Low-Rank Approximation for Link-Based Ranking

Anonymous Author(s)
Affiliation
Address
email

Abstract

Given a graph G, ranking the importance of nodes by PageRank and the similarity of node pairs by SimRank are two fundamental problems in link-base ranking for many applications such as search and recommendation. In this project, we first show the correlation between PageRank and SimRank in mathematical deduction, and then transform the original iterative computation into a non-iterative form. We propose a uniform low-rank approximation framework for both PageRank and SimRank using Truncated SVD decomposition. In experiments, we show the quality and performance of low-rank approximation with varying number of principle components.

1 Introduction

Authority ranking and similarity ranking are two types of fundamental problems in link-based ranking. Given a graph $G(V, E)$ and $|V| = n$, authority ranking aims to compute an importance vector, where each element depicts the authority degree of the corresponding node. Then, the problem of structural similarity measure is to compute a n-by-n similarity matrix, where the entry on the i-th row and j-th column corresponds to the similarity between node i and j. From the motivation, link-based similarity method is proposed to measure the similarity between two nodes by comparing the environment where a node pair lies — the in-links and out-links, direct and indirect neighbors, all the stuff around this node pair.

The goal of this paper is to efficiently approximate authority and similarity scores in a simple graph using the techniques of low-rank approximation. In this report, for authority ranking of each node, we use PageRank [1], which is a well-known algorithm invented by Google. For similarity ranking between each node pair, we use the structural similarity definition given by [2], whose basic intuition says "two objects are similar, if they are pointed by similar objects". Notice that this intuition is derived from PageRank, whose intuition is "a page is important, because it is pointed by many important pages". SimRank can also be understood from random walk on graphs [3]. Their relationships will be illustrated by mathematics in Section 2. In Section 3, we use truncated SVD to approximate both PageRank and SimRank, aided by Sherman-Morrison Lemma [4].

To clarify, some of proofs from the original SimRank shown in [2] to a uniform form that is very close to PageRank are adopted from [5]. Our exclusive contributions in this article can be listed as follows:

- We show the correlation between PageRank and SimRank by transforming SimRank problem on graph G to an authority ranking problem on the product graph G^2;
- Based on the uniform form of PageRank and SimRank, we show a framework to perform PageRank and SimRank low-rank approximation;
- We choose truncated SVD as the low-rank decomposition and show it is effective in the above framework.

The rest of paper is organized as follows. In Section 2, we introduce PageRank and SimRank and show the mathematical equivalence between them. In Section 3, we state low-rank approximation for PageRank and SimRank, based on a uniform SVD framework. Section 4 shows experimental study and we show a discuss in Section 5.

2 PageRank and SimRank

In this section, we introduce PageRank [1] first, which is an authority ranking approach for each *node* in a graph G. Then we introduce SimRank [2], which is a similarity ranking approach for each *node pair* in a graph G. Thereafter, we take the concept of square graph G^2 and by viewing SimRank scores as the authority of a node in G^2, we show that PageRank and SimRank share great commonality in the principle, by reducing to the same form of iterative computation.

2.1 PageRank

PageRank was introduced in [1]. Given a graph G, PageRank measures the authority of each node in this graph by a value between 0 and 1. From the view of random walk, the PageRank value of a page is the stationary distribution of fractional frequency with which the page will be visited over a long period of time. In concrete, at time step $t+1$, the computation of PageRank for page p_i yields

$$PR(p_i; t+1) = d \sum_{p_j \in M(p_i)} \frac{PR(p_j; t)}{L(p_j)} + \frac{1-d}{N} \tag{1}$$

where d is a damping factor and usually set as 0.85. $M(p_i)$ is the set of all pages that point to p_i and $L(p_j)$ is the number of out-links in page p_j. N is the whole amount of web pages in the dataset. We can rewrite Equation 1 in matrix notation. That is,

$$\mathbf{R}(t+1) = d\mathcal{M}\mathbf{R}(t) + \frac{1-d}{N}\mathbf{1} \tag{2}$$

where $\mathbf{R}_i(t) = PR(p_i; t)$ and $\mathbf{1}$ is the column vector of length N that contains only ones. $\mathcal{M}$ is a column-normalized transition matrix, which is defined as

