Directed Graph Embedding

Mo Chen
Tsinghua University
Beijing, 100084, P.R.China
mochen80@gmail.com

Qiong Yang, Xiaoou Tang
Microsoft Research Asia
Beijing, 100080, P.R.China
{qyang, xitang}@microsoft.com

Abstract

In this paper, we propose the Directed Graph Embedding (DGE) method that embeds vertices on a directed graph into a vector space by considering the link structure of graphs. The basic idea is to preserve the locality property of vertices on a directed graph in the embedded space. We use the transition probability together with the stationary distribution of Markov random walks to measure such locality property. It turns out that by exploring the directed links of the graph using random walks, we can get an optimal embedding on the vector space that preserves the local affinity which is inherent in the directed graph. Experiments on both synthetic data and real-world Web page data are considered. The application of our method to Web page classification problems gets a significant improvement comparing with state-of-art methods.

1 Introduction

We consider the problem that embeds nodes on directed graph into a Euclidean vector space while preserving locality which is inherent in the graph structure. There is a large amount of problems which can be naturally represented as a directed graph. Typical examples are web information retrieval based on hyperlink structure, document classification based on citation graphs [C. Lee Giles *et al*., 1998] and protein clustering based on the pairwise alignment scores [W. Pentney and M. Meila, 2005]. Some works have been done to deal with the ranking problem on link structure of the Web including the PageRank [S. Brin and L. Page, 1998] and HITS [J. Dean and M. Henzinger, 1999] algorithms, yet it is still a hard task to do general data analysis on directed graphs such as classification and clustering. In [D. Zhou *et al*., 2005] the authors proposed a semi-supervised learning algorithm for classification on directed graph, and also an algorithm to partition the directed graph. In [W. Pentney and M. Meila, 2005] the authors proposed algorithms to do clustering on protein data which was formulated into a directed graph based on asymmetric pairwise alignment scores. However, up to now, works are quite limited due to the difficulty in exploring the complex structure of directed graphs. On the other hand, there are a lot of data mining and machine learning techniques, such as Support Vector Machine (SVM), operating data on a vector space or an inner product space. Embedding the data of directed graphs to vector spaces becomes quite appealing for tasks of data analysis of directed graphs. The motivations are:

1) Instead of designing new algorithms for each task in data mining on directed graphs that are directly applied to link structure data, we can first provides a unified framework to embed the link structure data into the vector space, and then utilize the mature algorithms that already exist for mining on the vector space.
2) Directly analyzing data on directed graphs is quite hard, since some concepts such as distance, inner product, and margin, which are important for data analysis, are hard to define in directed graph. But for vector data, these concepts are already well defined. Tools for analyzing data can be easily obtained.
3) Given a huge directed graph with complex link structure, it is highly difficult to perceive the latent relations of the data. Such information may be inherent in the topological structure and link weights. Embedding these data into vector spaces will help people to analyze these latent relations visually.

Some works have been done for embedding on the undirected graph. Manifold learning techniques [M. Belkin and P. Niyogi , 2002] [S.T.Roweis and L.K.Saul, 2000] first connect data into an undirected graph in order to approximate the manifold structure where the data is assumed to be lying on. Then they embed the vertices of the graph into a low dimensional space. Edges of the graph reflect the local affinity of node pairs in the input space. In the next, an optimal embedding is achieved by preserving such a local affinity. However, in the directed case, the edge weight between two graph nodes is not necessarily symmetric. It can not be directly used as a measure of affinity. Motivated by [D. Zhou *et al*., 2005], we formulate the directed graph in a probabilistic framework. We use random walks to measure the local affinity of vertices on the directed graph. Based on that, we propose an algorithm embedding the nodes on the directed graph into a vector space by using random walk metric.

The rest of the paper is organized as follows: In section 2, we give denotations used in our paper. In section 3, we introduce the details of our method. Implementation issues are addressed in section 4. The relation between our method and

previous works is considered in section 5. Experimental results are shown in section 6 and the last part is the conclusion.

2 Preliminary

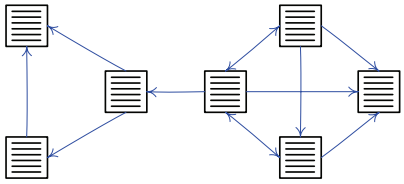


Figure 1. The World Wide Web can be modeled as a directed graph. Web pages and hyperlink can be represented as vertices and directed edge of the graph.

