Semi-Supervised Learning Using Random Walk Limiting Probabilities

Thiago Henrique Cupertino and Liang Zhao

Institute of Mathematics and Computer Science
University of São Paulo, São Carlos, Brazil
{thiagohc,zhao}@icmc.usp.br

Abstract. The semi-supervised learning paradigm allows that a large amount of unlabeled data be classified using just a few labeled data. To account for the minimal *a priori* label knowledge, the information provided by the unlabeled data is also used in the classification process. This paper describes a semi-supervised technique that uses random walk limiting probabilities to propagate label information. Each label is propagated through a network of unlabeled instances via a biased random walk. The probability of a vertex receiving a label is expressed in terms of the limiting conditions of the walk process. Simulations show that the proposed technique is competitive with benchmarked techniques.

Keywords: network-based learning, semi-supervised learning, random walk, limiting probabilities, stationary distribution.

1 Introduction

Semi-supervised learning (SSL) is a machine learning paradigm that overcomes the problem originated when labeling a training data set becomes expensive and very time-consuming. The main idea behind this paradigm is to classify data using just a few labeled instances and the information provided by many unlabeled instances [1]. This is practicable due to three SSL assumptions: manifold, smoothness and cluster. The manifold assumption states that the high-dimensional data lies on a low-dimensional manifold whose properties ensure more accurate density estimation and more appropriate similarity measures. The smoothness assumption states that if two points are close to each other in a high density region, then their correspondent labels should be close to each other as well. Finally, the cluster assumption states that if two points are in the same cluster, then they are likely to be of the same class (or, in other words, to have the same label). In this way, the SSL approach can provide high classification accuracies using less human effort and exploiting the unlabeled massive group of data.

Random walk theory has been applied in many machine learning problems. In image analysis, for example, a random walk process can be executed through pixels represented by network vertices. Texture discrimination and edge detection were performed by comparing *boundary distributions* of such process [2, 3].

As an alternative way to perform edge detection and image segmentation, other measurements were derived. In [4], the *first time passage* probability is computed when a random walker passes through a labeled vertex (pixel) after starting from an unlabeled vertex. A similar approach applied to content-based image retrieval can be found in [5]. An agglomerative network-based classification method was introduced in [6]. In this work, the hierarchy is based on *life-time* of restricted random walks, in which the steps of a random walker are limited by a distance function of antecedent steps. The time approach was also used in [7], in which classification is achieved by comparing *commute times* to labeled points of different classes. In [8], an unlabeled instance is classified as the class which maximizes a *posterior probability* by considering the time a walker starting in a vertex of the same class reaches the unlabeled instance.

Despite many concepts of random walk processes were already applied to classification and correlate tasks, *limiting probabilities* has never been directly applied, to our knowledge, as is done in this work. To account for the usage of the information provided by the unlabeled instances in the SSL task, the random walk process takes into account the whole network, which is composed by the unlabeled instances. The few labeled instances are inserted into the network by a specific weight composition responsible for creating a bias to the classification process. This bias is taken into account when the limiting probabilities are calculated. As simulation results showed, this is an effective measurement to capture intrinsic relations among labeled and unlabeled vertices in a network.

2 Background: Random Walks and Limiting Probabilities

Random walks can be understood in terms of Markov chains [9]. Consider a stochastic process Ω with a finite state space Γ. For each $n \in \mathbb{N} = \{0, 1, 2, \ldots\}$, $\Omega_n \in \Omega$ is an element from Γ. Then, the stochastic process $\Omega = \{\Omega_n\}$ is called a Markov chain if probability $P(\Omega_{n+1} = i|\Omega_0, \ldots, \Omega_n) = P(\Omega_{n+1} = i|\Omega_n)$, $i \in \Gamma$, that is, the process is independent of past states provided that the current state Ω_n is known. In this work, a time-homogeneous chain is considered: when $P(\Omega_{n+1} = j|\Omega_n = i) = p_{ij}$ is independent of n [10]. The probabilities p_{ij} can be arranged into a Markov matrix $\mathcal{P} = \{p_{ij}\}$.

