

Outline

- Mathematical background
- PCA
- SVD
- Some PCA and SVD applications
- Case study: LSI

Mathematical Background

Variance

If we have **one** dimension:

- English: The average square of the distance from the mean of the data set to its points
- Definition: $Var(X) = E(X - E(X))^2 = E(x^2) - (E(X))^2$
- Empirical: $Var(x) = \sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$

Many datasets have more than one dimension. Example: we might have our data set both the height of all students and the mark they received, and we want to see if the height has an effect on the mark.

Mathematical Background

Covariance

Always measured between **two** dimensions.

- English: For each data item, multiply the difference between the x value and the mean of x , by the difference between the y value and the mean of y .

- Definition: $cov(X, Y) = E[(X - E(X))(Y - E(Y))] = E(X \cdot Y) - E(X) \cdot E(Y)$

- Empirical: $cov(X, Y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$

the inner product of values on the X dimension with the values on the Y dimension (after subtracting the means): $\frac{1}{N} (X^T \cdot Y)$

Mathematical Background

Covariance properties

- $cov(X,X)=Var(X)$
- $cov(X,Y)=cov(Y,X)$
- If X and Y are independent (uncorrelated) $\rightarrow cov(X,Y)=0$
- If X and Y are correlated (both dimensions increase together) $\rightarrow cov(X,Y)>0$
- If X and Y are anti-correlated (one dimension increases, the other decreases) $\rightarrow cov(X,Y)<0$

Correlation

Is a scaled version of covariance: $cor(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$

Always $-1 \leq cor(X,Y) \leq 1$

Mathematical Background

Covariance Matrix

Recall that covariance is a measure between two dimensions.

For example, if we have 3 dimensional data set (dimensions x, y, z), we should calculate $cov(x,y)$, $cov(y,z)$, and $cov(x,z)$.

For n dimensional data, we calculate $n!/(n-2)!*2 = n(n-1)/2$ different covariance values.

The covariance matrix for a set on data with n dimensions is:

$$C(n \times n) = (c[i,j]=c[j,i]=cov(Dim[i],Dim[j]))$$

The covariance matrix for a 3 dimensional data is

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

Mathematical Background

Covariance Matrix Properties

$$C = \begin{pmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{cov}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{cov}(z, z) \end{pmatrix}$$

- C is square and symmetric matrix.
- The diagonal values are the variance for each dimension and the off-diagonal are the covariance between measurement types.
- Large term in the diagonal correspond to interesting dimensions, whereas large values in the off-diagonal correspond to high correlations (redundancy).

Because we want to minimize the correlation (redundancy) and maximize the variance, we would like to have a diagonal covariance matrix.

Mathematical Background

eigenvectors

Example:
$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \times \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

if we thought of the squared matrix as a transformation matrix, then multiply it with the Eigenvector don't change its direction.

eigenvalues and eigenvectors always come in pairs. In the example: 4 is the eigenvalue of our eigenvector.

No matter what multiple of the eigenvector we took, we get the same eigenvalue.

Example:
$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \times \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

Mathematical Background

eigenvectors

The length of a vector doesn't affect whether it's an eigenvector or not, whereas the direction does. So to keep eigenvectors standard, we scale them to have length 1.

So we scale our vector: $\begin{pmatrix} 3 \\ 2 \end{pmatrix} \div \sqrt{13} = \begin{pmatrix} 3/\sqrt{13} \\ 2/\sqrt{13} \end{pmatrix}$

eigenvectors properties

1. eigenvectors can only be found for **square** matrices.
2. Not every square matrix has eigenvectors.

Mathematical Background

eigenvectors properties (continue)

3. If we scale the eigenvector by an amount, we will still get the same eigenvalue (*as we saw*).
4. **Symmetric** matrices $S(n \times n)$ also satisfies two properties:
 - I. has exactly n eigenvectors.
 - II. All the eigenvectors are orthogonal (perpendicular). This is important because we can express the data in term of eigenvectors, instead of expressing them in the original space.

We say that eigenvectors are *orthonormal*, which means orthogonal and has length 1.

What are the eigenvectors of the identity matrix?

Any vector is an eigenvector to the identity matrix.

Outline

- Mathematical background
- PCA
- SVD
- Some PCA and SVD applications
- Case study: LSI

PCA and SVD

PCA: Principle Components Analysis, also known as KLT (Karhunen-Loeve Transform).

SVD: Singular Value Decomposition.

SVD and PCA are closely related.

Why we use SVD and PCA?

- A powerful tool for analyzing data and finding patterns.
- Used for compression. So you can reduce the number of dimensions without much loss of information.

PCA

Objective: project the data onto a lower dimensional linear space such that the variance of the projected data is maximized.

Equivalently, it is the linear projection that minimizes the average projection cost (mean squared distance between the data points and their projections).

Different from the feature subset selection !!!

Problem to solve: In high dimensional space, we need to learn a large number of parameters. Thus if the dataset is small, this will result in large variance and over-fitting.