An example of directed graph is the World Wide Web as shown in Fig.1. A directed graph $G=(V,E)$ consists of a finite vertex set V which contains n vertices, together with an edge set $E \subseteq V \times V$. An edge of a directed graph is an ordered pair vertex (u,v) from u to v. Each edge may associate a positive weight w. An un-weighted directed graph can be simply viewed as all the weight of the edge is one. The out-degree $d_O(v)$ of a vertex v is defined as $d_O(v)=\sum_{u,v\to u} w(v,u)$, where the in-degree $d_I(v)$ of a vertex v is defined as $d_I(v)=\sum_{u,u\to v} w(u,v)$, $u \to v$ means u has a directed link pointing to v. On the directed graph, we can define a transition probability matrix $P=[p(u,v)]_{u,v}$ of a Markov random walk through the graph. It satisfies $\sum_v p(u,v)=1, \forall u$. We also assume the stationary distribution for each vertex v is π_v ($\sum_v \pi_v = 1$), which can be guaranteed if the chain is irreducible. For a connected directed graph, a natural definition of the transition probability matrix can be $p(u,v)=w(u,v)/d_O(u)$ in which a random walker on a node jumps to its neighbors with a probability proportion to the edge weight. For a general directed graph, we can define a slightly different transition matrix. We will be back to this issue in the implementation section.

3 Algorithm

We aim to embed vertices on directed graph into a vector space preserving the locality property of vertex u to all its neighbors. First, let's consider the problem mapping the connected directed graph to a line. We define a general optimization target as:

$$\sum_u T_V(u) \sum_{v,u\to v} T_E(u,v)(y_u - y_v)^2$$

y_u is the coordinate of vertex u in embedded one dimension space. The term T_E is used to measure the importance of a directed edge between two vertices. If $T_E(u,v)$ is large, then the two vertices u and v should be close to each other on the embedded line. The term T_V is used to measure the importance of a vertex on the graph. If $T_V(u)$ is large, then the relation between vertex u and its neighbors should be emphasized. By minimizing such a target, we are able to get an optimized embedding for the graph on one dimensional space. The embedding considers both the local relation of node pairs and global relative importance of nodes.

In the following, we address the embedding problem of directed graphs under two assumptions:

1) Two vertices are relevant if there is edge between them. The relevance strength is related to the edge weight.
2) The out-link of the vertex which has many out-links carries relatively low information about the relevance between vertices.

The assumptions are reasonable in many tasks. Let's again take the Web by example. Web page authors usually insert links to pages which are relevant to their own pages. Therefore, if a Web page A has a hyper-link pointing to Web page B, we assume A and B might be relevant in some sense. We should preserve such a relation in the embedded space.

Let's consider a web page which has many out links, such as the home page of www.yahoo.com. The page linked by the home page of yahoo may share less similarity with it, and then in the embedded feature space the two web pages should have a relatively large distance. We can use the transition probability of random walks to measure the locality property. When a web page has many out-links, each out-link will have a relatively low transition probability. Such a measure meets the assumption 1 and 2.

Different web pages have different importance in the Web environment. Ranking web pages according to their importance is a well studied area. The stationary distribution of random walks on the link-structure environment is well known as a good measure of such importance which is used in many ranking algorithms including PageRank. In order to emphasize those important pages in the embedding feature space, we use the stationary distribution π_u of random walks to weigh the page u in the optimization target.

Taking all above into account, we rewrite the optimization target as follows:

$$\sum_u \pi_u \sum_{v,u\to v} p(u,v)(y_u - y_v)^2$$

We can further rewrite the formula as follows:

$$\sum_u \pi_u \sum_{v,u\to v} p(u,v)(y_u - y_v)^2 = \sum_{u,v} (y_u - y_v)^2 \pi_u p(u,v)$$

$$= \frac{1}{2}\left(\sum_{u,v} (y_u - y_v)^2 \pi_u p(u,v) + \sum_{v,u} (y_v - y_u)^2 \pi_v p(v,u) \right)$$

$$= \frac{1}{2} \sum_{u,v} (y_u - y_v)^2 \left(\pi_u p(u,v) + \pi_v p(v,u) \right)$$