The probability for a random walk starting at state i_0 to end at a state i_m is given by the probability of the chain $P(i_0, i_m) = P(i_0, i_1)P(i_1, i_2)\ldots P(i_{m-1}, i_m)$. If an infinite number of transitions ($m \to \infty$) is considered, then a limiting probability (stationary state) needs to be calculated. It can be shown that the limiting probability $P^{\infty}(i_m) = \lim_{n\to+\infty} P^n(i_0, i_m)$ exists given a recurrent, non-null and aperiodic state i_m. Therefore, considering $|\Gamma| = q$, one is able to calculate the Markov matrix. The limiting probability of the final state i_m is independently of the initial state i_0 [10].

3 The Proposed Semi-Supervised Technique

Given a dataset $\mathcal{X} = \{\mathbf{x}_i, i = 1, \ldots, r\}$, the objective is to classify the subset of unlabeled instances $\mathcal{X} \supset \mathcal{X}^{(u)} = \{\mathbf{x}_i^{(u)},\ i = 1, \ldots, n\}$ using the subset composed of just a few labeled instances $\mathcal{X} \supset \mathcal{X}^{(l)} = \{\mathbf{x}_i^{(l)},\ i = 1, \ldots, m\}$, $l \in \{1, 2, \ldots, c\}$ ($\mathcal{X}^{(l)} \cap \mathcal{X}^{(u)} = \emptyset$, $\mathcal{X}^{(l)} \cup \mathcal{X}^{(u)} = \mathcal{X}$). To characterize a SSL task, $m \ll n$.

First, an undirected network $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ without self-loops is created. In this network, instances are represented by vertices, $\mathcal{V} = \mathcal{X}^{(u)}$, and similarities among instances are represented by edges, $\mathcal{E} = [\mathcal{W}_{ij}],\ i, j = 1, \ldots, n$. The network similarity matrix $\mathcal{W} = \{w_{ij}\}$ is calculated by using some sort of distance function as, for example, the Euclidean distance. w_{ij} is the similarity between a pair of instances $\mathbf{x}_i^{(u)}$ and $\mathbf{x}_j^{(u)}$. $w_{ij} = 0$ means that there is no link between $\mathbf{x}_i^{(u)}$ and $\mathbf{x}_j^{(u)}$.

In the next step, an labeled instance $\mathbf{x}_j^{(l)}$ is inserted into the network $\mathcal{N}$. To do so, the similarities $\mathcal{S}_j = [s_{j1}, s_{j2}, \ldots, s_{jn}]^T$, between $\mathbf{x}_j^{(l)}$ and all other vertices $\mathbf{x}_i^{(u)} \in \mathcal{V}$ are calculated by using a distance function, and a new asymmetric $n \times n$ modified similarity matrix $\hat{\mathcal{W}}_j$ is constructed by composing it with the matrix of weight biases $\hat{\mathcal{S}}_j$:

$$\hat{\mathcal{W}}_j = \mathcal{W} + \epsilon \hat{\mathcal{S}}_j, \tag{1}$$

where ϵ is a non-negative parameter and $\hat{\mathcal{S}}_j$ is the following $n \times n$ matrix composition:

$$\hat{\mathcal{S}}_j = \begin{bmatrix} \mathcal{S}_j^T \\ \mathcal{S}_j^T \\ \vdots \\ \mathcal{S}_j^T \end{bmatrix}, \tag{2}$$

where $\mathcal{S}_j^T$ is the transpose of vector $\mathcal{S}_j$.

In Eq. 1, it can be observed that the weight biases of $\mathbf{x}_j^{(l)}$, encoded in matrix $\hat{\mathcal{S}}$, are applied over all edges w_{ij} of network $\mathcal{N}$, that is, the weight of each edge is linearly added up with the corresponding weight bias. The idea behind this operation is that the distance between any pair of vertices is reduced because of the new route introduced by the insertion of the labeled data instance. The higher the proximity between the labeled instance and a vertex, say vertex i, the more strengthened the connections from all other vertices to vertex i are. The parameter ϵ controls the influence of weight bias provided by matrix $\hat{\mathcal{S}}$ on the original network. The larger is the value of parameter ϵ, the greater will be the influence of the bias weights provided by $\mathbf{x}_j^{(l)}$.