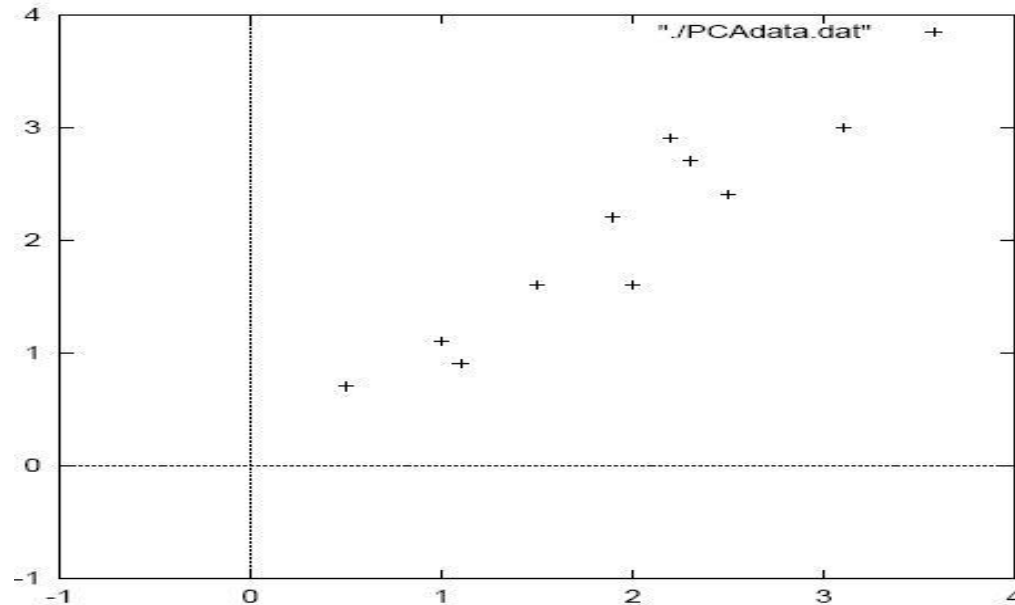
We want to represent the vector x in a different space (p dimensional) using a set of orthonormal vectors U (where u_i is a principle component).

PCA

Method to perform PCA on a data

➤ Step 1: get some data

Let $A (N,n)$ be the data matrix: N is the number of data points, n is the number of dimensions. It could represent N patients with n numerical symptoms each (blood pressure, cholesterol level etc) or N documents with n terms in each document (used in IR).



PCA

Method to perform PCA on a data

➤ Step 2: Subtract the mean

Intuitively, we translate the origin to the center of gravity. We obtain the matrix B (N,n):

$$B = [b_{i,j}] = [a_{i,j} - \overline{a_{*,j}}].$$

This produces a zero mean data (the column averages are zero).

➤ Step 3: Calculate the covariance matrix C .

$$C(n,n) = \frac{1}{N} [B^T(n,N) \times B(N,n)]: T \text{ is transpose.}$$

Since our data is 2D, the covariance matrix will be (2 x 2).

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

Notice that the non-diagonal elements are positive, why?

PCA

Method to perform PCA on a data

- Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix

Since the covariance matrix is square, we can calculate the eigenvectors

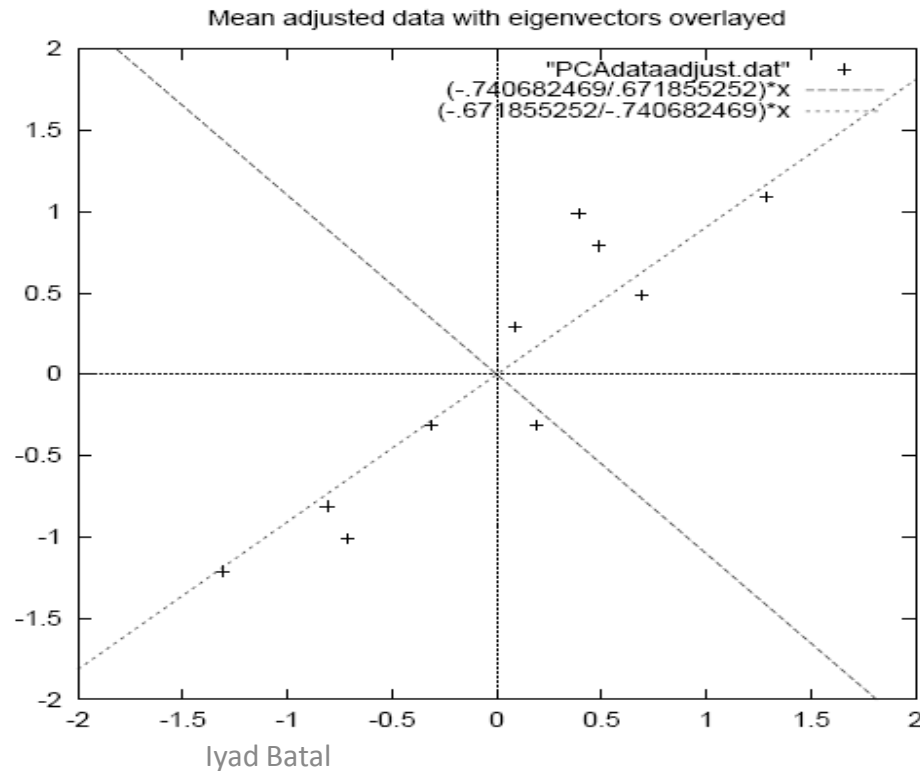
$$\lambda_1 \approx 1.28, V_1 \approx [-0.677 \ -0.735]^T, \lambda_2 \approx 0.49, V_2 \approx [-0.735 \ 0.677]^T$$

Notice that V_1 and V_2
Are orthonormal, why?

$$V_1 \cdot V_2 = 0$$

$$|V_1| = 1$$

$$|V_2| = 1$$



PCA

Method to perform PCA on a data

➤ Step 5: Choosing components and forming a feature vector

Objective: project the n dimensional data on a p dimensional subspace ($p \leq n$), minimizing the error of the projections (sum of squared difference).

Here is where we reduce the dimensionality of the data (for example, to do compression).

How: Order the eigenvalues from highest to lowest to get the components in order of significance. Project on the p eigenvectors that corresponds to the highest p eigenvalues.