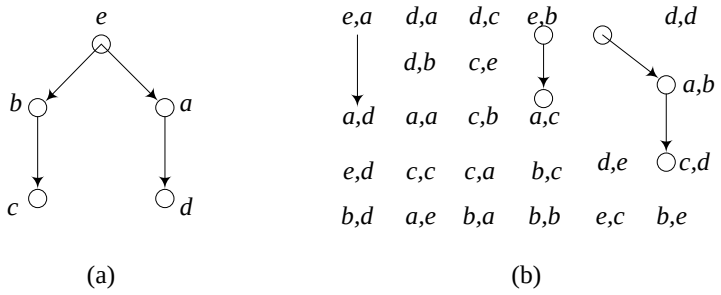
$$\mathcal{M}_{ij} = \begin{cases} 1/L(p_j), & \text{if } j \text{ links to } i \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

2.2 SimRank

The basic problem aroused by the structural similarity measure is, given two nodes in a graph, how can we measure the similarity between these two nodes effectively? The earliest intuition comes from PageRank. PageRank says a node is important, because it is pointed by many important nodes. [2] proposed SimRank, whose intuition is "two nodes are similar, if they are pointed by similar nodes". This intuition is easy to be accepted by human and soon becomes a basic assumption for link-based similarity measure.

We illustrate the idea of structural similarity using the example graph shown in Figure 1(a). To judge the similarity between node c and d, we consider they are similar to each other because they are both pointed by node e. But for the case of node a and b, the situation is a little complicated, because a and b do not have any common in-neighbors. From the human's perspective, we think a and b are still a little similar, because they are pointed by c and d and c and d are similar due to our prior analysis. Let $S(a, b)$ denote the similarity between a and b. From the human's viewpoint, we conclude that $S(c, d) > S(a, b) > 0$. In the following, we will show how SimRank captures human intuitions in the mathematic notation. Before detailed description, there are two basic assumptions:

- The similarity between one node and itself is defined as 1;
- Supposing a and b belong to two disconnected graphs respectively, $S(a, b)$ is defined as 0 whatever takes.



(a) (b)

Figure 1: (a) A small directed graph G. (b) G^2, The product of G. There are 25 nodes and 16 edges in G^2, and we draw all nodes but only 4 edges for simplicity.

Based on the intuition and assumptions, SimRank can be written as

$$\begin{cases} S_{k+1}(a,b) = \frac{d}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} S_k(I_i(a), I_j(b)) \\ S_0(a,b) = \begin{cases} 1 \text{ if } a = b \\ 0 \text{ if } a \neq b \end{cases} \end{cases} \tag{4}$$

d is a damping factor and usually set as 0.8. Basically, SimRank describes an iterative computation and considers all nodes are disconnected before the first iteration. Again, we can rewrite SimRank in matrix notation like this

$$S_{k+1} = dWS_k\tilde{W} + (1-d)I \tag{5}$$

where W is a column-normalized transition matrix and $\tilde{W}$ is its transpose. S is a n-by-n similarity matrix and I is a n-by-n identity matrix. Notice that in most cases, the diagonal elements of S produced by Equation 5 are not equal to 1, which is controversial with the first assumption. However, these elements are near 1 during iterating and will be corrected as 1 after the completion of all necessary iterations. As stated in [5], this is a minor question because the relative ranking of similarity scores will remain unchanged in most cases.

2.3 The Correlation between PageRank and SimRank

In this section, we transform the similarity measure problem on G into an authority ranking problem in the product of G. The product of G is defined as follows:

- If node $a \in G, b \in G$, then there is node $(a,b) \in G^2$;
- If edge $a \to c, b \to d$ in G, there is an edge $(a,b) \to (c,d)$ in G^2.

Basically, every node pair in G forms a node in G^2 and every edge pair in G forms an edge in G^2. Supposing the node size and edge size in G are $|V|$ and $|E|$ respectively, the node size and edge size in G^2 will be $|V|^2$ and $|E|^2$.

We try to understand SimRank from the viewpoint of authority propagation and aggregation. Simply, our target is transforming $S(a,b)$ in G into $R(ab)$ in G^2. For the example of G^2 shown in Figure 1(b), node (d,d) will have an initial authority 1 and the initial authority of (a,b) and (c,d) will be 0. Then on the first time step, (d,d) will propagate its authority to (a,b), so $S_1(a,b) = d = 0.8$. On the second time step, (a,b) propagates its authority to (c,d) and $S_2(c,d) = d^2 = 0.64$. These results are exactly the same we produced using SimRank in Equation 4. So from the viewpoint of authority propagation and aggregation, we construct an equivalent between PageRank and SimRank.

To show how this transformation is performed smoothly and accurately, we introduce Kronecker product and vec-operator in the first, as shown in Definition 1 and 2. The process of this transformation can also be found in [5].

