NOTES
C.E TSOURAKAKIS

ABSTRACT. Notes for the graph mining lecture.

1. Matrix Vector Multiplication

The goal of the corresponding slide is to present the two ways of thinking of matrix vector multiplication.

Consider matrix $A \in \mathbb{R}^{mxn}$ and a vector $x \in \mathbb{R}^n$ and the matrix vector multiplication $Ax = b$, $b \in \mathbb{R}^m$.

- The coordinates of vector b are given by: $b_i = \sum_{j=1}^{n} a_{ij}x_j$ for $i = 1 \ldots m$. Thus, we view the matrix vector multiplication as an **operation** that returns a vector b, whose coordinates are given by the inner products of the rows of A with x.
- $b = Ax = [\boldsymbol{a_1}| \ldots |\boldsymbol{a_n}]x = x_1\boldsymbol{a_1} + \ldots + x_n\boldsymbol{a_n}$. Thus b is a linear combination of the columns of matrix A.

From now on, think A as **linear map**, i.e., taking as "inputs" vectors in $\mathbb{R}^n$ and returning "outputs" in $\mathbb{R}^m$. The map $x \longmapsto Ax$ is linear because $A(x + y) = Ax + Ay$ and $A(\alpha x) = \alpha Ax$, where $\alpha \in \mathbb{R}$.

Consider now $x = A^{-1}b$, where A^{-1} is the inverse of the square matrix A. Again there are two ways to see x:

- The obvious one: $A^{-1}b$, apply A^{-1} to b.
- The i-th coordinate x_i of the vector x is the coefficient of the i-th column of A so that this holds: $x_1\boldsymbol{a_1} + \ldots + x_n\boldsymbol{a_n} = \boldsymbol{b}$

The second view point is effectively a change of the basis. x is the vector of the coordinates of a different representation of b, i.e., we just changed the basis from the standard one $E = \{e_1, \ldots, e_n\}$ to the basis given by the columns of A (why is it a basis?). This is very important. For example later, I will explain that seeing things from this point of view, will allow us to say that effectively all real symmetric matrices are diagonal. And it is the eigendecomposition that will give us the correct basis of representation.

2. Orthogonality

The point of the corresponding slide is that orthogonality implies independence. E.g., if you are given two non-zero vectors $\{x_1, x_2\} \in \mathbb{R}^2$ such that $x_1^T x_2 = 0$, you know immediately that these two vectors form a basis for $\mathbb{R}^2$. Orthogonality is a fundamental concept in linear algebra, more in any standard textbook such as "Linear Algebra and Application" by G. Strang.

3. Vector Norms

A norm is a function, i.e., it takes as "input" a vector in $\mathbb{R}^n$ and returns a real number. The most important family of vector norms are the p-norms. The rest of the lecture will refer (at certain points) to the 2-norm.

2-norm ($x \in \mathbb{R}^n$)

$$||x||_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} = \sqrt{x^T x} \tag{1}$$

4. Matrix Norms

Consider $A \in \mathbb{R}^{mxn}$. Discriminate two types of matrix norms, the ones that are induced by vector norms and the ones that are not. We will talk for the 2 induced norm and the Frobenius norm.

Observe that even if you permute randomly the elements of your matrix the Frobenious norm will remain unchanged. The Frobenious norm captures the "average" stretch and the $L2$ norm captures the maximum stretch [1].

Just for illustration purposes, I ran the following command in my laptop norm_demo([1 2;3 4]) after downloading the code from this link `http://www.caam.rice.edu/~caam453/norm_demo.m`. You see in figure 1 the images of the unit balls defined by the 2-norm, the 1-norm and the ∞-norm of $\mathbb{R}^2$ and their images under the matrix A:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Just by looking the unit balls can you say how the $1-$ norm and the ∞- norm of a vector x are defined? See also `http://en.wikipedia.org/wiki/Lp_space`.

5. SVD

Remember (again): we view matrix $A \in \mathbb{R}^{mxn}$ as an operator. Let's see the geometry behind the Singular Value Decomposition of $A = U\Sigma V^T$

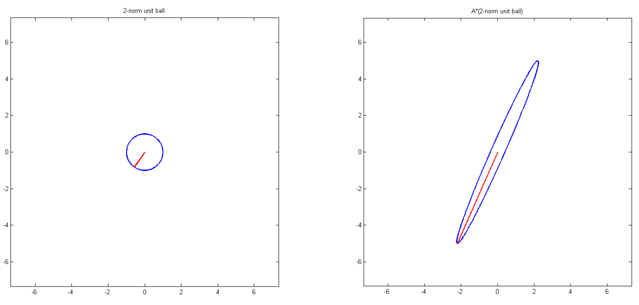
- The singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$ are the lengths of the principal semiaxes of the hyperellipse.
- The left singular vectors u_i's give the direction of the axes.
- The right singular vectors v_i's give the points in $\mathbb{R}^n$ on the unit circle that are mapped on the hyperellipse along the semiaxes.