PCA

Method to perform PCA on a data

➤ Step 5 (continue)

The eigenvector with the highest eigenvalue is the *principle component* of the data.

if we are allowed to pick only one dimension to project the data on it, then the principle component is the best direction.

the PC of our example is :

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

PCA

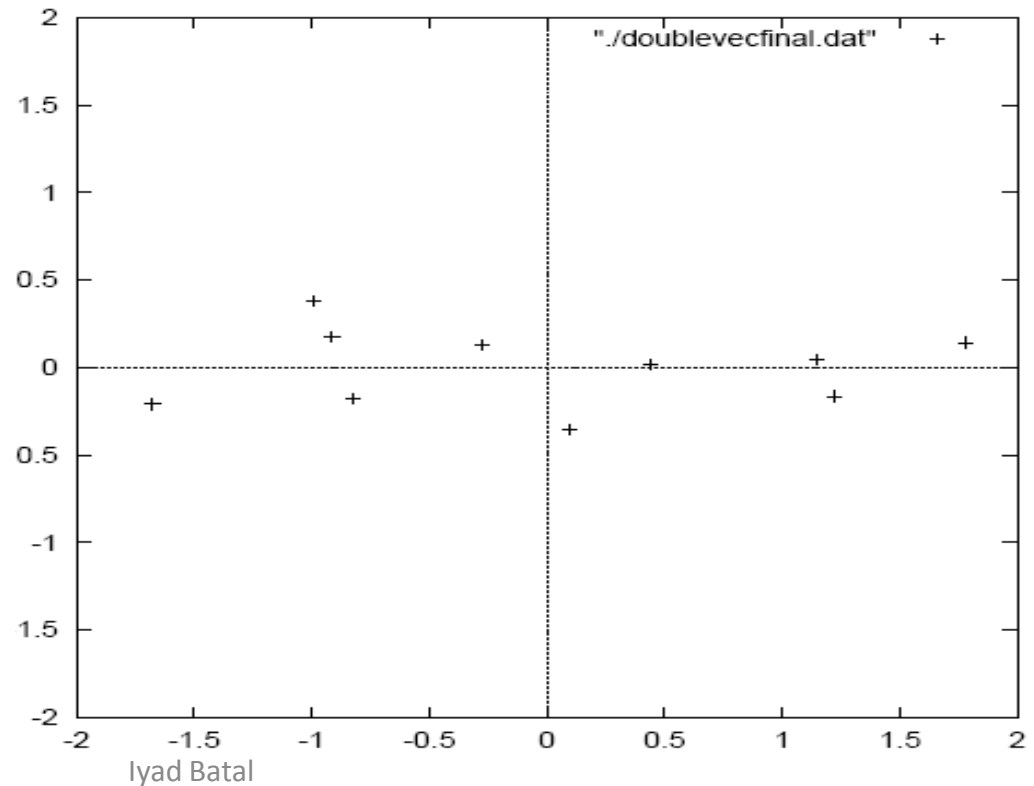
Method to perform PCA on a data

➤ Step 6: Derive the new data set

Let's denote Feature space matrix by $U(n \times p)$: where the columns are the eigenvectors. Let the final data be F .

$$F(N,p)=B(N,n) \times U(n,p)$$

The final data F resides in a p -dimensional feature space.

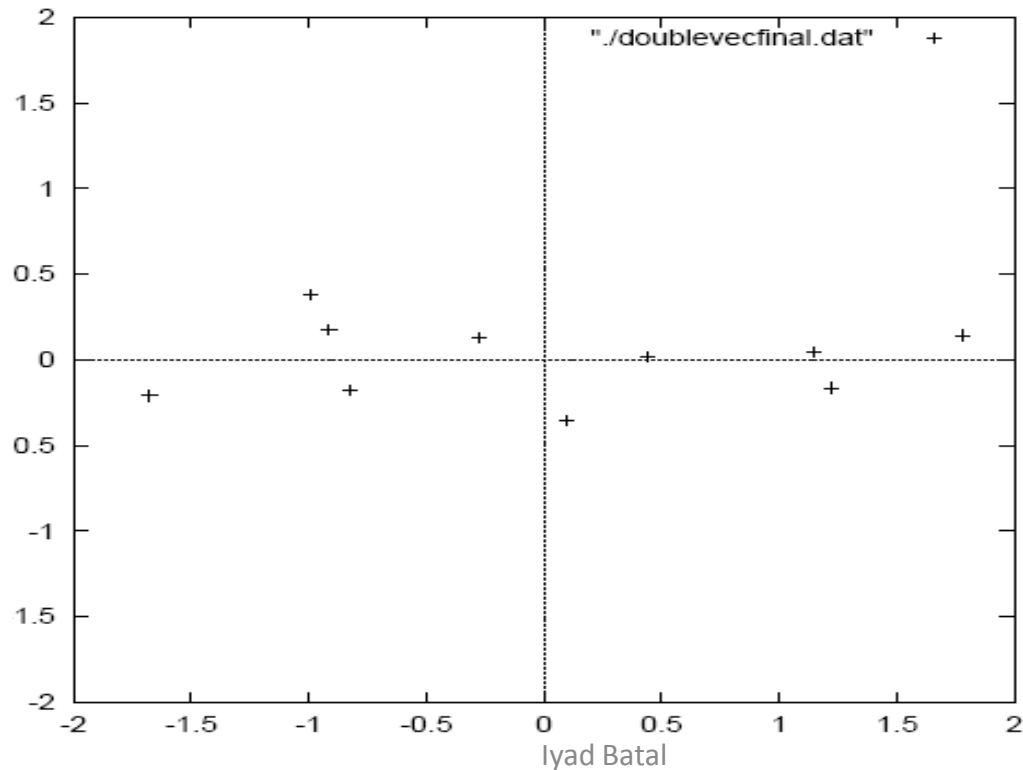


PCA

Method to perform PCA on a data

➤ Step 6: (continue)

For our example, if we keep both eigenvectors, we get the original data, rotated so that The eigenvectors are the axes. (we've lost no information)



PCA

Method to perform PCA on a data

- Step 7 (optional): getting the old data back. (if we are doing compression.)
If we keep all eigenvectors, we will get exactly the same data.