Thus the problem is equivalent to embedding the vertices into a line while preserving the local symmetric measure $(\pi_u p(u,v) + \pi_v p(v,u))/2$ of each vertices pair, here $\pi_u p(u,v)$ is the probability of a random walker jumps to

vertex u then to v, i.e. the probability of the random walker passes the edge (u,v). This also can be deemed as a percentage of flux in the total flux at stationary state when we continuously import water into the graph. This directed force manifests the impact of u on v, or the volume of message that u conveys to v.

By optimizing the target we consider not only the local property reflected by edges between node pairs, but also the global reinforcement to the relation by taking the stationary distribution of random walks into account.

We denote

$$L = \Phi - \frac{\Phi P + P^T \Phi}{2}$$

where P is the transition matrix, i.e. $P_{ij} = p(i,j)$, Φ is the diagonal matrix of the stationary distribution, i.e. $\Phi = diag(\pi_1,...,\pi_n)$. Clearly, from the definition L is symmetric. Then we have the following proposition.

Proposition 1

$$\sum_u \pi_u \sum_{v,u\to v} p(u,v)(y_u - y_v)^2 = 2y^T Ly$$

where $y = (y_1,...,y_n)^T$.

The proof of this proposition is given in the appendix. The above L is known as combinatorial Laplacian on a directed graph [F. R. K. Chung, 2005]. From the proposition we can see that L is a semi-positive definite matrix.

Therefore, the minimization problem reduces to find

$$\underset{y}{\operatorname{argmin}}\ y^T Ly$$

$$s.t. \quad y^T \Phi y = 1$$

The constraint $y^T \Phi y = 1$ removes an arbitrary scaling factor of the embedding. Matrix Φ provides a natural measure of the vertex on the graph. The problem is solved by the general eigendecomposition problem:

$$Ly = \lambda \Phi y$$

Alternatively, we could use $y^T y = 1$ as the constraint. Then the solution is achieved by solving $Ly = \lambda y$.

Let e be a vector with all entry 1. It can be easily shown that e is an eigenvector with eigenvalue 0 for L. If the transition matrix is primitive, e is the only eigenvector for $\lambda = 0$. The meaning of the first eigenvector is to map all data to a single point, which minimizes the optimization target. To eliminate this trivial solution we put an addition constraint of orthogonality:

$$\underset{y}{\operatorname{argmin}}\ y^T Ly$$

$$s.t. \quad y^T \Phi y = 1$$

$$y^T \Phi e = 0$$

Thus the solution is given by the eigenvector of the smallest non-zero eigenvalue. Generally, embedding the graph into R^k (k>1) is given by the $n \times k$ matrix $Y = [y_1 ... y_k]$ where the ith row provides the embedding of the ith vertex. Therefore we minimize

$$\sum_u \pi_u \sum_{v,u\to v} p(u,v) \left\| Y_u - Y_v \right\|^2 = 2tr(Y^T LY)$$

It can be rewrite as

$$\min\ tr(Y^T LY)$$

$$s.t.\ \ Y^T \Phi Y = I$$

The solution is given by $Y^* = [\nu_2^*,...,\nu_{k+1}^*]$ where ν_i^* is the eigenvector of ith smallest eigenvalue of the generalized eigenvalue problem $Ly = \lambda \Phi y$.

4 Implementation Issue

Input: adjacency matrix W, dimension of target space k and a perturbation factor α

1. Compute $P = \alpha(D_O^{-1} W + \frac{1}{n}\mu e^T) + (1-\alpha)\frac{1}{n} ee^T$, where μ is a vector that $\mu_i = 1$ if row i of W is 0, and D_O is the diagonal matrix of the out degrees.
2. Solve the eigenvalue problem $\pi^T P = \pi^T$ subject to a normalized equation $\pi^T e = 1$.
3. Construct the combinatorial Laplacian of the directed graph $L = \Phi - \frac{\Phi P + P^T \Phi}{2}$, where $\Phi = diag(\pi_1,...,\pi_n)$
4. Solve the generalized eigenvector problem $Ly = \lambda \Phi y$, let $\nu_1^*,...,\nu_n^*$ be the eigenvectors ordered according to their eigenvalues with ν_1^* having the smallest eigenvalue λ_1 (in fact zero). The image of X_i embedded into k dimensional space is given by $Y^* = [\nu_2^*,...,\nu_{k+1}^*]$.

