# Matrix Perturbation Theory and its Applications

Yaniv Shmueli

School of Computer Science
Tel-Aviv University

November 29, 2012

# Prologue

Our hero is the intrepid, yet sensitive matrix A.
Our villain is E, who keeps perturbing A.
When A is perturbed he puts on a crumpled hat: $\tilde{A} = A + E$.

G. W. Stewart and J.-G. Sun, Matrix Perturbation Theory (1990)

# Introduction

- In a nutshell : how does a small change in the input affects the output?
- Given a matrix $A$ and some function $\phi$ which operates on $A$, we are interested in understanding how a small perturbation added to the matrix, affects the behavior of $\phi$.
- That is, to understand the relation between $\phi(A)$ and $\phi(A + \epsilon)$, where $\epsilon$ is a small perturbation (can be noise).
- Interesting $\phi$ operators : finding the singular values of $A$, the eigenvectors $A$, $\|A\|$ and so on.

# Introduction

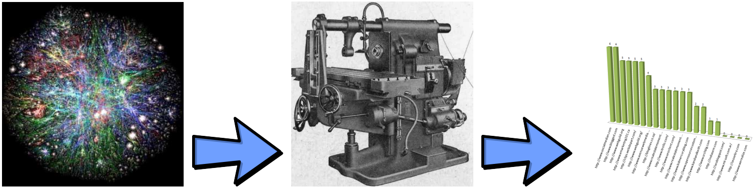
- Example application(1) : Computation of the Google page rank algorithm - how to update the ranks without recomputing the entire algorithm.
- Example application(2) : updating a training set profile with small changes to the input set.
- Books
    - Matrix Analysis - R. Bhatia.
    - Matrix Perturbation Theory - Stewart and Sun.

# Example Application - Google PageRank Calculation I

- Pagerank : The importance of a web page is set by the number of important pages pointing to it.
- $r(P) = \sum_{Q \in B_P} \frac{r(Q)}{|Q|}$ where $B_P$=[all pages pointing to P], $|Q|$ = [ links out of Q].
- Random walk over the entire web (the probability to reach it).
- Can be calculated by iterating $\pi_j^T = \pi_{j-1}^T P$
- Here P is a matrix with $p_{ij} = \frac{1}{|P_i|}$ if $P_i$ links to $P_j$ (0 otherwise)
- The pagerank vector will be $\pi^T = \lim_{j \to \infty} \pi_j^T$. “The stationary probability distribution vector”.

# Example Application - Google PageRank Calculation II

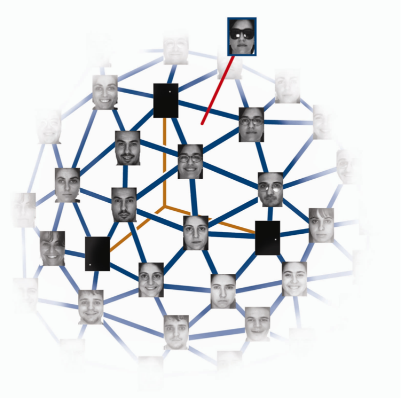
- How to change $P$ to be stochastic and irreducible (no looped chains)?
- Change the transition probability matrix to be $\tilde{P} = \alpha P + (1-\alpha)\frac{1}{n}ee^T$.
- This should run on billions of pages (Takes Google days to run it).
- What if I added a new link to my homepage?



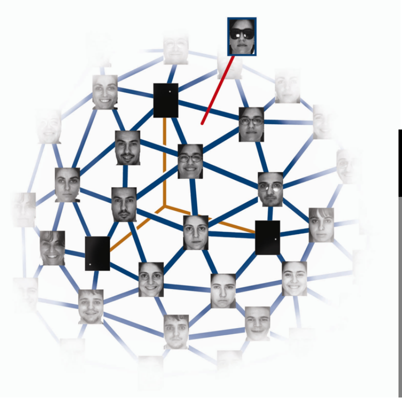
# Another Example - Face Recognition Application

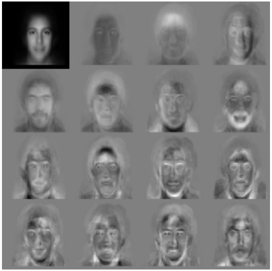
# Another Example - Face Recognition Application