Given this interpretation, find on your own the SVD of simple matrices 2x2 [2], e.g., diagonal matrices with positive, negative elements, the all 1s matrix etc.

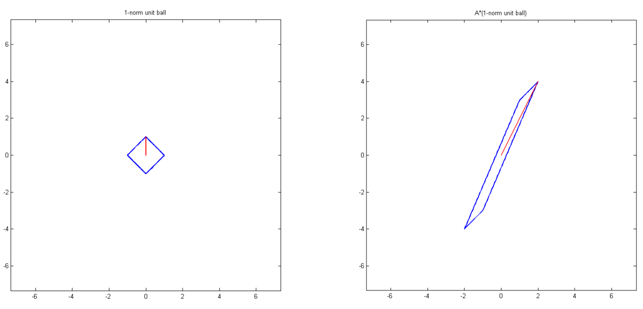
One standard use of the SVD is to perform dimensionality reduction.

[1] stretch that some vector x undergoes when we apply A to it

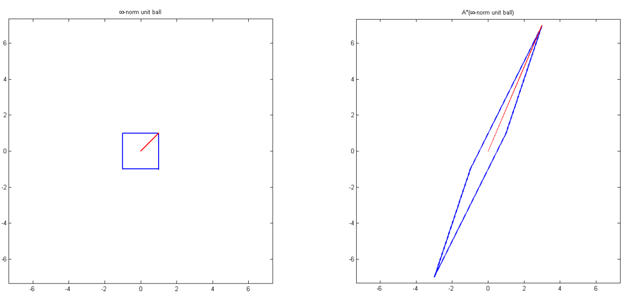
[2] I remember this was one of the problems in the last year's exam



(a) 2-norm



(b) 1-norm



(c) ∞-norm

FIGURE 1. Unit balls in $\mathbb{R}^2$ and their images under matrix A.

5.1. **SVD and dimensionality reduction.** Let's assume in the following that the rank of $A \in \mathbb{R}^{mxn}$ is equal to r. From the SVD of A it follows that: $A = \sum_{j=1}^{r} \sigma_j u_j v_j^T$. There is a subtle point here, that is another application of SVD. The rank r equals to the number of non-zero singular values (why? [3]) .

The following two theorems show the optimality of SVD with respect to the $L2$ and Frobenious norm as a dimensionality reduction tool.

Theorem 1. *For any $0 \leq k < r$ define $A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T$. The following equations hold for any matrix $B \in \mathbb{R}^{mxn}$ whose rank is k or less:*

$$||A - A_k||_2 \leq ||A - B||_2 \tag{2}$$

$$||A - A_k||_F \leq ||A - B||_F \tag{3}$$

A standard application of the SVD as a dimensionality reduction tool, is image compression. Since it is a nice application and easy to implement in MATLAB let's see it.

```
I = imread('xilouris.jpg');
red = double(I(:,:,1));
green = double(I(:,:,2));
blue = double(I(:,:,3));
figure
```

[3]The answer is because U and V are of full rank, and therefore rank(A)=rank(Σ)

```
imshow(I);
title('Original image');
% compression using k rank approximation
k = [10 30 60];


for i = 1 : length(k)
    % svd for the red
    [u_r s_r v_r] = svds(red,k(i));
    red_k = uint8(u_r * s_r * transpose(v_r));
    % svd for the green
    [u_g s_g v_g] = svds(green,k(i));
    green_k = uint8(u_g * s_g * transpose(v_g));
    % svd for the blue
    [u_b s_b v_b] = svds(blue,k(i));
    blue_k = uint8(u_b * s_b * transpose(v_b));
    im(:,:,1) = red_k;
    im(:,:,2) = green_k;
    im(:,:,3) = blue_k;
    figure
    imshow(im)
    mytitle = strcat('Rank k=',int2str(k(i)));
    title(mytitle)
end
```

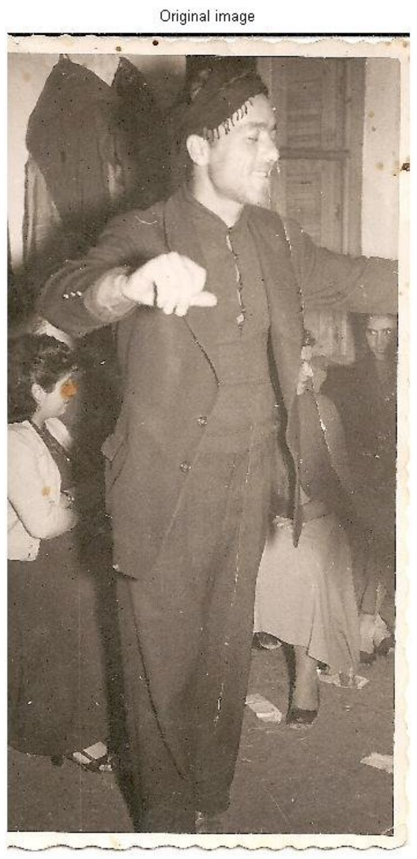
The result of running this simple script on the two images of figure 2 are shown in figures 3 and 4. Can you make an estimate of the savings we get, given that the initial matrix is nxm and we make a k rank approximation?