Table 1. The DGE algorithm

The irreducibility of the Markov chain guarantees that the stationary distribution vector π exists. We here will build a Markov chain with a primitive transition probability matrix P. In general, for a directed graph, the matrix of transition probability P defined by $p(u,v) = w(u,v)/d_O(u)$ is not irreducible. We will use the so called teleport random walk [A. Langville and C. Meyer, 2004] on a general directed graph. The transition probability matrix is given by

$$P = \alpha(D_O^{-1} W + \frac{1}{n}\mu e^T) + (1-\alpha)\frac{1}{n} ee^T$$

where W is the adjacent matrix of the directed graph, μ is a vector that $\mu_i = 1$ if row i of W is 0, and D_O is the diagonal matrix of the out degree. Then P will be stochastic, irreducible and primitive. This can be interpreted as a probability α of transiting to an adjacent vertex and a probability $1-\alpha$ of jumping to any point on the graph uniform ran-

domly. For those vertices that don't have any out link, just uniform randomly jump to any point on the graph. Such a setting can be viewed as adding a perturbation to the original graph. The smaller the perturbation is the more accurate result we can get. So in practice we only need to set α to a very small value. In this paper, we simply set α to be 0.01.

The stationary distribution vector then can be obtained by solving an eigenvalue problem $\pi^T P = \pi^T$ subject to a normalized equation $\pi^T e = 1$.

The algorithm for embedding vertices on a directed graph into a vector space is summarized in Table1.

5 Relation with Previous Works

In [M. Belkin and P. Niyogi , 2002] the authors proposed the Laplacian Eigenmap algorithm for nonlinear dimensional reduction. We can see if our algorithm is applied to an undirected graph we will get a similar solution to Laplacian Eigenmap. In the case of undirected graph, we can define the transition probability as $p(u,v) = w(u,v)/d_u$, where $w(u,v)$ is the weight of the undirected edge (u,v), d_u is the degree of vertex u. If the graph is connected then the stationary distribution on vertex u can be proved equal to $d_u / Vol(G)$, where $Vol(G)$ is the volume of the graph, thus

$$\sum_u \pi_u \sum_{v,u\to v} p(u,v)(y_u - y_v)^2 = \sum_{u,v}(y_u - y_v)^2 \pi_u p(u,v)$$

$$= \sum_{u,v}(y_u - y_v)^2 w(u,v)/Vol(G)$$

$$y^T \Phi y = y^T D y / Vol(G)$$

where $D = diag(d_1,...,d_n)$. Then the problem reduces to the Laplacian Eigenmap.

In [D. Zhou *et al*., 2005] Zhou proposed a semi-supervised classification algorithm on a Directed Graph, by solving an optimization problem. The basic assumption is the smooth assumption that the class labels of the vertices on the directed graph should be similar if the vertices are closely related. The algorithm is to minimize a regularization risk between the least square error and a smooth term. Consider the problem that the data in the same class is scattered and the decision boundary is complicated, and the smooth assumption does not hold. Then the classification result may be hindered. Another problem is that by using least square error the data far away from the decision boundary also contribute a large penalty in the optimization target. Thus considering the imbalanced data, the side with more training data may have more total energy, and the decision boundary is biased. In the experiment section we will show a comparison between Zhou's algorithm and our method used together with an SVM classifier.