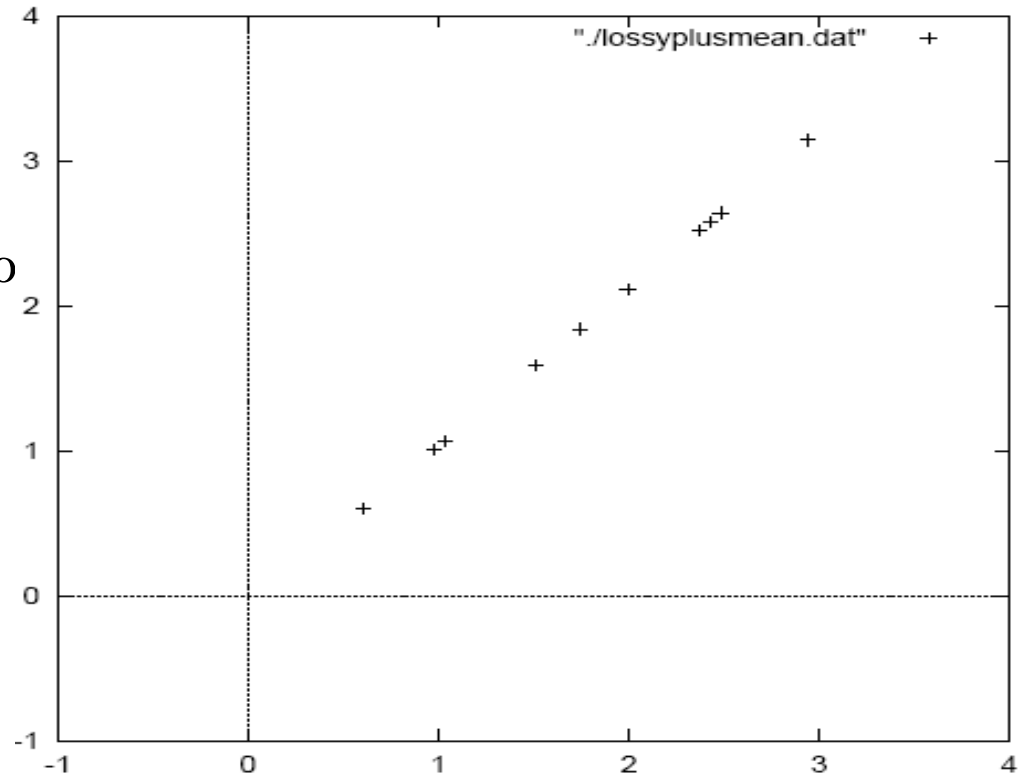
We know $F=B \times U$

$$\rightarrow B = F \times U^{-1}$$

$$\rightarrow B = F \times U^T : (\textit{proof later})$$

To get A, we add the mean to
vector to B.

Notice: The variance along the
Other Component has
gone (a lossy compression)



PCA and FLD

FLD: Fisher's Linear Discriminant.

Is a supervised learning method (utilizes the class label) that is used to select the projection that maximizes the class separation.

Specifically: we have $y = w^T x$, FLD tries to adjust the components of w such that it maximizes the distance between the projected means, while minimizing the variance within each class.

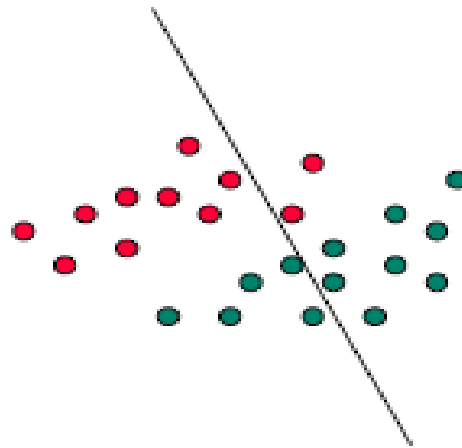
$J(W) = \frac{m_2 - m_1}{s_1^2 + s_2^2}$ where m_1 and m_2 are the projected means and

$$s_k^2 = \sum_{i \in C_k} (y_i - m_k)^2$$

PCA and FLD

PCA is unsupervised learning technique that finds the dimensions useful to represent the data, but maybe bad for discrimination between different classes.

Example: Project 2D data into 1D. We have two classes: red and green.



What direction will PCA choose?

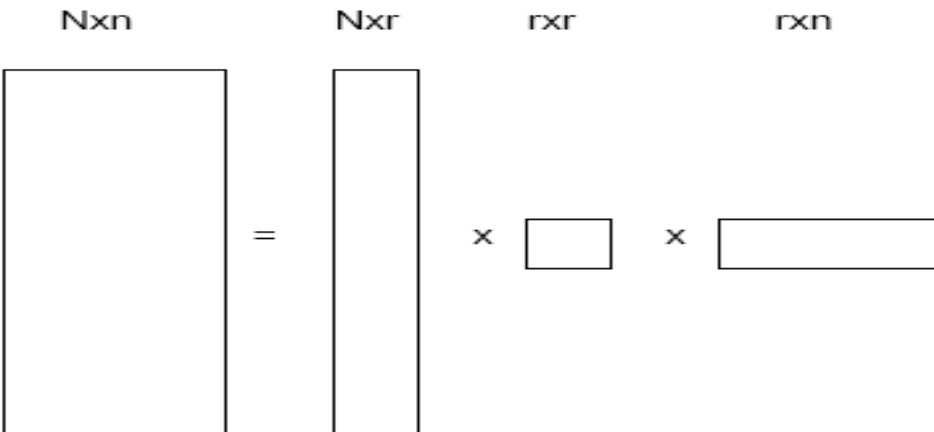
Outline

- Mathematical background
- PCA
- SVD
- Some PCA and SVD applications
- Case study: LSI

SVD

The eigenvalues and eigenvectors are defined for squared matrices. For rectangular matrices, a closely related concept is Singular Value Decomposition (SVD).