Since computing the SVD is expensive, randomized algorithms exist that "exchange" some accuracy (hopefully small) in favor of speed. Let's try out the method suggested in the paper *A randomized singular value decomposition algorithm for image processing*, by **E. Drinea, P.Drineas, P. Huggins**. You can see what we get for $k = 90$ in figure 5.1. Can you really tell the difference? Therefore, the proposed method which relies on a simple sampling procedure, gives a nice trade-off between losing accuracy and speedup.

5.2. **HITS algorithm.** Hyperlink-Induced Topic Search (HITS) was developed by J. Kleinberg and is an algorithm to rank web pages given a query. The algorithm computes the hubs and the authorities in the graph. Hubs are pages that point to the authority nodes, and at the same time the authorities are nodes that are pointed by the hubs! "Hubs and authorities exhibit what could be called a mutually reinforcing relationship" (Authoritative Sources in a Hyperlinked Environment, by J.Kleinberg). So the user wants to see the pages with the highest authorities **w.r.t to his query** as the result of his search and the hubs are good pointers to what the user wants to see. We associate therefore two numbers to each web page, let a_i and h_i be the authority and the hub weight of the i-th node, the higher they are the better they are considered.

Call A the adjacency matrix of the graph, and let $\boldsymbol{h}, \boldsymbol{a}$ be the vectors containing in the i-th coordinate the hub weight and the authority of the i-th node. How do we compute them? The following equations hold: $\boldsymbol{h} = A\boldsymbol{a}$ and $\boldsymbol{a} = A^T\boldsymbol{h}$ (why?). These two iterative rules converge under the realistic assumption of the existence of a non-zero spectral gap, to the top right ($\boldsymbol{a}$) and the top left ($\boldsymbol{h}$) singular vectors of A [4]. To see why, substituting, gives that the convergence points satisfy: $\boldsymbol{a} = A^TA\boldsymbol{a}$ and $\boldsymbol{h} = AA^T\boldsymbol{h}$. Power method.

[4]By top we mean corresponding to the largest singular value σ_1



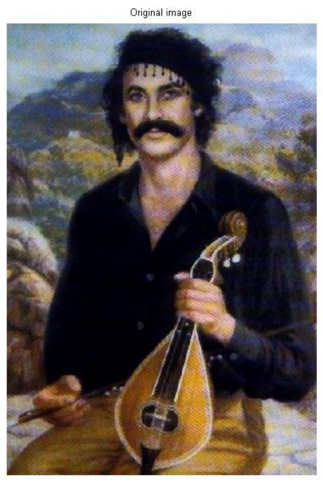


Figure 2. Original Images

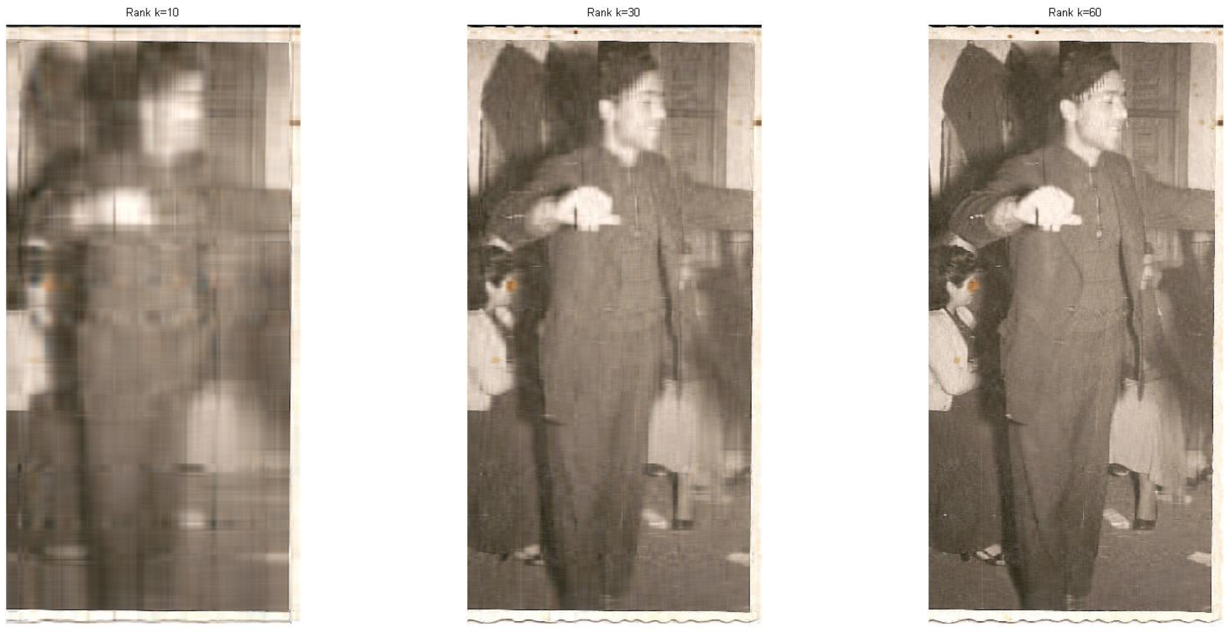


Figure 3. Result on first image after applying SVD for k=10,30 and 60.