In [D. Zhou *et al*., 2005] the author also proposed the directed version of normalized cut algorithm. The solution is given by the eigenvector corresponding to the second largest eigenvalue of matrix $\Theta = (\Phi^{1/2} P \Phi^{-1/2} + \Phi^{-1/2} P^T \Phi^{1/2})/2$. It can be seen that the eigenvector corresponding to the second largest eigenvalue of Θ is in fact the eigenvector v corresponding to the second smallest eigenvalue of $\mathbb{L} = I - \Theta$. Note that we have such an equation

$$\frac{y^T L y}{y^T \Phi y} = \frac{v^T \Phi^{-1/2} L \Phi^{-1/2} v}{v^T v} = \frac{v^T \mathbb{L} v}{v^T v}, \quad y = \Phi^{-1/2} v$$

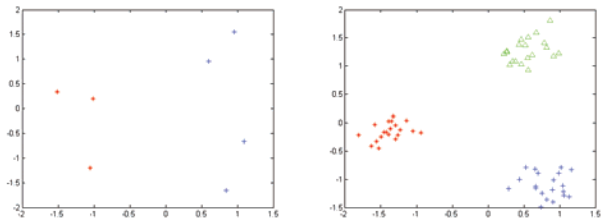
Therefore, the cutting result is equal to embedding data into a line by DGE, then using threshold 0 to cut the data.

6 Experiments

In this section, experiments are designed to show the embedding effect in both toy problems and real world data. Using DGE as a preprocess procedure, we also consider an application of the proposed directed graph embedding algorithm to a web page classification problem with comparisons to a state-of-art algorithm.

Toy Problems

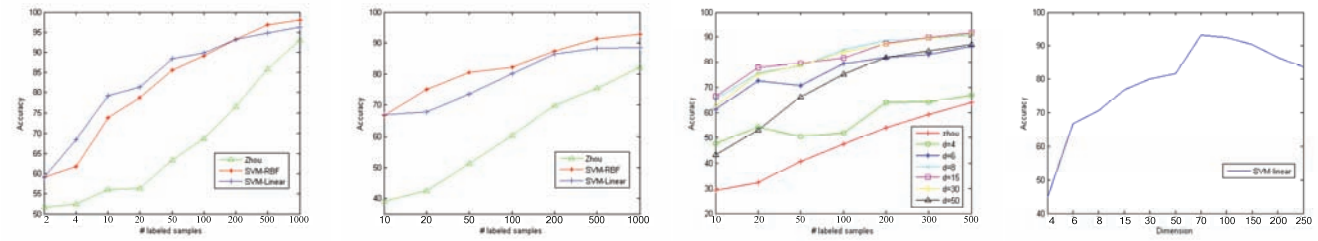
We first test our algorithm on the toy data shown in Fig.1. The edge weights are set as binary values. Fig.2(a) shows the result of embedding the directed graph into a plane. The red nodes and blue nodes in Fig.2(a) are corresponding to the three nodes on the left and four nodes on the right in Fig.1 respectively. From the figure, we can see the locality property of the graph is well preserved, and the embedding result reflects subgraph structure of the original graph.



(a) **(b)**

Figure 2. Embedding result of two toy problems on 2D space

In another experiment, a directed graph consisting of 60 vertices is generated. There are three subgraphs, and each consists of 20 vertices. Weights of the inner directed edges in the subgraph are drawn uniformly from interval [0.25, 1]. Weights of directed edges between the subgraphs are drawn uniformly from interval [0, 0.75]. By generating the edge weights in such a manner, each subgraph is relatively impact. The graph is a full-connected directed graph. If only given the graph without a prior knowledge of the data, we can hardly see the latent relation of the data. The embedding result by DGE in two dimensional space is shown in Fig.2(b). We can see that tightly related nodes on the directed graph are clustered in the 2D Euclidean space. After embedding the data into a vector space, we can easily perceive the clustered structure of the original graph, which gives us insight on the principal issues such as latent complexity of the directed graph.



(a) (b) (c) (d)

Figure 4. Classification Result: a) two-class problem, dimension=20; b) multi-class problem, dimension=20; c) multi-class in different dimension spaces by nonlinear SVM; d) accuracy against dimension on a fixed (500 labeled samples) training set by linear SVM

Web Page Data

To address directed graph embedding task on real world data, we test our method on the WebKB dataset. We consider a subset containing the pages from the three universities Cornell, Texas and Wisconsin. We remove the isolated pages, resulting in 847, 809 and 1227 pages respectively. We may assign a weight to each hyperlink according to the textual content or the anchor text. However here we are only interested in how much we can obtain from link structure only and hence adopt the binary weight function. Fig.3 shows the embedding result of the WebKB data in three dimensional space. The red, blue and black nodes are corresponding to the web pages of three universities: Cornell Texas and Wisconsin respectively. From the figure we can see that the embedding results of web pages in each university are relatively impact, while those of web pages in different universities are well separated. This shows our method is effective in analyzing the link structure between different universities, where the inner links in one university are denser than that between universities.

