_6.x.
[35] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio, Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11 (2010) 625–660.
[36] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (2010) 3371–3408.
[37] M. Lichman, UCI Machine learning repository, 2013, http://archive.ics.uci.edu/ml.
[38] G. Zhong, M. Cheriet, Tensor representation learning based image patch analysis for text identification and recognition, Pattern Recogn. 48 (4) (2015) 1207–1220. http://dx.doi.org/10.1016/j.patcog.2014.09.025.
[39] J. Lu, G. Wang, P. Moulin, Localized multi-feature metric learning for image set based face recognition, IEEE Trans. Circuits Syst. Video Technol. 26 (3). (2015) 1–1. http://dx.doi.org/10.1109/TCSVT.2015.2412831.
[40] J. Lu, V.E. Liong, G. Wang, P. Moulin, Joint feature learning for face recognition, IEEE Trans. Inf. Forensics Secur. 10 (7) (2015) 1371–1383. http://dx.doi.org/10.1109/TIFS.2015.2408431.
[41] J. Lu, V.E. Liong, X. Zhou, J. Zhou, Learning compact binary face descriptor for face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (10) (2015) 2041–2056. http://dx.doi.org/10.1109/TPAMI.2015.2408359.