Theorem: Given an $N \times n$ real matrix A , we can express it as:

$$\mathbf{A} = \mathbf{U} \times \mathbf{\Lambda} \times \mathbf{V}^T$$


The diagram shows the equation $\mathbf{A} = \mathbf{U} \times \mathbf{\Lambda} \times \mathbf{V}^T$ with corresponding matrix dimensions: $N \times n$ for \mathbf{A} , $N \times r$ for \mathbf{U} , $r \times r$ for $\mathbf{\Lambda}$, and $r \times n$ for \mathbf{V}^T . The matrices are represented by rectangles of varying sizes and orientations to indicate their dimensions.

where U is a column-orthonormal $N \times r$ matrix, r is the rank of the matrix A (number of linearly independent rows or columns), Λ is a diagonal $r \times r$ matrix where the elements are sorted in descending order, and V is a column-orthonormal $n \times r$ matrix.

SVD decomposition for a matrix is unique.

SVD

The values of the diagonal Λ are called singular values. (we will see later that they correspond to the square root of the eigenvalues of the covariance matrix).

Theorem: the inverse of an orthonormal matrix is its transpose.

Proof: we know that $(A^T A)_{ij} = a_i^T a_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$

Therefore $A^T \times A = I$ where I is the identity matrix.

From the definition of A^{-1} : $A^{-1} \times A = I$

$$\rightarrow A^{-1} = A^T$$

$$U^T \times U = I \text{ and } V^T \times V = I$$

A also can be written using spectral decomposition as:

$$A = \lambda_1 U_1 V_1^T + \lambda_2 U_2 V_2^T + \dots + \lambda_r U_r V_r^T$$

SVD

Theorem: if S is a real and symmetric ($S=S^T$) matrix then

$\mathbf{S} = \mathbf{U} \mathbf{x} \mathbf{\Lambda} \mathbf{x} \mathbf{U}^T$. Where the columns of U are the eigenvectors, and Λ is a diagonal matrix with values corresponding to eigenvalues.

Proof: let U be the matrix of eigenvectors placed in the columns:

$$U=[u_1 \ u_2 \ \dots \ u_n]$$

We can write: $\mathbf{S} \mathbf{x} \mathbf{U}=\mathbf{U} \mathbf{x} \mathbf{\Lambda}$

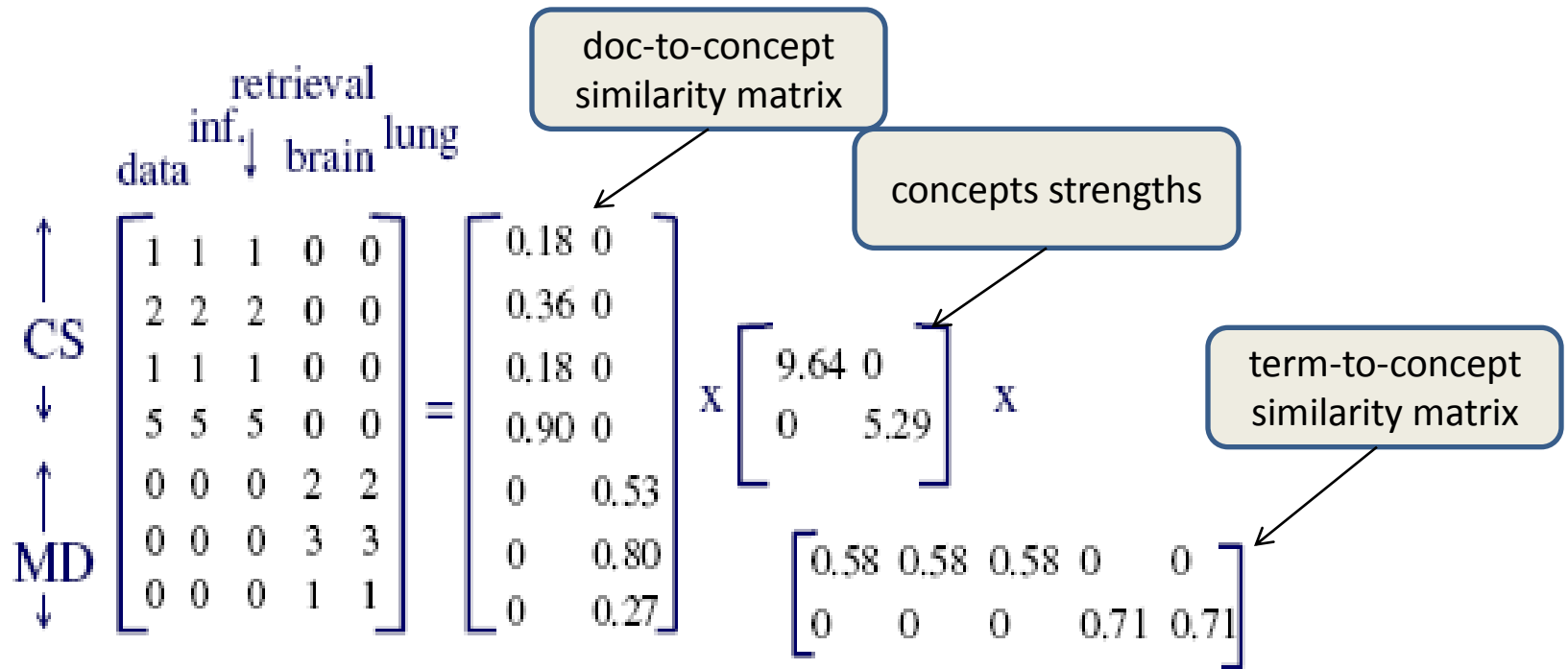
$[\mathbf{S} \mathbf{x} u_1 \ \mathbf{S} \mathbf{x} u_2 \ \dots \ \mathbf{S} \mathbf{x} u_n]=[\lambda_1 \cdot u_1 \ \lambda_2 \cdot u_2 \ \dots \ \lambda_n \cdot u_n]$ which is the definition of the eigenvectors.