After the bias composition, $\mathbf{x}_j^{(l)}$ is effectively inserted into network $\mathcal{N}$. To do so, an $(n+1) \times (n+1)$ adjacency matrix $\mathcal{A}_j = \{a_{ij}\}$ is constructed:

$$\mathcal{A}_j = \begin{bmatrix} \hat{\mathcal{W}} & \mathcal{S}_j \\ \mathcal{S}_j^T & 0 \end{bmatrix}. \tag{3}$$

In this formulation, without loss of generality, $\mathbf{x}_j^{(l)}$ is inserted as the last entry $(n+1)$ of matrix $\mathcal{A}_j$.

The two steps described above can be easily understood by using the toy example depicted in Fig. 1. In this example, a network is formed by 4 unlabeled

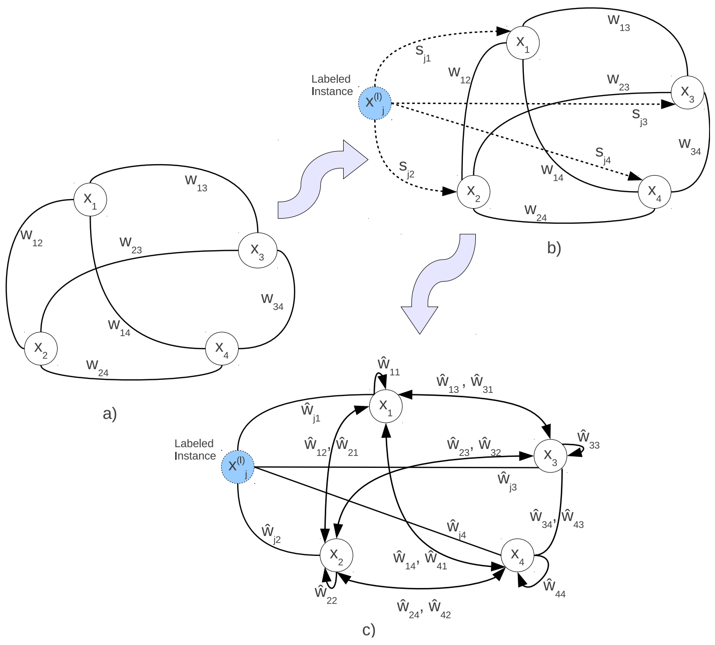


Fig. 1. Composition process for the modified similarity matrix $\hat{\mathcal{W}}_j$. a) An undirected and complete network $\mathcal{N}$ is formed by using 4 unlabeled instances; b) similarities $\mathcal{S}_j$ are calculated for the labeled instance $\mathbf{x}_j^{(l)}$; c) modified network $\mathcal{A}_j$, directed with self-loops, after bias composition (Eq. 1) and insertion of the labeled virtual state $\mathbf{x}_j^{(l)}$ (Eq. 3).

instances (Fig. 1a). In this initial network, the links are undirected and the similarity matrix is symmetric. Next, the similarity vector $\mathcal{S}_j$ between $\mathbf{x}_j^{(l)}$ and all other vertices is computed (Fig. 1b). After this computation, the weight biases of $\mathbf{x}_j^{(l)}$ are added up to the original similarity matrix to form a biased similarity matrix for the same network (Eq. 1). After that, $\mathbf{x}_j^{(l)}$ is effectively inserted into the network (Eq. 3). It can be seen from Fig. 1c that the network becomes directed and with self-loops (except for $\mathbf{x}_j^{(l)}$, that has no self-loops) and the biased weight matrix is no more symmetric.

After the insertion of $\mathbf{x}_j^{(l)}$, we are able to compute the entries of the transition Markov matrix $\mathcal{P}_j = [p_{ik}]$ by scaling the entries of matrix $\mathcal{A}_j$:

$$p_{ik} = a_{ik} / \sum_{k=1}^{n+1} a_{ik}. \tag{4}$$

With the above matrix $\mathcal{P}_j$ at hand, the limiting probabilities are calculated. This calculation can be performed by two ways: finding the eigenvector corresponding to the unit eigenvalue of matrix $\mathcal{P}_j$ or, in a faster way, iterating the system

$$\mathbf{p}_j(t+1) = \mathcal{P}_j \mathbf{p}_j(t), \tag{5}$$

to the stationary state, where $\mathbf{p}_j$ is an $(n+1) \times (n+1)$ normalized vector. It results in the following vector:

$$\mathbf{p}_j^{\infty} = [p_1 \, p_2 \ldots p_{n+1}] \tag{6}$$

where each column represents an unlabeled vertex, and each entry p_i represents the probability that $\mathbf{x}_i^{(u)}$ belongs to the class l of the labeled vertex $\mathbf{x}_j^{(l)}$ (the probability p_{n+1} is ignored as it represents $\mathbf{x}_j^{(l)}$, which is inserted into $\mathcal{N}$ by means of Eq. 3).