Definition 1 ***Kronecker Product****. Let $A \in R^{s\times t}$, $B \in R^{p\times q}$. Then the Kronecker product of A and B is defined as the matrix*

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1t}B \\ \vdots & \ddots & \vdots \\ a_{s1}B & \dots & a_{st}B \end{bmatrix} \tag{6}$$

Definition 2 ***Vec-Operator****. Let $c_i \in R^s$ denote the columns of $C \in R^{s\times t}$ so that $C = [c_1, \cdots, c_t]$. Then $vec(C)$ is defined to be the st-vector formed by stacking the columns of C on top of one another, i.e.,*

$$vec(C) = \begin{bmatrix} c_1 \\ \vdots \\ c_t \end{bmatrix} \tag{7}$$

The composition of Kronecker product and vec-operator has many useful features. For the purpose of further discussion, we introduce the following theorem.

Theorem 1 *For any three matrices A, B and C,*

$$vec(ABC) = (\tilde{C} \otimes A)vec(B) \tag{8}$$

The theorem is easy to be proved based on the definitions and the proof is omitted here. Performing vec-operator on Equation 5 and applying Equation 8, we get the following deduction:

$$\begin{array}{rc} & vec(S_{k+1}) = vec(dWS_k\tilde{W}) + (1-d)vec(I) \\ \Leftrightarrow & vec(S_{k+1}) = d(\tilde{W} \otimes \tilde{W})vec(S_k) + (1-d)vec(I) \\ \Leftrightarrow & S(t+1) = dMS(t) + (1-d)\Gamma \end{array} \tag{9}$$

$$where \begin{cases} S = vec(S) \\ M = \tilde{W} \otimes \tilde{W} \\ \Gamma = vec(I) \end{cases}$$

As we can see, Equation 9 is very similar to Equation 2. Let $R = S$ and $\Gamma = \frac{\mathbf{1}}{N}$ ($\mathbf{1}$ is a vector with all ones), and actually we can rewrite Equation 2 by Equation 9. The above mathematical transformation makes a successful equivalence between SimRank and PageRank.

3 Low-Rank Approximation

Since we have made a successful equivalence between SimRank and PageRank and take Equation 9 as the uniform summarization, now we can make a low-rank approximation for both of them based on Equation 9. Besides Γ, the only difference is, for PageRank M is a column-normalized transition matrix, for SimRank $M = \tilde{W} \otimes \tilde{W}$ and $\tilde{W}$ is a row-normalized transition matrix. In this section, we use rank-k decomposition to approximate the computation of both PageRank and SimRank.

3.1 PageRank Low-Rank Approximation

The iterative computation of Equation 2 is assumed to be convergent and the proof is given in [1]. In the fix-point we know $R(t+1) = R(t)$ and Equation 2 can be rewritten as

$$R = \frac{(1-d)}{N}(I - dM)^{-1}\mathbf{1} \tag{10}$$

M is positive definite because it is a column-normalized transition matrix. Since among all the possible rank-k approximations, SVD gives the best approximation in terms of squared error, we adopt it as our low-rank approximation method. The rank-k SVD decomposition for M is given by

$$M = U_k\Sigma_k V_k \tag{11}$$

Note that U_k, Σ_k and V_k are truncated SVD while keeping the k largest singular values in Σ. They reflect the k-largest principle components in approximation. We now import an important deduction rule called Sherman-Morrison Lemma [4]. According to this lemma, supposing

$$\Lambda = ((\Sigma_k)^{-1} - d(V_k)(U_k))^{-1} \tag{12}$$

we can obtain the following reduction:

$$(I - dM)^{-1} = I + dU_k \Lambda V_k \tag{13}$$

which can be used in Equation 10 to solve R directly.

3.2 SimRank Low-Rank Approximation

The low-rank approximation for SimRank follows the same way, but the process is more complicated. First, due to fix-point iteration we have $S(t+1) = S(t)$ and can rewrite Equation 9 as

$$vec(S) = (1-d)(I - d(\tilde{W} \otimes \tilde{W}))^{-1} vec(I) \tag{14}$$

The matrix inverse operation is very time-consuming and we use low-rank approximation to approach it. $\tilde{W}$ is positive definite because it is a row-normalized transition matrix. Similarly, we use SVD to approximate $\tilde{W}$ and the rank-k decomposition for $\tilde{W}$ is given by

$$\tilde{W} = U_k \Sigma_k V_k \tag{15}$$

Before the deduction of Equation 14 using SVD and Sherman-Morrison Lemma, we show the following theorem. Due to the space constraint, the proof is omitted here but you can deduct by yourself following definitions of Kronecker product and vec-operator.