Therefore: $\mathbf{S}=\mathbf{U} \mathbf{x} \mathbf{\Lambda} \mathbf{x} \mathbf{U}^{-1}$

Because U is orthonormal $\rightarrow U^{-1} = U^T$

$$\rightarrow \mathbf{S} = \mathbf{U} \mathbf{x} \mathbf{\Lambda} \mathbf{x} \mathbf{U}^T$$

SVD

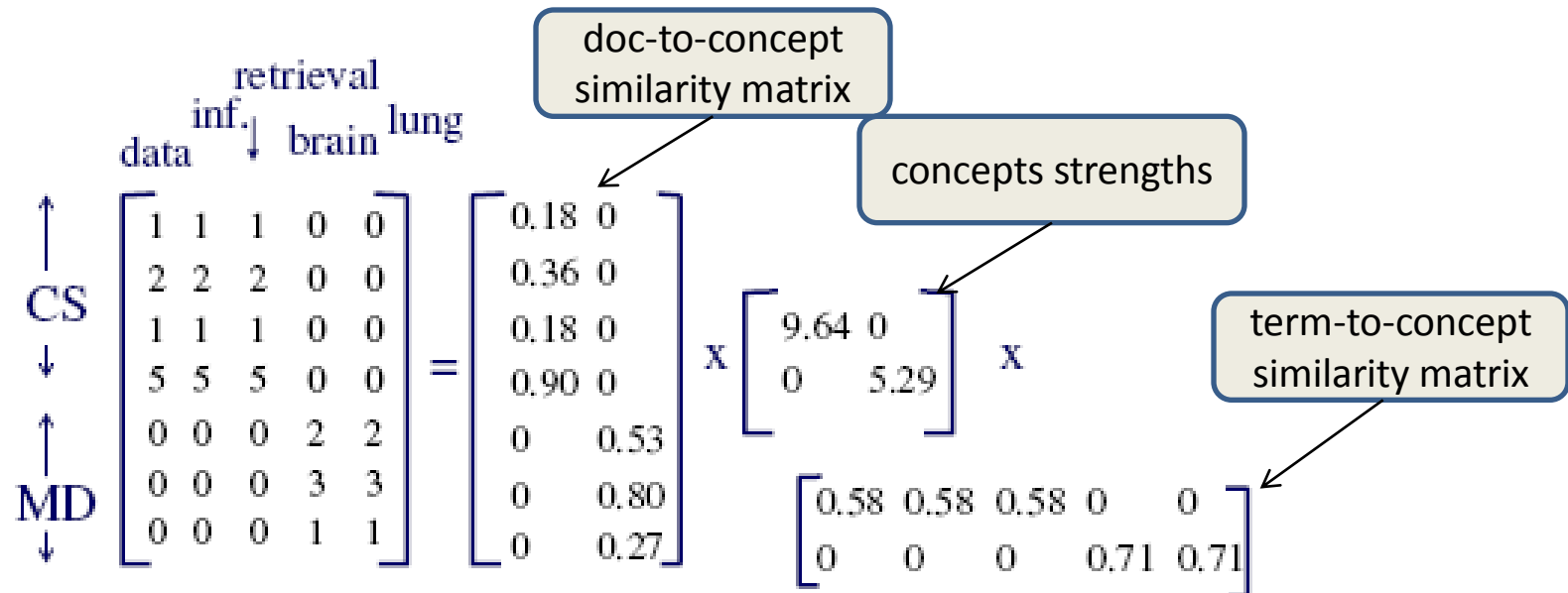


Example: the matrix contains 7 documents with the corresponding frequencies of each term.

In real IR applications, we take into considerations the normalized TF and IDF when calculating term weights.

The rank of this matrix $r=2$ because we have 2 types of documents (CS and Medical documents), i.e. 2 concepts.

SVD



U can be thought as the document-to-concept similarity matrix, while V is the term-to-concept similarity matrix.

For example, $U_{1,1}$ is the weight of CS concept in document d_1 , λ_1 is the strength of the CS concept, $V_{1,1}$ is the weight of the first term 'data' in the CS concept, $V_{2,1}=0$ means 'data' has zero similarity with the 2nd concept (Medical).

What does $U_{4,1}$ means?

SVD

The $N \times N$ matrix $D = A \times A^T$ will intuitively give the document-to-document similarities.

$$D = A \times A^T = \begin{bmatrix} 3 & 6 & 3 & 15 & 0 & 0 & 0 \\ 6 & 12 & 6 & 30 & 0 & 0 & 0 \\ 3 & 6 & 3 & 15 & 0 & 0 & 0 \\ 15 & 30 & 15 & 75 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 12 & 4 \\ 0 & 0 & 0 & 0 & 12 & 18 & 6 \\ 0 & 0 & 0 & 0 & 4 & 6 & 2 \end{bmatrix}$$

The eigenvectors of the D matrix will be the columns of the U matrix of the SVD of A . Proof?

$$\begin{aligned} A \times A^T &= U \times \Lambda \times V^T \times V \times \Lambda \times U^T \quad (\text{remember that } (A \times B)^T = B^T \times A^T) \\ &= U \times \Lambda \times \Lambda \times U^T \quad : V^T \times V = I \text{ because } V \text{ is orthonormal} \\ &= U \times \Lambda^2 \times U^T \quad : \text{because } \Lambda \text{ is a diagonal matrix.} \end{aligned}$$

Notice that D is symmetric because $(U \times \Lambda^2 \times U^T)^T = U \times \Lambda^2 \times U^T$

$$\text{or simply because } (A \times A^T)^T = A^{TT} \times A^T = A \times A^T$$

Because D is symmetric, it can be written as $D = S = U \times \Lambda \times U^T$ where U are the D 's eigenvectors and Λ are D 's eigenvalues.