Finally, the classification is completed by repeating the above steps (Eq. 1 through 6) for all labeled instances $\mathbf{x}_j^{(l)} \in \mathcal{X}^{(l)}$. The classification of $\mathbf{x}_i^{(u)}$ is accomplished by assigning it the most representative label, that is, after averaging all $\mathbf{p}_j^{\infty}$ for each label (each $\mathbf{x}_j^{(l)}$), the following matrix is constructed:

$$\mathbf{p}^{\infty} = \begin{bmatrix} p_{11} & p_{21} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1c} & p_{2c} & \cdots & p_{nc} \end{bmatrix}, \tag{7}$$

where p_{il}, $l = 1, 2, \ldots, c$, is the averaged probability that instance $\mathbf{x}_i^{(u)}$ belongs to class l. Then, the label $p_{max} = argmax\{p_{il}\}$, corresponding to the largest probability value, is assigned to $\mathbf{x}_i^{(u)}$.

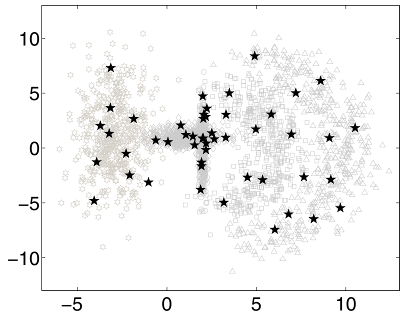
In a concise form, the proposed semi-supervised transductive classification can be summarized by Algorithm 1.

4 Illustrative Toy Example

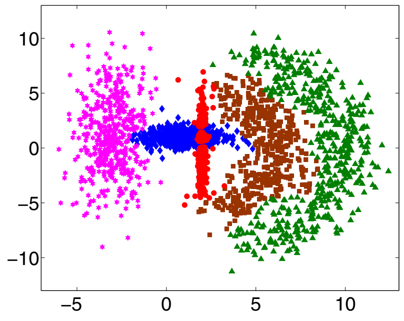
In this subsection, simulation results on a toy example are presented. It was used a toy data set that captures different class characteristics, such as different shapes and densities, to illustrate the behavior of the semi-supervised technique. The toy example in Fig. 2 encompasses a challenging classification task. This data set is composed of 3 different class distributions (from left to right): Gaussian, Highleyman and Lithuanian. The data was generated by using the PRTools toolbox [11]. Each class has 500 instances, totaling 2500 instances for the entire data set. In addition, each class comprises 10 labeled instances, representing (2%) of the entire data. Figure 2b shows that the proposed technique satisfactorily detected the 5 classes.

Algorithm 1. The proposed semi-supervised technique

```
Input:
$c$ : number of classes
$\mathcal{X}^{(u)}$ : unlabeled dataset
$\mathcal{X}^{(l)}$ : labeled dataset
Parameters:
$\epsilon$ : bias weighting
Output:
Estimated class ($l \in \{1, \ldots, c\}$) for $\mathbf{x}_i^{(u)} \in \mathcal{X}^{(u)}$
Training:
1. $\mathcal{N}$ = Create a network from $\mathcal{X}^{(u)}$
Classification:
for each $\mathbf{x}_j^{(l)} \in \mathcal{X}^{(l)}$ do
    2. $\hat{\mathcal{W}}_j$ = Compose bias weights into $\mathcal{N}$ (Eq. 1)
    3. $\mathcal{A}_j$ = Insert the virtual state $\mathbf{x}_j^{(l)}$ into $\mathcal{N}$ (Eq. 3)
    4. $\mathbf{p}_j^{\infty}$ = Compute limiting probabilities (Eq. 5)
end for
5. $\mathbf{p}^{\infty}$ = Averaged $\mathbf{p}_j^{\infty}$
6. Assign $\mathbf{x}_i^{(u)}$ the most representative class in $\mathbf{p}^{\infty}$
```



(a) Black-filled stars represent labeled instances.