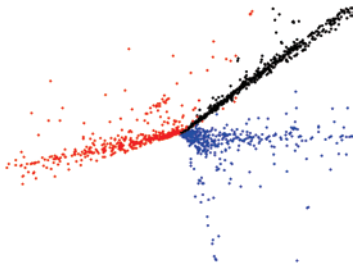


Figure 3. Embedding result of WebKB data

Application in Web Page Classification

Our method can be used in many applications, such as classification, clustering, information retrieval. As an example, we apply our algorithm to a web page classification task. Web pages of four universities Cornell, Texas, Washington and Wisconsin in WebKB dataset are used. Still the binary edge weight setting is adopted. We first use DGE to embed the vertices into certain Euclidean space, and then train an SVM classifier to do the classification task. After that, we compare the results with the state-of-art classification algorithm (referred as Zhou) proposed in [D. Zhou *et al*., 2005]. Here we use nu-SVM [B. Schölkopf and A. J. Smola, 2002], a modified version of SVM, which is easy for model selection. Both linear and nonlinear SVM are tested. In the nonlinear setting, RBF kernel is used. In all experiments, the training data are randomly sampled from the data set. To ensure that there is at least one training sample for each class, we conduct the sampling again when there is no labeled point for some class. The testing accuracies are averaged over 20 times' experimental results. Different dimensional embedding spaces are also considered to study the dimensionality of the embedded space.

The comparing results of binary classification problem are shown in Fig.4(a). We consider the web pages of two Universities randomly selected from the WebKB data set. We first use DGE to embed the whole dataset into a 20-dimensional space, then use SVM to do the classification task. The parameter nu is set to 0.1 for both linear SVM and nonlinear SVM. The parameter σ of RBF kernel is set to be 38 for nonlinear SVM. The parameter α for Zhou's algorithm is set to be 0.9 as proposed in his paper. From the figure, we can see that in all cases where the number of training samples varies from 2 to 1000, DGE used together with either liner or nonlinear SVM consistently achieves better performance than Zhou's algorithm. The reason might be that Zhou's method directly applies the least square risk to the direct graph, which is convenient and suitable for regression problems, but not so efficient in some case of classification problems, such as imbalanced data. The reason is that the nodes far away from the decision boundary also contribute large penalty for the shape of the decision boundary. After embedding the data into vector space, we are able to analyze the decision boundary carefully. Nonlinear SVM can show its advantage in such a situation.

Fig.4(b) show the result of the multi-class problem, in which each university is considered as a single class, and then training data are randomly sampled. For SVM, we use one-against-one extension for multi-class problem. For Zhou's algorithm, the multi-class setting in paper [D. Zhou *et al*., 2004] is used. The parameter setting is the same as the binary class experiments. From the figure we can see that significant improvements are achieved by our method. Zhou's method is not very efficient in multi-class problem. Besides the reason we discussed above, another problem of

Zhou's algorithm is the smooth assumption. When the data in one class are scattered in the space, the smooth assumption can not be well satisfied, and the decision boundary will be complicated, especially in the case of multi-class problem. Directly analyzing the decision boundary on the graph is a difficult task. When embedding the data into vector space, complicated geometry analysis can be performed and sophisticated alignment of the boundary can be achieved using methods such as nonlinear SVM.

We also test different dimension settings for classification task. The same parameter setting for SVM is used for training models on different dimensional spaces. Fig.4(c) shows the comparing experimental results of nonlinear SVM on embedded vector spaces where the dimension of the embedded space varies from 4 to 50. From the figure we can see that by first using DGE to embed data into vector space the classification accuracies are higher than Zhou's work in a large range of dimension settings.

Fig.4(d) shows the experimental results of linear SVM on the dimension settings ranging from 4 to 250. The best result is achieved on about 70-dimensional space. In lower dimension spaces, the data may be not linear separable, but still has a rather clear decision boundary. This is why the nonlinear SVM works well in those cases (Fig.3(c)). In a higher dimension the data become more linear separable, and the classification errors get lower. When the dimension is larger than 70, the data become too sparse to train a good classifier, which hinders the classification accuracy. The experimental results suggest that the data on the directed graph may have a latent dimension in a Euclidean vector space which is suitable for further analysis.