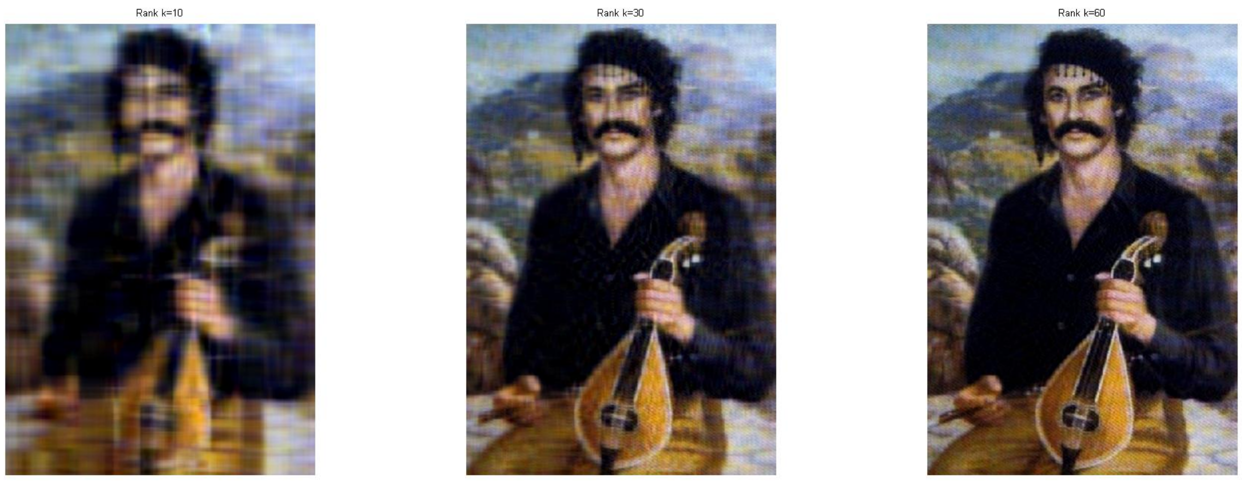
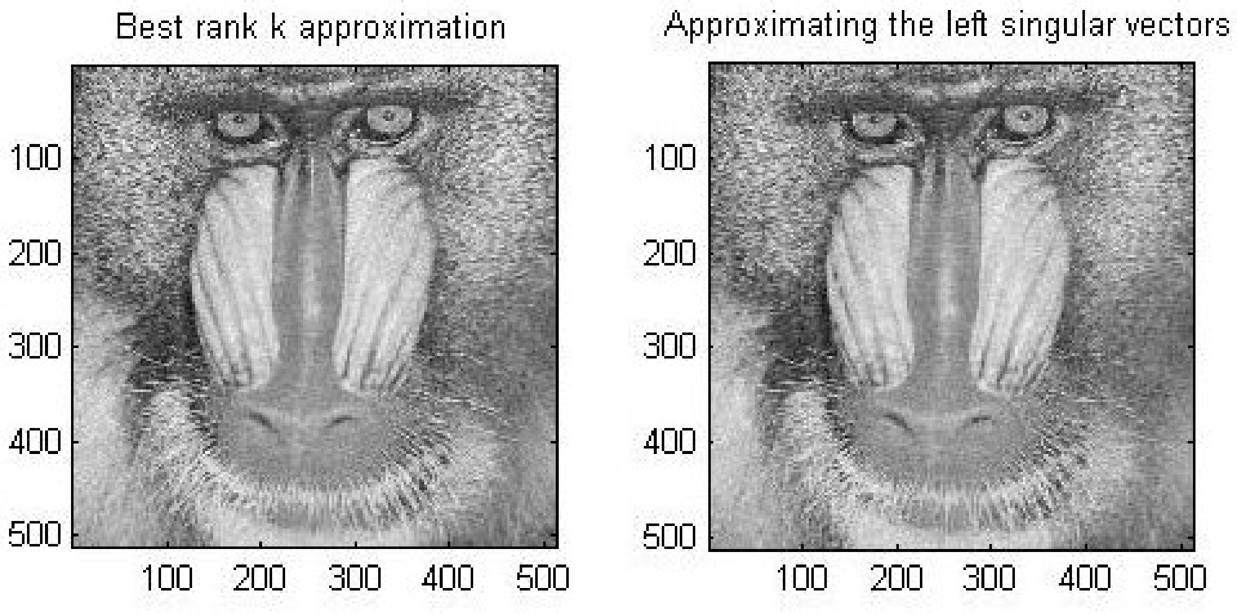


FIGURE 4. Result on second image after applying SVD for k=10, 30 and 60.