(b) Classification achieved by the proposed technique.

Fig. 2. Mix of different cluster shapes for semi-supervised classification. This artificial data set is composed of 2500 instances divided into 5 balanced and distinct clusters shapes: Gaussian, Highleyman and Lithuanian. Each cluster contains 10 labeled instances.

5 Benchmark Data Sets

The proposed semi-supervised technique was tested and compared using 7 benchmark data sets. Table 1 shows a brief description of them. Three artificial sets (*g241c*, *g241d* and *Digit1*) were created in order to encompass some of the semi-supervised assumptions: manifold, smoothness and cluster. The other four data

Table 1. Meta-data of the datasets composing the SSL benchmark

Data Set	Classes	Dimension	Points	Type
g241c	2	241	1500	artificial
g241d	2	241	1500	artificial
Digit1	2	241	1500	artificial
USPS	2	241	1500	unbalanced
COIL	6	241	1500	
BCI	2	117	400	
Text	2	11960	1500	sparse discrete

Table 2. References for semi-supervised learning techniques used for comparisons

Abbreviation	Technique	Ref.
MVU + 1-NN	Maximum Variance Unfolding	[14]
LEM + 1-NN	Laplacian Eigenmaps	[15]
QC + CMR	Quadratic Criterion and Class Mass Reg.	[16]
Discrete Reg.	Discrete Regularization	[17]
TSVM	Transductive Support Vector Machines	[18]
SGT	Spectral Graph Transducer	[19]
Cluster-Kernel	Cluster Kernels	[20]
Data-Dep. Reg.	Data-Dependent Regularization	[21]
LDS	Low-Density Separation	[18]
Laplacian RLS	Laplacian Regularized Least Squares	[22]
CHM (normed)	Conditional Harmonic Mixing	[23]
LGC	Local and Global Consistency	[12]
LP	Label Propagation	[24]
LNP	Linear Neighborhood Propagation	[13]

sets (*USPS*, *COIL*, *BCI* and *Text*) were derived from real data. The benchmarks were developed to evaluate the power of different algorithms as neutral as possible [1]. For each data set, 24 independent splits of labeled data for the training set are available. 12 splits contain 10 labeled instances for each data set and the other 12 splits contain 100 labeled instances. For each split, at least 1 instance of each class is labeled. A more detailed explanation of each data set can be found in [1]. The proposed technique was compared to 16 well-known and established SSL techniques. Table 2 shows a brief description and the related references for them. All simulation results were extracted from the reference [1], where it can be found values for parameter optimization and model selection in order to minimize test errors. For LGC, LP and LNP, σ was selected from the set $\{0, 1, \ldots, 100\}$ and α was fixed to $\alpha = 0.99$ (the same setup done in [12] and [13]). For the LNP, k was evaluated for the values in $\{1, 2, \ldots, 100\}$. The configuration and parameter optimization for the proposed technique was done as follows. For the network construction (Step 1 of Alg. 1), the k-nearest neighbor technique was used: each vertex was linked with its k most similar neighbors. Parameter k was evaluated for the values in $\{1, 2, \ldots, 100\}$ and parameter ϵ was evaluated for the values in $\{0, 0.1, 0.2, \ldots, 10\}$. In all simulations, no data preprocessing was performed by the techniques and the Euclidean distance was used as a distance function.

Tables 3 and 4 shows the simulations results for 10 and 100 labeled instances, respectively. For 10 labeled instances, the proposed technique achieved an average rank of 5.86 (2*nd* place) and, for 100 labeled instances, an average rank of 9.29 (11*th* place). Overall, it achieved an average rank of 7.57 (5*th* place) - preceded by LP (7.21), LDS (6.93), Laplacian RLS (5.43) and SGT (5.33). Interestingly, it achieved a very high position (2*nd*) in the case of only 10 labeled instances, a very challenging semi-supervised task in which as only as a small portion of 0.67% of the dataset is labeled. Hence, concerning the 17 techniques and the 7 benchmark datasets, we conclude that the proposed technique is at least comparable to the best known semi-supervised techniques.