SVD

Symmetrically, the $n \times n$ matrix $T = A^T \times A$ will give the term-to-term similarities (the covariance matrix).

$$T = A^t \times A \begin{bmatrix} 31 & 31 & 31 & 0 & 0 \\ 31 & 31 & 31 & 0 & 0 \\ 31 & 31 & 31 & 0 & 0 \\ 0 & 0 & 0 & 14 & 14 \\ 0 & 0 & 0 & 14 & 14 \end{bmatrix}$$

The eigenvectors of the T matrix are the columns of the V matrix of the SVD of A .

T is a symmetric matrix because $(A^T \times A)^T = A^T \times A^{TT} = A^T \times A$

$$A^T \times A = V \times \Lambda \times U^T \times U \times \Lambda \times V^T = V \times \Lambda \times \Lambda \times V^T = V \times \Lambda^2 \times V^T$$

Notice that both D and T have the same eigenvalues, which are the squares of the λ_i elements (The singular values of A).

These observations shows the close relation between SVD and PCA , which uses the eigenvectors of the covariance matrix.

SVD

Very important property: (we will see it again in Kleinberg algorithm)

$(A^T x A)^k x v` \approx (\text{constant}) v_1$ where $k \gg 1$, $v`$ is a random vector, v_1 is the first right singular vector of A , or equivalently is the eigenvector of $A^T x A$ (as we already proved). *Proof?*

$$\begin{aligned}(A^T x A)^k &= (A^T x A) x (A^T x A) x \dots = (V x \Lambda^2 x V^T) x (V x \Lambda^2 x V^T) x \dots \\ &= (V x \Lambda^2 x V^T) x \dots = (V x \Lambda^4 x V^T) x \dots = (V x \Lambda^{2k} x V^T)\end{aligned}$$

Using spectral decomposition:

$$(A^T x A)^k = (V x \Lambda^{2k} x V^T) = \lambda_1^{2k} v_1 v_1^T + \lambda_2^{2k} v_2 v_2^T + \dots + \lambda_n^{2k} v_n v_n^T$$

Because $\lambda_1 > \lambda_{i \neq 1} \rightarrow \lambda_1^{2k} \gg \lambda_{i \neq 1}^{2k}$

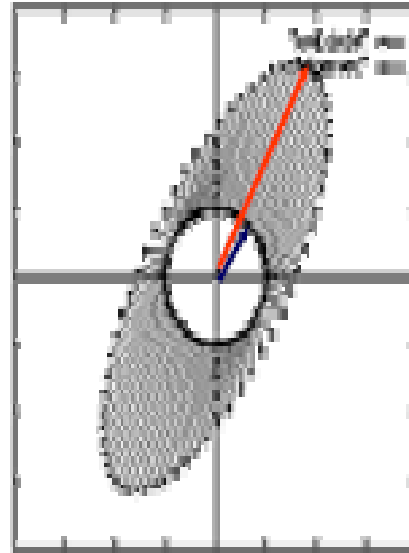
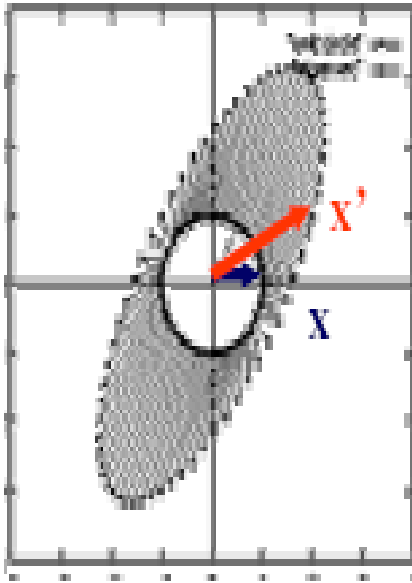
$$\text{Thus } (A^T x A)^k \approx \lambda_1^{2k} v_1 v_1^T$$

$$\text{Now } (A^T x A)^k x v` = \lambda_1^{2k} v_1 v_1^T x v` = (\text{const}) v_1$$

because $v_1^T x v`$ is a scalar.

SVD

Geometrically, this means that if we multiple any vector with matrix $(A^T x A)^k$, then result is a vector that is parallel to the first eigenvector.



PCA and SVD

Summary for PCA and SVD

Objective: find the principal components P of a data matrix A(n,m).

1. First zero mean the columns of A (translate the origin to the center of gravity).
2. Apply PCA or SVD to find the principle components (P) of A.

PCA:

- I. Calculate the covariance matrix $C = \frac{A A^T}{n}$
- II. p = the eigenvectors of C.
- III. The variances in each new dimension is given by the eigenvalues.

SVD:

- I. Calculate the SVD of A.
 - II. $P = V$: the right singular vectors.
 - III. The variances are given by the squaring the singular values.
3. Project the data onto the feature space. $F = P \times A$
 4. Optional: Reconstruct A' from Y where A' is the compressed version of A.