Theorem 2 *Let $A \in R^{m \times n}$, $B \in R^{r \times s}$, $C \in R^{n \times p}$ and $D \in R^{s \times t}$. Then*

$$(A \otimes B)(C \otimes D) = AC \otimes BD \tag{16}$$

Based on Theorem 2, we conveniently get the following expression:

$$\tilde{W} \otimes \tilde{W} = (U_k \otimes U_k)(\Sigma_k \otimes \Sigma_k)(V_k \otimes V_k) \tag{17}$$

Based on Sherman-Morrison Lemma [4], supposing

$$\Lambda = ((\Sigma_k \otimes \Sigma_k)^{-1} - d(V_k \otimes V_k)(U_k \otimes U_k))^{-1} \tag{18}$$

we have the following reduction:

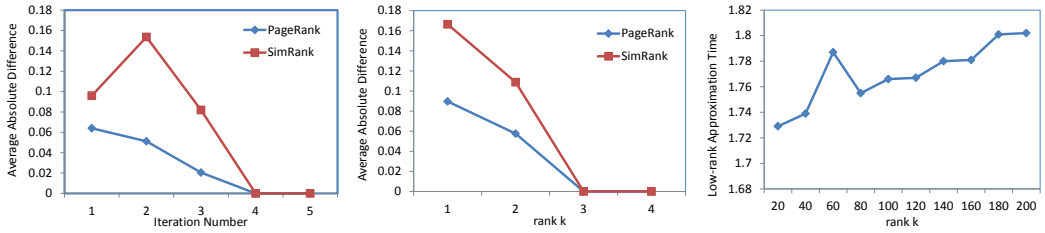
$$(I - d(\tilde{W} \otimes \tilde{W}))^{-1} = I + d(U_k \otimes U_k)\Lambda(V_k \otimes V_k) \tag{19}$$

which in turn can be used in Equation 14 to solve the similarity S.

3.3 A Uniform Low-Rank Approximation Framework

Following major steps summarize the low-rank approximation for both PageRank and SimRank:

1. Do low-rank approximation for transition matrix (Equation 11 or 15);
2. Compute the core matrix Λ (Equation 12 or 18);
3. Compute the inverse matrix (Equation 13 or 19);
4. Compute the resulting authority or similarity vector (Equation 10 or 14).



(a) (b) (c)

Figure 2: For the small graph shown in Figure 1(a), (a) and (b) show the average absolute difference (L_1 Norm) for each iteration and different rank k respectively. (c) shows the running time of PageRank low-rank approximation w.r.t. different rank k, on a graph with 1000 nodes.

4 Experiments

Appendix 1 and Appendix 2 show the python code for SVD approximation of PageRank and SimRank respectively. To evaluate the effectiveness and efficiency of low-rank approximation on PageRank and SimRank, we perform the following experiments and the results are shown in Figure 2.

Approximation Analysis. We take average absolute difference (L_1 Norm) as the quality measure for low-rank approximation. For the small graph shown in Figure 1(a), the average absolute difference for original iterative computation (as presented in Equation 9 and 2) is shown in Figure 2(a). Since it is a very small graph, both PageRank and SimRank converge within 5 iterations. Notice that there is an increase of error for SimRank on the 2nd iteration, because Equation 9 is actually a simplicity of the original SimRank proposed in [2]. Figure 2(b) shows more interesting result, when we increase the rank k. As we can see, the error decreases distinctly as k increases, which accords with the assumption of truncated SVD.

Performance. We are interested in the running time when changing rank k. Figure 2(c) shows the running time of PageRank low-rank approximation w.r.t. different rank k, on a graph with 1000 nodes and 5000 edges. It shows the running time is steadily increasing, not exponentially, with the number of rank k (principle components). The reason is larger k will result in larger U_k, S_k and V_k. We do not show the experiments for SimRank, due to the memory constraint: suppose W is a 100-by-100 matrix, $\tilde{W} \otimes \tilde{W}$ will be a 10000-by-10000 matrix, which is huge. Since $\tilde{W} \otimes \tilde{W}$ is very sparse, sparse matrix computation techniques can be applied to make the computation efficient but it is beyond the scope of this project.

5 Conclusion

In this project, we focus on the low-rank approximation of PageRank and SimRank using truncated SVD. The correlation between PageRank and SimRank is presented in mathematical form, and we successfully transform the original iterative computation into a non-iteration way in approximation.

References

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Technical Report 1999-66, 1999.

[2] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *KDD*, 2002, pp. 538–543.