Best rank k approximation
Approximating the left singular vectors
100
200
300
400
500

5.3. **Triangle Counting.** Problem: Given a graph simple, undirected graph $G(V, E)$, count the number of triangles. Fastest Method: $O(n^{2.376})$ (matrix multiplication). Listing Methods: In practice for large graphs listing methods are preferred. The easiest ones after the obvious $O(n^3)$, are the NODEITERATOR and the EDGEITERATOR. Idea: consider each node (edge) and count the number of triangles that the node (edge) under consideration participates in.

Running time $O(nd_{max}^2)$, which in dense graphs becomes $O(n^3)$.

Can we do anything better than this?

This is another application of the SVD, which works well under certain conditions that seem to hold frequently for "real-world" networks.

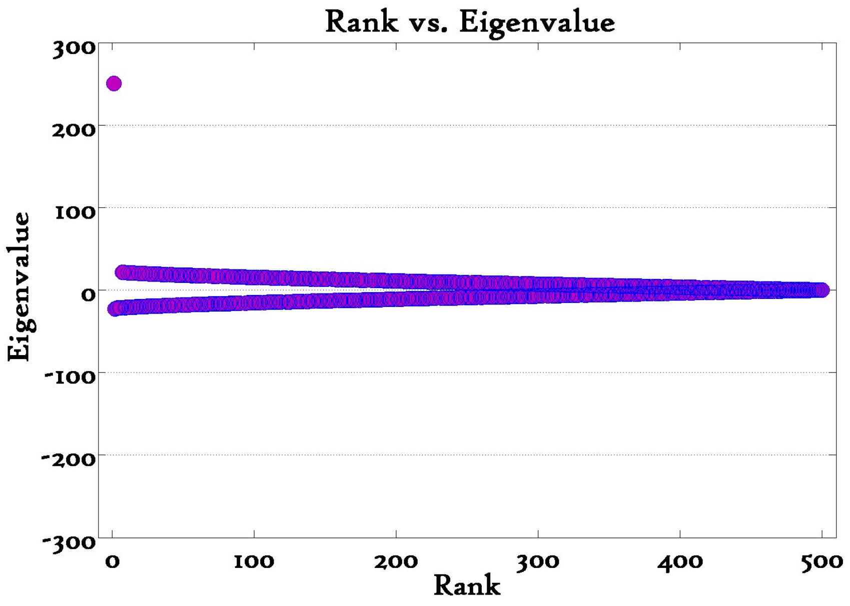


FIGURE 5. This figure plots the value of the eigenvalue vs. its rank for $G_{n,\frac{1}{2}}$.

To reduce the problem of counting triangles to an SVD decomposition we have to represent the graph as a matrix. Three are the major matrix representation of a graph: the adjacency matrix, the Laplacian and the Normalized Laplacian. The Laplacian and the its normalized (by the degrees) version [5] are the ones that "pop-out" all the time due to their amazing properties (Spectral Graph Theory) but for the triangles, the adjacency matrix will do the work and the reason is simple: if A is the adjacency matrix, then through A^k we can count easily paths of length k (think why).

Theorem 2 (EIGENTRIANGLE). *The total number of triangles in a graph is equal to the sum of cubes of its adjacency matrix eigenvalues divided by 6, namely:*

$$\Delta(G) = \frac{1}{6}\sum_{i=1}^{n} \lambda_i^3 \tag{4}$$

Theorem 3 (EIGENTRIANGLELOCAL). *The number of triangles Δ_i that node i participates in, can be computed from the cubes of the eigenvalues of the adjacency matrix*

$$\Delta_i = \frac{\sum_j \lambda_j^3 u_{i,j}^2}{2} \tag{5}$$

where $u_{i,j}$ is the j-th entry of the i-th eigenvector.

Let's consider first the random graph $G_{n,\frac{1}{2}}$ before going to the real world networks [6].

You can see the Eigenvalue-Rank plot in figure 5.

This figure suggests that if we consider just the first eigenvalue of the adjacency matrix, we can get a very good approximation in the number of triangles. If we had a similar spectrum for 'real-world" networks, then we could apply the same idea there. So let's see the same type of plot for a small[7] "real world" network in figure 6.

[5]when the graph is not d-regular, else we are talking about the same thing. Why?

[6]Obviously, a good approximation is the expected number of triangles is $\frac{\binom{n}{3}}{8}$