# Another Example - Face Recognition Application



# Another Example - Face Recognition Application





# Eigenpair Approximation

- Using matrix perturbation theory to update the eigenpairs.
- Can update the left principal eigenvector $\pi$ of a stochastic matrix P where $\pi = \pi P$ (stationary distribution of a Markov chain).
- Such methods can accelerate algorithms like Pagerank and HIT that use the stationary distribution values as rating scores. [1] [2]
- Suitable for updating the principle eigenvector of the perturbed matrix. eigenvectors.

[1] Adaptive methods for the computation of PageRank. Kamvar, Haveliwala and Golub, 2004.

[2] Updating Markov chains with an eye on Google's PageRank. Langville and Meyer, 2006.

# Eigenpair Approximation I

$$\begin{pmatrix} \pi_1 \\ \vdots \\ \pi_n \end{pmatrix}^T = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_n \end{pmatrix}^T \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

- **Power Iteration Method**
  - iterates on $\phi_{next} = \frac{A\phi}{||A\phi||}$
  - converges to the (dominant) eigenvector of the largest eigenvalue
- **Adaptive Power Method** uses the fact that most coordinates of the eigenvector become stable within few iterations, and we can compute only ones which have not converged.

# Eigenpair Approximation II

- **Aggregated Power Iteration** reduces the unchanged states of the Markov chain into a single super state, and creates a smaller matrix. This seed eigenvector is used as the guess for each full power iteration.

# Updating A low Dimensional Representation

- We start with a symmetric matrix $A$ which is the affinity matrix of the dataset.
- That is, $[A]_{ij}$ is the similarity level between elements $i$ and $A$ can be computed in various ways using different kernels and distance metrics.
- A low dimensional embedding for the dataset is computed using the spectral decomposition of $A$.

# Updating A low Dimensional Representation - Cont.

- We are now given the perturbation matrix $\tilde{A}$ of the matrix $A$.
- We can assume that the perturbations are sufficiently small, that is $\|\tilde{A} - A\| < \varepsilon$ for some small $\varepsilon$.
- We also assume that $\tilde{A}$ is symmetric since we compute it in the same way as $A$ using the updated $\tilde{X}$.
- We wish to update the perturbed eigenpairs of $\tilde{A}$ based on $A$ and its eigenpairs.

# Updating A low Dimensional Representation - Cont.

- Given a symmetric $n \times n$ matrix $A$ with its $k$ dominant eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_k$ and eigenvectors $\phi_1, \phi_2, ..., \phi_k$, respectively, and a perturbed matrix $\tilde{A}$ such that $\|\tilde{A} - A\| < \varepsilon$, find the perturbed eigenvalues $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq ... \geq \tilde{\lambda}_k$ and its eigenvectors $\tilde{\phi}_1, \tilde{\phi}_2, ..., \tilde{\phi}_k$ of $\tilde{A}$ in the most efficient way.

# Computing the Eigenpairs First Order Approximation I

- Compute the approximation of each eigenpair.
- Given an eigenpair $(\phi_i, \lambda_i)$ of a symmetric matrix $A$ where $A\phi_i = \lambda_i\phi_i$, we compute the first order approximation of the eigenpair of the perturbed matrix $\tilde{A} = A + \Delta A$.
- We assume that the change $\Delta A$ is sufficiently small, which result in a small perturbation in $\phi_i$ and $\lambda_i$.
- We look for $\Delta\lambda_i$ and $\Delta\phi_i$ that satisfy the equation

$$(A + \Delta A)(\phi_i + \Delta\phi_i) = (\lambda_i + \Delta\lambda_i)(\phi_i + \Delta\phi_i). \tag{1}$$

# Eigenpair Approximation

## Theorem

*If $(\phi_i, \lambda_i)$ is an Eigenpair of $A$ and $\tilde{A} = A + \Delta A$ then*

$$\tilde{\lambda}_i = \lambda_i + \phi_i^T(\tilde{A} - A)\phi_i + o(\|\Delta A\|^2), \tag{2}$$

$$\tilde{\phi}_i = \phi_i + \sum_{j \neq i} \frac{\phi_j^T(\tilde{A} - A)\phi_i}{\lambda_i - \lambda_j}\phi_j + o(\|\Delta A\|^2). \tag{3}$$

# The Recursive Power Iteration (RPI) Algorithm

## Overview

- Power Iteration method has proved to be effective when calculating the principle eigenvector of a matrix.
- In the RPI algorithm the first order approximation of the eigenpairs of $A$ will be the initial guess for the power iteration method.