[3] P. Li, H. Liu, J. X. Yu, J. He, and X. Du, "Fast single-pair simrank computation," in *SDM*, 2010, pp. 571–582.

[4] W. Piegorsch and G. Casella, "Inverting a sum of matrices," *SIAM Review*, vol. 32, pp. 470–470, 1990.

[5] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu, "Fast computation of simrank for static and dynamic information networks," in *EDBT*, 2010, pp. 465–476.

Appendix 1.

Python Code for PageRank Low-Rank Approximation:

```
import numpy as np

# adjacency matrix: from row to column
W = np.array([[0., 0., 0., 1., 0], [0., 0., 1., 0., 0], [0., 0., 0., 0., 0], [0., 0., 0., 0., 0], [1., 1., 0., 0., 0]])
print W

def rowNormalize(W):
  rowNumber = W.shape[0]
  for i in range(0, rowNumber, 1):
    sum = np.sum(W[i, :])
    if sum > 0:
      W[i, :] = W[i, :] / sum
  return W

W = rowNormalize(W)
print W

def pagerank(W):
  # column normalization
  W = rowNormalize(W).T
  d = 0.8
  MAX = 10
  size = W.shape[0]
  one = np.zeros((size, 1))
  for i in range(0, size, 1):
    one[i] = 1. / size
  R = one.copy()
  for i in range(0, MAX, 1):
    RPlus = d * np.dot(W, R) + (1 - d) * one
    print sum(abs(RPlus - R))
    R = RPlus
  return R

IterPR = pagerank(W)
print IterPR

def pagerankSVD(W, k):
  # column normalization
  W = rowNormalize(W).T
  d = 0.8
  size = W.shape[0]
  U, S, V = np.linalg.svd(W)
  S = np.resize(S, [size, 1]) * np.eye(size, size)
  Uk = U[:, 0:k]
  Sk = S[0:k, 0:k]
  Vk = V[0:k, :]
  Gamma = np.linalg.inv(np.linalg.inv(Sk) - d * np.dot(Vk, Uk))
  Inverse = np.eye(size, size) + d * np.dot(Uk, np.dot(Gamma, Vk))
  return ((1 - d) / size) * np.dot(Inverse, np.ones((size, 1)))

SVDPR = pagerankSVD(W, 3)
print SVDPR
```

Appendix 2.

Python Code for SimRank Low-Rank Approximation:

```
import numpy as np

def vec(W):
```

```
    row, column = W.shape
    vec = np.zeros(row * column)
    for i in range(0, column, 1):
        for j in range(0, row, 1):
            vec[i * row + j] = W[i, j]
    return vec

W = np.array([[0., 0., 0., 1., 0], [0., 0., 1., 0., 0], [0., 0., 0., 0., 0], [0., 0., 0., 0., 0], [1., 1., 0., 0., 0]])
print W

def rowNormalize(W):
    rowNumber = W.shape[0]
    for i in range(0, rowNumber, 1):
        sum = np.sum(W[i, :])
        if sum > 0:
            W[i, :] = W[i, :] / sum
    return W

W = rowNormalize(W)
print W

def simrank(W):
    # row normalization
    W = rowNormalize(W.T)
    M = np.kron(W, W)
    d = 0.8
    MAX = 10
    size = W.shape[0]
    vecI = vec(np.eye(size, size))
    S = vecI.copy()
    for i in range(0, MAX, 1):
        SPlus = d * np.dot(M, S) + (1 - d) * vecI
        print sum(abs(SPlus - S))
        S = SPlus
    return S

IterSR = simrank(W)
print IterSR

def simrankSVD(W, k):
    # row normalization
    W = rowNormalize(W.T)
    d = 0.8
    size = W.shape[0]
    vecI = vec(np.eye(size, size))
    U, S, V = np.linalg.svd(W)
    S = np.resize(S, [size, 1]) * np.eye(size, size)
    Uk = U[:, 0:k]
    Sk = S[0:k, 0:k]
    Vk = V[0:k, :]
    Gamma = np.linalg.inv(np.linalg.inv(np.kron(Sk, Sk)) - d * np.dot(np.kron(Vk, Vk), np.kron(Uk, Uk)))
    tmp = d * np.dot(np.kron(Uk, Uk), np.dot(Gamma, np.kron(Vk, Vk)))
    Inverse = np.eye(tmp.shape[0], tmp.shape[1]) + tmp
    return (1 - d) * np.dot(Inverse, vecI)

SVDSR = simrankSVD(W, 2)
print SVDSR
```