7 Conclusion

In this paper, an efficient algorithm DGE for embedding vertices on directed graphs to vector spaces is proposed. DGE explores the inherent pairwise relation between vertices of the directed graph by using transition probability and the stationary distribution of Markov random walks, and embeds the vertices into vector spaces preserving such relation optimally. Experiments show the effectiveness of our method for both embedding problems and applications to classification task.

References

[M. Belkin and P. Niyogi, 2002] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering, NIPS 2002.

[S. Brin and L. Page, 1998] S. Brin and L. Page. The anatomy of a large scale hypertextual web search engine. In Proc. 7th Intl. WWW Conf., 1998.

[F. R. K. Chung, 2005] F. R. K. Chung. Laplacians and the Cheeger inequality for directed graphs. Annals of Combinatorics, 9, 1-19. 2005

[J. Dean and M. Henzinger, 1999] J. Dean and M. Henzinger. Finding related Web pages in the World Wide Web. In Proc. 8th Intl. WWW Conf., 1999.

[A. Langville and C. Meyer, 2004] A. Langville and C. Meyer. Deeper inside PageRank. Internet Mathematics, 1(3), 2004.

[C. Lee Giles *et al*., 1998] C. Lee Giles, K. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In Proc. 3rd ACM Conf. on Digital Libraries, 1998.

[W. Pentney and M. Meila, 2005] W. Pentney and M. Meila. Spectral Clustering of Biological Sequence Data. AAAI 2005: 845-850

[S.T. Roweis and L.K. Saul, 2000] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, volume 290, pages: 2323-2326, 2000.

[B. Schölkopf and A. J. Smola, 2002] B. Schölkopf and A. J. Smola. Learning with kernels. Cambridge, MA: MIT Press, 2002.

[D. Zhou *et al*., 2004] D. Zhou, O. Bousquet, T. Lal, J Weston and B. Schölkopf. Learning with local and global consistency. NIPS 2003

[D. Zhou *et al*., 2005] D. Zhou, J. Huang and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. ICML 2005

Appendix

Proof of the **Proposition 1** in section 3:

$$\sum_u \pi_u \sum_{v,u\to v} p(u,v)(y_u - y_v)^2 = \sum_{(u,v)\in E} \pi_u p(u,v)(y_u - y_v)^2$$

$$= \sum_{v\in V}\left\{\sum_{u\to v} \pi_u p(u,v)(y_u - y_v)^2 + \sum_{u\leftarrow v} \pi_v p(v,u)(y_v - y_u)^2\right\}$$

$$= \sum_{v\in V}\left\{\sum_{u\to v} \pi_u p(u,v) y_u^2 + \sum_{u\to v} \pi_u p(u,v) y_v^2 - 2\sum_{u\to v} \pi_u p(u,v) y_u y_v\right\}$$

$$+ \sum_{v\in V}\left\{\sum_{u\leftarrow v} \pi_v p(v,u) y_v^2 + \sum_{u\leftarrow v} \pi_v p(v,u) y_u^2 - 2\sum_{u\leftarrow v} \pi_v p(v,u) y_u y_v\right\}$$

The first term of the right term of above equation:

$$\sum_{v\in V}\sum_{u\to v} \pi_u p(u,v) y_u^2 = \sum_{(u,v)} \pi_u p(u,v) y_u^2$$

$$= \sum_{u\in V}\left(\sum_{v\leftarrow u} p(u,v)\right)\pi_u y_u^2 = \sum_{u\in V} \pi_u y_u^2 = \sum_{v\in V} \pi_v y_v^2$$

We can do similarly deduction for other terms. Thus

$$\sum_u \pi_u \sum_{v,u\to v} p(u,v)(y_u - y_v)^2$$

$$= \sum_{v\in V}\left\{2\pi_v y_v^2 - \left(\sum_{u\to v} \pi_u p(u,v) y_u y_v + \sum_{u\leftarrow v} \pi_v p(v,u) y_u y_v\right)\right\}$$

$$= 2y^T L y$$