[7]So that we can easily plot the whole spectrum

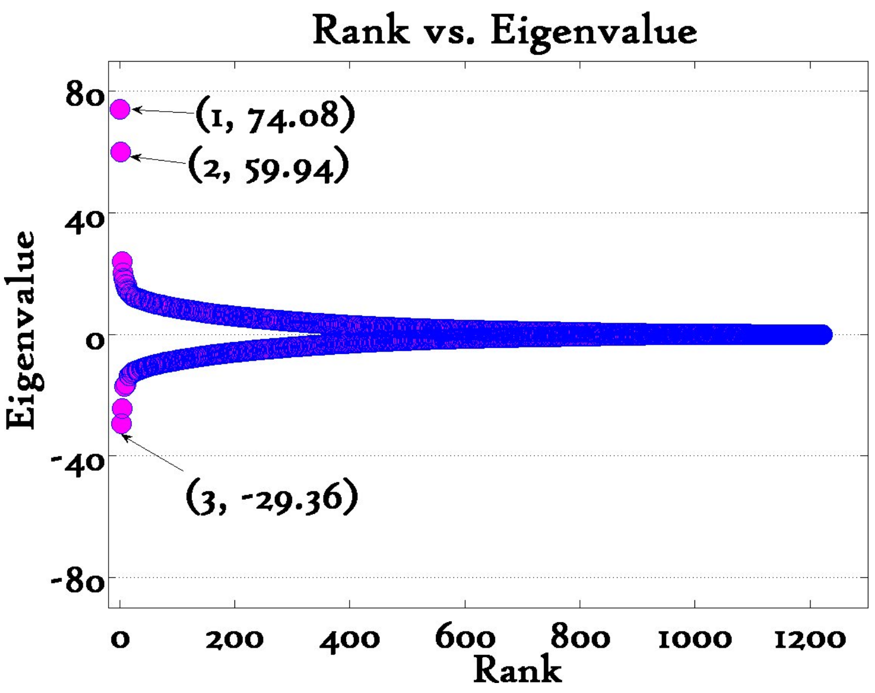


FIGURE 6. This figure plots the value of the eigenvalue vs. its rank for a network with ≈ 1,2K nodes, 17K edges.

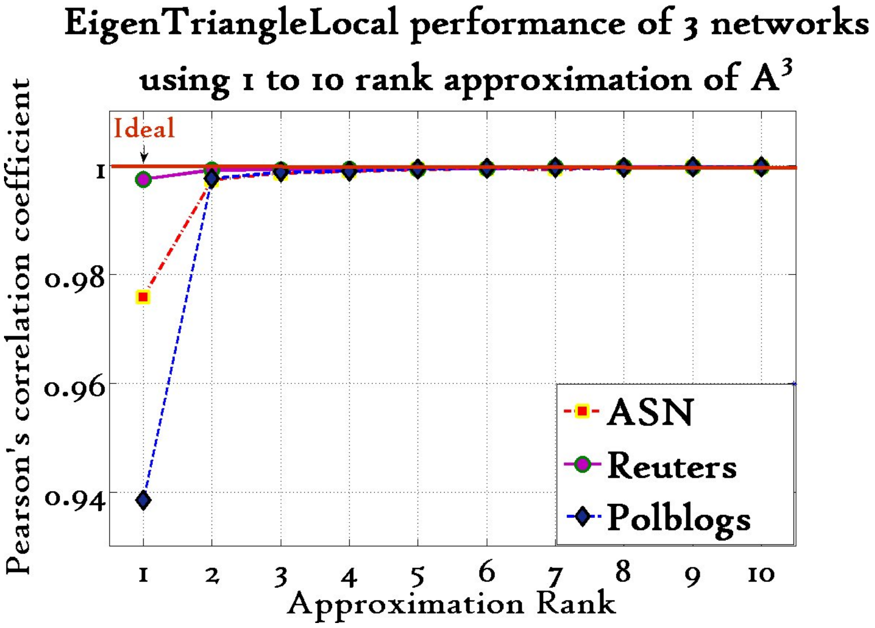


FIGURE 7. Local Triangle Reconstruction for three networks, Flickr, Pol Blogs and Reuters.

In figure 7 you see the quality of the estimate of triangles per node if you apply this idea to three real world networks (for the details, `http://www.cs.cmu.edu/~ctsourak/tsourICDM08.pdf`) It is remarkable how many networks exhibit this property [8].

I did not mention anything for SVD, but I was just talking for the eigenvalues of the adjacency matrix. However, this is essentialy the SVD, since the graph is undirected and therefore A is symmetric. Therefore the absolute

[8]Why you would not apply this idea to a network that you know that is of the form AA^T?Can you think of any such networks?

values of the eigenvalues are the singular values and the signs can be retrieved by looking the left and singular vectors (how?).

6. PageRank

PageRank (PR) is the algorithm reportedly used by Google. PR models the web as a directed graph and assigns to each node in the graph a numerical score.

Few words first on random walks. Consider a particle that jumps from node to node, with this simple rule. Given its current position, i.e., let's say node i, it chooses uniformly at random one of the neighbors of i. Observe that this simple rule for a given graph defines a Markov chain. Think of the nodes as the states of the Markov Chain. The transition matrix is a nxn matrix where $P_{ij} = \frac{1}{outdeg(i)}$ if $i \to j$ exists, otherwise $P_{ij} = 0$.

The PR is the limiting distribution of the iterative process $x_j^{(t+1)} = \sum_i P_{ij} x_j^{(t)} = \sum_{i \to j} \frac{x_i^{(t)}}{outdeg(i)}$.

Observe that in a directed graph we can have dangling nodes. If during the random walk we get there then we are "stack". So what can we do about it?