Table 3. Classification error rate (%) and the corresponding average rank of each technique. Best results are in bold face. Data sets with 10 labeled points.

	g241c	*g241d*	*Digit1*	*USPS*	*COIL*	*BCI*	*Text*	**Avg. Rank**
1-NN	47.88	46.72	13.65	16.66	63.36	49.00	38.12	9.57
SVM	47.32	46.66	30.60	20.03	68.36	49.85	45.37	14.00
MVU + 1-NN	47.15	45.56	14.42	23.34	62.62	47.95	45.32	9.86
LEM + 1-NN	44.05	43.22	23.47	19.82	65.91	48.74	39.44	10.00
QC + CMR	39.96	46.55	9.80	13.61	59.63	50.36	40.79	7.71
Discrete Reg.	49.59	49.05	12.64	16.07	63.38	49.51	40.37	10.57
TSVM	24.71	50.08	17.77	25.20	67.50	49.15	31.21	10.71
SGT	**22.76**	**18.64**	8.92	25.36	N/A	49.59	29.02	6.17
Cluster-Kernel	48.28	42.05	18.73	19.41	67.32	48.31	42.72	10.86
Data-Dep. Reg.	41.25	45.89	12.49	17.96	63.65	50.21	N/A	9.83
LDS	28.85	50.63	15.63	17.57	61.90	49.27	**27.15**	8.29
Laplacian RLS	43.95	45.68	**5.44**	18.99	**54.54**	48.97	33.68	6.00
CHM (normed)	39.03	43.01	14.86	20.53	N/A	46.90	N/A	7.20
LGC	45.82	44.09	9.89	**9.03**	63.45	47.09	45.50	7.29
LP	42.61	41.93	11.31	14.83	55.82	**46.37**	49.53	**5.57**
LNP	47.82	46.24	8.58	17.87	55.50	47.65	41.06	7.14
Proposed Method	40.30	41.74	13.94	19.98	59.40	46.69	34.32	5.86

Table 4. Classification error (%) and the corresponding average rank of each technique. Best results are in bold face. Data sets with 100 labeled points.

	g241c	*g241d*	*Digit1*	*USPS*	*COIL*	*BCI*	*Text*	**Avg. Rank**
1-NN	43.93	42.45	3.89	5.81	17.35	48.67	30.11	12.57
SVM	23.11	24.64	5.53	9.75	22.93	34.31	26.45	9.29
MVU + 1-NN	43.01	38.20	2.83	6.50	28.71	47.89	32.83	11.71
LEM + 1-NN	40.28	37.49	6.12	7.64	23.27	44.83	30.77	12.00
QC + CMR	22.05	28.20	3.15	6.36	10.03	46.22	25.71	7.43
Discrete Reg.	43.65	41.65	2.77	4.68	**9.61**	47.67	24.00	8.14
TSVM	18.46	22.42	6.15	9.77	25.80	33.25	24.52	8.71
SGT	17.41	9.11	2.61	6.80	N/A	45.03	**23.09**	**4.50**
Cluster-Kernel	**13.49**	**4.95**	3.79	9.68	21.99	35.17	24.38	6.71
Data-Dep. Reg.	20.31	32.82	**2.44**	5.10	11.46	47.47	N/A	6.83
LDS	18.04	23.74	3.46	4.96	13.72	43.97	23.15	5.43
Laplacian RLS	24.36	26.46	2.92	4.68	11.92	**31.36**	23.57	4.86
CHM (normed)	24.82	25.67	3.79	7.65	N/A	36.03	N/A	8.80
LGC	41.64	40.08	2.72	**3.68**	45.55	43.50	46.83	9.86
LP	30.39	29.22	3.05	6.98	11.14	42.69	40.79	8.86
LNP	44.13	38.30	3.27	17.22	11.01	46.22	38.48	12.14
Proposed Method	27.92	26.19	3.57	9.59	20.01	44.11	25.54	9.43

6 Conclusions

This paper has presented a new network-based semi-supervised classification technique. The set of unlabeled instances compose a network in which vertices represent the state space for a random walker. Via a specific matrix composition, each labeled instance is inserted into the network to provide a bias to the classification process. This label bias is propagated through the unlabeled vertices of the network by means of the limiting probabilities. The local and global topology are taken into account by the random walk process and thus both clustering and smoothness SSL assumptions are satisfied, making the usage of the unlabeled instances information effective. Simulations have showed that the proposed technique is capable of detecting classes that present different shapes and distribution. The technique has also been demonstrated to be competitive with some well-known semi-supervised techniques using benchmark data sets.