# The Recursive Power Iteration (RPI) Algorithm - Cont.

## Overview

- The first order approximation should be close to the actual solution we seek and therefore requires fewer iteration steps to converge.
- Once the eigenvector $\tilde{\phi}_i$ is obtained in step $i$, we transform $\tilde{A}$ into a matrix that has $\tilde{\phi}_{i+1}$ as its principle eigenvector. We iterate this step until we recover the $k$ dominant eigenvectors of $\tilde{A}$.

# The Recursive Power Iteration (RPI) Algorithm - Cont.

**Algorithm 1:** Recursive Power Iteration Algorithm

**Input**: Perturbed symmetric matrix $\tilde{A}_{n\times n}$, number of eigenvectors to calculate $k$, initial eigenvectors guesses $\{v_i\}_{i=1}^{k}$, admissible error *err*

**Output**: Approximated eigenvectors $\left\{\tilde{\phi}_i\right\}_{i=1}^{k}$, approximated eigenvalues $\left\{\tilde{\lambda}_i\right\}_{i=1}^{k}$

# The Recursive Power Iteration (RPI) Algorithm - Cont.

**for** $i = 1 \rightarrow k$ **do**

  $\phi \leftarrow v_i$

  **repeat**

    $\phi_{next} \leftarrow \frac{\tilde{A}\phi}{\|\tilde{A}\phi\|}$

    $err_\phi \leftarrow \left\|\phi - \phi_{next}\right\|$

    $\phi \leftarrow \phi_{next}$

  **until** $err_\phi \leq err$

  $\tilde{\phi}_i \leftarrow \phi$

  $\tilde{\lambda}_i \leftarrow \frac{\tilde{\phi}_i^T \tilde{A} \tilde{\phi}_i}{\tilde{\phi}_i^T \tilde{\phi}_i}$

  $\tilde{A} \leftarrow \tilde{A} - \tilde{\phi}_i \tilde{\lambda}_i \tilde{\phi}_i^T$

**end for**

# Correctness of the RPI Algorithm

- The correctness of the RPI algorithm is proved based on the fact that the power iteration method converges, and on the spectral decomposition properties of $\tilde{A}$.
- In step $i$ we find the $i$ largest eigenpair using the power method with the first order approximation as the initial guess.
- We then subtract the matrix $\tilde{\phi}_i \tilde{\lambda}_i \tilde{\phi}_i^T$ from $\tilde{A}$. This step force the next eigenpair to become the principal eigenpair which will be found on the next step.
- We use the fact that $\tilde{A}$ is symmetric and has a spectral decomposition of the form $\tilde{A} = \sum_{i=1}^{n} \tilde{\phi}_i \tilde{\lambda}_i \tilde{\phi}_i^T$, where $\tilde{\phi}_i, \tilde{\lambda}_i$ are the eigenpairs of $\tilde{A}$.

# Matrix Factorization

Determine bounds for the change in the factors of a matrix when the matrix is perturbed.

## Theorem

*Stuart, 1977 If $A = QR$ and $A + \Delta A = (Q + \Delta Q)(R + \Delta R)$ are the QR factorizations, then, for sufficiently small $\Delta A$*

$$\frac{\|\Delta R\|_F}{\|R\|_F} \leq c\kappa(A)\frac{\|\Delta A\|_F}{\|A\|_F}, \|\Delta Q\|_F \leq c\kappa(A)\frac{\|\Delta A\|_F}{\|A\|_F} \tag{4}$$

*where c is a small constant and $\kappa(A) = \|A\|\|A^{-1}\|$ is the condition number of A.*