Consider an indicator vector δ that has its i-th coordinate equal to 1 if and only if the i-th node is dangling. Now consider matrix $P' = P + \delta * v^T$ where v is some probability distribution vector over the nodes, called the teleportation distribution. Think: Which property does P' have that P does not?[9] One can choose the v vector to assign uniform teleportation, but non-uniform v leads to Personalized PR! The next step is to make the Markov Chain irreducible, i.e., every node can be reachable from any other node. In other words, we want to make the graph **strongly connected**.

The reason to do so, is to satisfy the conditions of the Perron-Frobenious theorem `http://en.wikipedia.org/wiki/PerronFrobenius_theorem`. Therefore we consider the transition matrix $P^{''} = cP' + (1-c)(1,\ldots,1)v^T$ where c is the teleportation coefficient (typical value c=0.85). So now we see what is PR. It is the power method, namely:

$$x^{(t+1)} = (P'')^T x^{(t)} \tag{6}$$

where now we know that P'' is row stochastic and and irreducible. The convergence point of the iterative procedure is the principal eigenvector x

$$x = (P'')^T x \tag{7}$$

Now, as an exercise consider a connected undirected graph $G(V,E)$. Why don't we need to form P', P'' [10]? Show that the PR is given effectively by the degrees of the nodes.

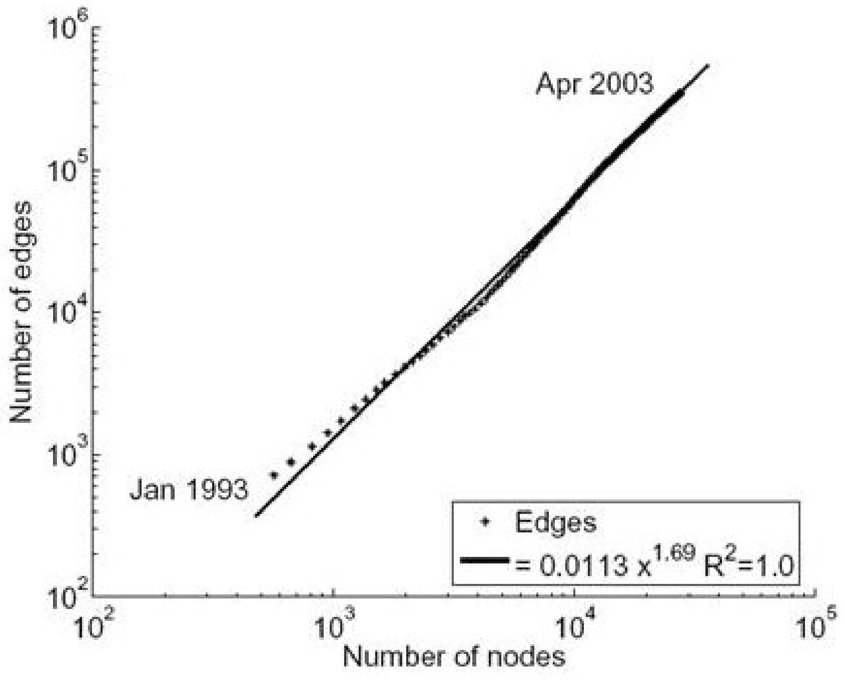
7. Graph Patterns

One of the tasks of graph mining is the detection of patterns in "real-world" graphs. These patterns are important for many reasons, such as link prediction, abnormal node detection but you can think of more. Knowing patterns provides us a way to make predictions and also say what is normal or not with respect to those patterns.

The power laws I have in the beamer slides are:

[9] P' is row-stochastic whereas P had some rows with sum equal to 0.

[10] Remember why we created each one



(a) arXiv

- Degree distribution
- Spectrum, main focus Top Eigenvalues
- Hop plot and the celebrated Small World Phenomenon
- Time evolving graphs
- Triangles

The other set of slides (the powerpoint files) contain more power laws, including the ones in the beamer. Few words on the densification law, the small world phenomenon, communities and Kronecker graphs.

Densification. Consider a given network. Think about it as in the following: initially it was the empty graph and through time it became as it is now. How did this happen however? How did the network **evolve**? Therefore it is interesting to experiment with graphs for which we have different snapshots and find patterns of evolution. Leskovec, Kleinberg and Faloutsos did this and found out -among other things- the densification power law which you see in figure 7. Jure's whole thesis is on Dynamics of real world networks `http://www.cs.cmu.edu/~jure/pubs/thesis/jure-thesis.pdf`.

Small World Phenomenon. This is one of the most surprising patterns in real-world networks. The first experiment was conducted by Milgram and even if the way that it was conducted has been criticized, see `http://en.wikipedia.org/wiki/Small_world_phenomenon` and `http://en.wikipedia.org/wiki/Six_degrees_of_separation`, it was pioneering work. The term "six-degrees" of separation has been used to describe the fact that our world is small, in the sense that it takes few "hops" for **any** person to reach **any** other person. Modeling the world as a graph, this means that the diameter, i.e., the longest of the shortest paths between two nodes is surprisingly small.

Some reading material:

- Kleinberg's web page under the section Small-World Phenomena and Decentralized Search.