Acknowledgments. The authors would like to acknowledge the São Paulo State Research Foundation (FAPESP) and the Brazilian National Council for Scientific and Technological Development (CNPq) for the financial support given to this research.

References

[1] Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge (2006)
[2] Wechsler, H., Kidode, M.: A random walk procedure for texture discrimination. IEEE Transactions on Pattern Analysis and Machine Intelligence 1(3), 272–280 (1979)
[3] Wechsler, H., Citron, T.: Feature extraction for texture classification. Pattern Recognition 12(5), 301–311 (1980)
[4] Grady, L.: Random walks for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(11), 1768–1783 (2006)
[5] Bul, S.R., Rabbi, M., Pelillo, M.: Content-based image retrieval with relevance feedback using random walks. Pattern Recognition 44(9), 2109–2122 (2011)
[6] Schll, J., Schll-Paschinger, E.: Classification by restricted random walks. Pattern Recognition 36(6), 1279–1290 (2003)
[7] Zhou, D., Schölkopf, B.: Learning from labeled and unlabeled data using random walks. In: Rasmussen, C.E., Bülthoff, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 237–244. Springer, Heidelberg (2004)
[8] Szummer, M., Jaakkola, T.: Partially labeled classification with markov random walks. In: Advances in Neural Information Processing Systems, pp. 945–952. MIT Press (2001)
[9] Gallager, R.G.: Discrete Stochastic Processes, 1st edn. Springer (1996)
[10] Çinlar, E.: Introduction to stochastic processes. Prentice Hall (1975)
[11] Duin, R., Juszczak, P., Paclik, P., Pekalska, E., de Ridder, D., Tax, D., Verzakov, S.: Prtools4.1, a matlab toolbox for pattern recognition (2007)
[12] Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schlkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems 16, vol. 16, pp. 321–328. MIT Press (2004)

[13] Wang, F., Zhang, C.: Label propagation through linear neighborhoods. IEEE Transactions on Knowledge and Data Engineering 20(1), 55–67 (2008)

[14] Sun, J., Boyd, S., Xiao, L., Diaconis, P.: The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. SIAM Review 48(4), 681–699 (2006)

[15] Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15(6), 1373–1396 (2003)

[16] Delalleau, O., Bengio, Y., Roux, N.L.: Nonparametric function induction in semi-supervised learning. In: Workshop Artif. Intell. Stat. (2005)

[17] Zhou, D., Schölkopf, B.: Adaptive computation and machine learning. In: Discrete Regularization, pp. 237–250. MIT Press, Cambridge (2006)

[18] Chapelle, O., Zien, A.: Semi–supervised classification by low density separation. In: Proceedings of the International Workshop on Artificial Intelligence and Statistics, pp. 57–64 (2005)

[19] Joachims, T.: Transductive learning via spectral graph partitioning. In: Proc. Int. Conf. Mach. Learn., pp. 290–297 (2003)

[20] Chapelle, O., Weston, J., Schölkopf, B.: Cluster Kernels for Semi-Supervised Learning. In: NIPS 2002, vol. 15, pp. 585–592. MIT Press, Cambridge (2003)

[21] Corduneanu, A., Jaakkola, T.: Adaptive computation and machine learning. In: Data-dependent Regularization, pp. 163–190. MIT Press, Cambridge (2006)

[22] Sindhwani, V., Niyogi, P.: Beyond the point cloud: from transductive to semi-supervised learning. In: Proc. 22nd Int. Conf. Mach. Learn., pp. 824–831 (2005)

[23] Burges, C.J.C., Platt, J.C.: Adaptive computation and machine learning. In: Semi-supervised Learning with Conditional Harmonic Mixing, pp. 251–273. MIT Press, Cambridge (2006)

[24] Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep (2002)