Outline

- Mathematical background
- PCA
- SVD
- **Some PCA and SVD applications**
- **Case study: LSI**

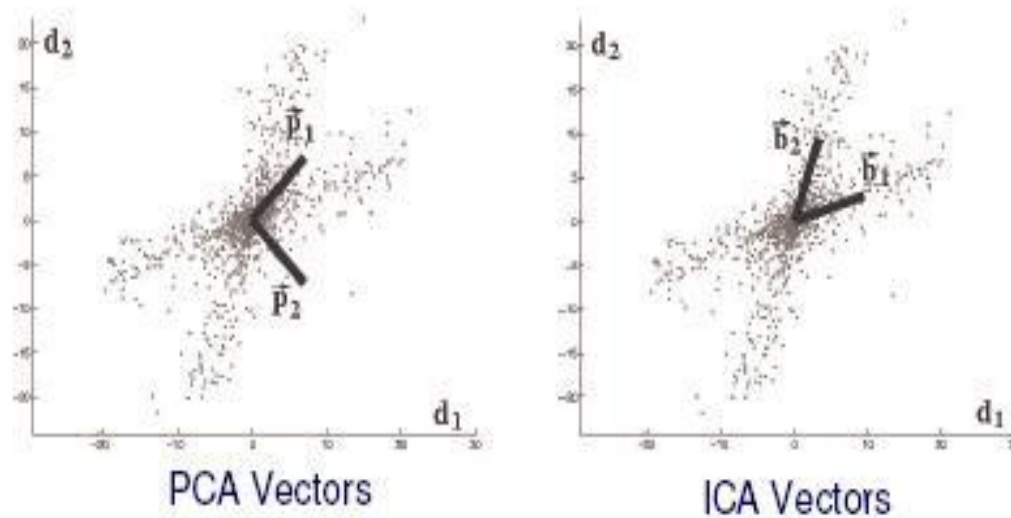
SVD and PCA applications

- LSI: Latent Semantic Indexing.
- Solve over-specified (no solution: least squares error solution) and under-specified (infinite number of solutions: shortest length solution) linear equations.
- Ratio rules (computer quantifiable association rules like *bread:milk:buffer=2:4:3*).
- Google/PageRank algorithm (random walk with restart).
- Kleinberg/Hits algorithm (compute hubs and authority scores for nodes).
- Query feedbacks (learn to estimate the selectivity of the queries: a regression problem).
- Image compression (*other methods: DCT used in JPEG, and wavelet compression*)
- Data visualization (by projecting the data on 2D).

Variations: ICA

ICA (Independent Components Analysis)

Relaxes the constraint of orthogonality but keeps the linearity. Thus, could be more flexible than PCA in finding patterns.



$$X (N,n)=H (N,n) \times B (n,n)$$

where X is the data set, H are hidden variables, and B are basis

vectors. $h_{i,j}$ can be understood as the weight of b_j in the instance X_i

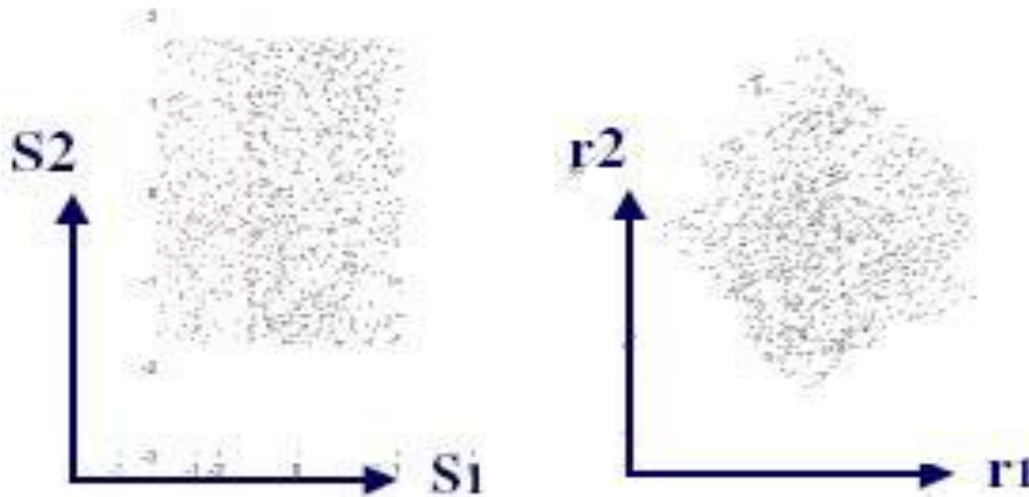
Variations: ICA

Linearity: $X_i = h_{i,1} b_1 + h_{i,2} b_2$

Problem definition: Knowing X, find H and B.

Make hidden variables h_i mutually independent:

$$p(h_i, h_j) = p(h_i) * P(h_j)$$



Which figure satisfies data independency?

Outline

- Mathematical background
- PCA
- SVD
- Some PCA and SVD applications
- **Case study: LSI**

SVD and LSI

LSI: Latent Semantic Indexing.

Idea: try to group similar terms together, to form a few concepts, then map the documents into vectors in the concept-space, as opposed to vectors in the n -dimensional space, where n is the vocabulary size of the document collection.

This approach automatically groups terms that occur together into concepts. Then every time the user asks for a term, the system determines the relevant concepts and search for them.

In order to map documents or queries into the concept space, we need the term-to-concept similarity matrix V .

SVD and LSI

Example: find the documents containing the term 'data'.

$$\vec{q} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

To translate q to a vector q_c in the concept space:

$$q_c = V^T \times q = \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.58 \\ 0 \end{bmatrix}$$

It means that q is related to the CS group of terms (with strength=0.58), and unrelated to the medical group of terms.

SVD and LSI

More importantly, q_c now involves the terms ‘information’ and ‘retrieval’,

$$\vec{q}_c = \begin{bmatrix} 0.58 \\ 0 \end{bmatrix}$$

thus LSI system may return documents that do not necessarily contain the term ‘data’

For example, a document d with a single word ‘retrieval’ $\vec{d} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

d will be mapped into the concept space $\vec{d}_c = V^t \times \vec{d} = \begin{bmatrix} 0.58 \\ 0 \end{bmatrix} = \vec{q}_c$
And will be a perfect match for the query.

Cosine similarity is one way to measure the similarity between the query and the documents.

Experiments showed that LSI outperforms standard vector methods with improvement of as much as 30% in terms of precision and recall.

Thank you for listening

Questions or Thoughts??