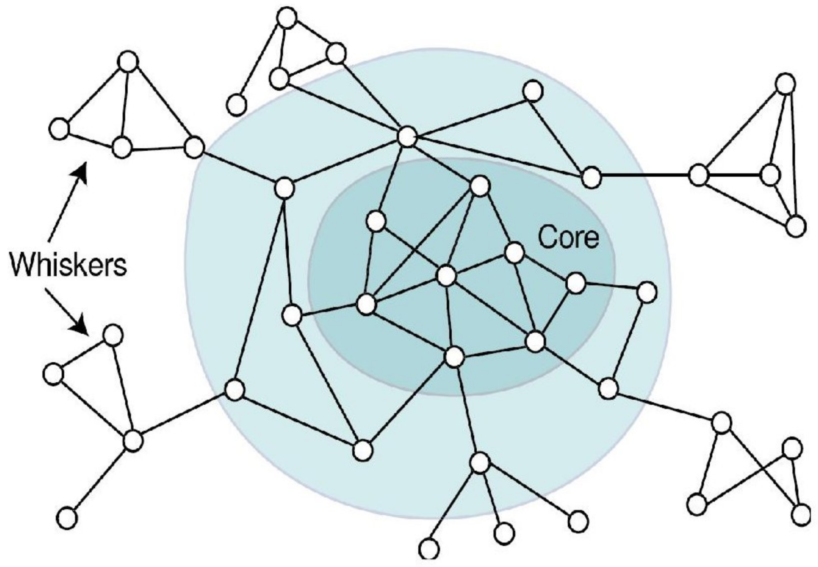


Figure 8. Figure from the Leskovec et al. paper. Caricature of how a real world network looks like. (courtesy of J. Leskovec).

- "Planetary-Scale Views on a Large Instant-Messaging Network " by **J. Leskovec, E. Horvitz**

FYI: A whole book (probably more than one) is devoted to this phenomenon, "Six Degrees: The Science of a Connected Age" by **D.Watts**.

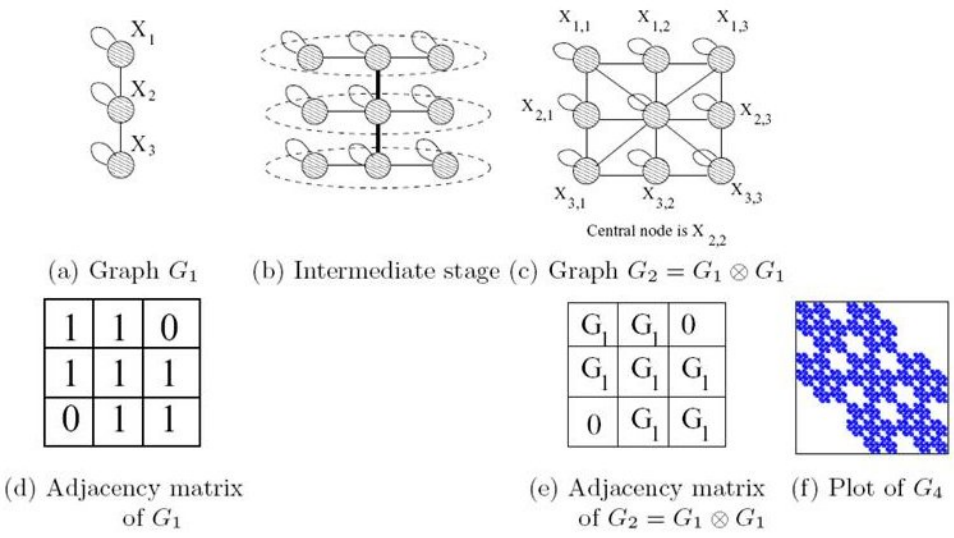
Communities. A lot of research in graph mining is focused on finding the communities in real world networks. A recent paper *Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters* by **J. Leskovec, Lang, Dasgupta, Mahoney** (`http://arxiv.org/abs/0810.1355`) presents surprising, counter-intuitive facts about communities. Among them, real world networks have a big core which looks "random", in other words this core cannot be partitioned in nice clusters, and then attached to its peripherym there are clusters that experimentally have a small size. A caricature of this fact is shown in figure 8.

Kronecker graphs. The following properties hold for deterministic Kronecker graphs:

- Power-law-tail in- and out-degrees
- Power-law-tail scree plots
- constant diameter
- perfect Densification Power Law
- communities-within-communities

Note that stochastic Kronecker graphs introduced by Leskovec and Faloutsos produce more realistic graphs, by a biased toin cossing procedure after doing the Kronecker multiplication.

Read *Kronecker Graphs: an approach to modeling networks* by **Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, Zoubin Gharamani** for all the details and much more `http://arxiv.org/abs/0812.4905`.



(a) Graph G_1 (b) Intermediate stage (c) Graph $G_2 = G_1 \otimes G_1$

(d) Adjacency matrix of G_1 (e) Adjacency matrix of $G_2 = G_1 \otimes G_1$ (f) Plot of G_4

Figure 9. Deterministic Kronecker Graphs