Application of random matrices

We will talk about three of the many practical applications of matrices:

- Application in Google web search engine
- Application in finance
- Application in image processing

Application in Google

The PageRank algorithm

If one goes to a website, it might find some links on that page which points to other websites. If there are many websites which points to a specific webpage, then that website must be pretty important.

If one goes to a website, it might find some links on that page which points to other websites. If there are many websites which points to a specific webpage, then that website must be pretty important.

The page rank algorithm gives each page a rating of its *importance*. The importance of a page is defined recursively by the importance of the pages which points to it.

If one goes to a website, it might find some links on that page which points to other websites. If there are many websites which points to a specific webpage, then that website must be pretty important.

The page rank algorithm gives each page a rating of its *importance*. The importance of a page is defined recursively by the importance of the pages which points to it.

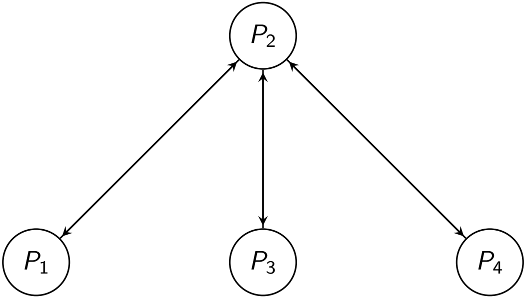
One way to think about it is to start surfing on the internet through the links from one page to another. The page rank of a website is proportional to the probability that we end up on that website after a specific very long time.

Mathematically, we represent the web as a directed graph with vertices $P_1, P_2, .., P_n$ for some n and we say that (P_i, P_j) is an edge iff the webpage P_i links to P_j.

Mathematically, we represent the web as a directed graph with vertices $P_1, P_2, .., P_n$ for some n and we say that (P_i, P_j) is an edge iff the webpage P_i links to P_j.

The PageRank algorithm generates a vector π which satisfy that the rank of the page P_i is the sum of the ranks of the pages that point to it, divided by their degree.

For example, one might check that for the following graph, the PageRank algorithm will generate $\pi = (1/6, 1/2, 1/6, 1/6)$.



To write this in terms of matrices, let A be the adjacency matrix of the directed graph. Let $d_{out}(P_i)$ be the out-degree of the vertex P_i and let

$$D := \mathrm{diag}(d_{out}(P_1), d_{out}(P_2), ..., d_{out}(P_n)).$$

Assume that $d_{out}(P_i) \neq 0$ for all i so we can continue surfing at any time. Let

$$W := A^T D^{-1}.$$

Then π satisfies:

$$\pi = W\pi,$$

So π is an eigenvector of W with eigenvalue 1.

Application in finance

The covariance matrix

A useful term in finance is that of a **stock**, which, simplified represents a proportion of the *value* of a company. This is not a rigorous definition, but it is enough to understand the example.

A useful term in finance is that of a **stock**, which, simplified represents a proportion of the *value* of a company. This is not a rigorous definition, but it is enough to understand the example.

Let's suppose we have some money and we want to invest it in stocks $A_1, A_2, .., A_{10}$. We do not want to take too much risk, so we decide that we want the variance of our portfolio to be upper bounded by some $c > 0$. From the past data, we can estimate the covariance matrix of the stocks $A_1, ..., A_{10}$, call this matrix V and note that V has to be positive definite, i.e. all eigenvalues are positive. We can also compute the *estimated returns* of the stocks, call them $r_1, .., r_{10}$ and let $r = (r_1, .., r_{10})$.

In terms of matrices, we want to find the proportion of the portfolio, $x \in \mathbb{R}^{10}$ such that

- $x^T \cdot r$ is maximized
- $x^T V x \leq c$

In terms of matrices, we want to find the proportion of the portfolio, $x \in \mathbb{R}^{10}$ such that

- $x^T \cdot r$ is maximized
- $x^T V x \leq c$

Let the singular value decomposition of V be:

$$V = U^T \Sigma U,$$

where $\Sigma := \text{diag}(\sigma_1^2, ..., \sigma_{10}^2)$ and the columns of U are u_i, where σ_i^2's and v_i's are the eigenvalues and the associated eigenvectors of V.

Let $y = x \cdot U \cdot \Sigma^{1/2}$ and $r' = \Sigma^{1/2} \cdot U \cdot r$. Our task becomes to find y with $\|y\|_2 \leq c$ which maximizes

$$y^T \cdot r'.$$

Let $y = x \cdot U \cdot \Sigma^{1/2}$ and $r' = \Sigma^{1/2} \cdot U \cdot r$. Our task becomes to find y with $\|y\|_2 \leq c$ which maximizes

$$y^T \cdot r'.$$

This implies that y' is collinear with r' and so we can compute x.

Application to Image Processing

Principal Component Analysis

An image can be view as an $m \times n$ matrix, where each entry of the matrix correspond to a pixel. In general, every color can be computed by a mixture of red blue and green, so we can associate each color with a triplet in which each coordinate is the amount of red, green and respectively blue in that color.

An image can be view as an $m \times n$ matrix, where each entry of the matrix correspond to a pixel. In general, every color can be computed by a mixture of red blue and green, so we can associate each color with a triplet in which each coordinate is the amount of red, green and respectively blue in that color.

The problem with a picture which has high resolution is that it contains a lots of informations which need to be stored, so it will take a lot of space. In many situations we do not necessarily need the high resolution, but a very good approximation of it.

For simplicity of the argument, let's suppose we are working with a square black and white picture, so the entries of our matrices can be seen as real numbers, 0 meaning white and 1 black. Let A be the associated matrix. In general A is full ranked, so if we want to store the image, we have to store n^2 bits.

For simplicity of the argument, let's suppose we are working with a square black and white picture, so the entries of our matrices can be seen as real numbers, 0 meaning white and 1 black. Let A be the associated matrix. In general A is full ranked, so if we want to store the image, we have to store n^2 bits. Let

$$A := \sum_{i=1}^{n} \sigma_i v_i^* v_i,$$

be the singular value decomposition of A.

For simplicity of the argument, let's suppose we are working with a square black and white picture, so the entries of our matrices can be seen as real numbers, 0 meaning white and 1 black. Let A be the associated matrix. In general A is full ranked, so if we want to store the image, we have to store n^2 bits. Let

$$A := \sum_{i=1}^{n} \sigma_i v_i^* v_i,$$

be the singular value decomposition of A. Define

$$A_k := \sum_{i=1}^{k} \sigma_i v_i^* v_i.$$

The idea is that if the eigenvalues of A are far from each other, then the matrix A_k is a good approximation for A.

For example

$$A - A_k = \sum_{i=k+1}^{n} \sigma_i v_i^* v_i,$$

which implies:

$$\|A - A_k\|_{Fr}^2 = \sum_{i=k+1}^{n} \sigma_i^2.$$

Note that in order to store A_k we only need $kn + k$ bits as we only store the first k eigenvalues and eigenvectors.

The following example appears in it Principal Component Analysis (PCA) by Vaclav Hlavac and it uses only the first 4 eigenvectors to reconstruct a 231×261 pixels image.

Questions?