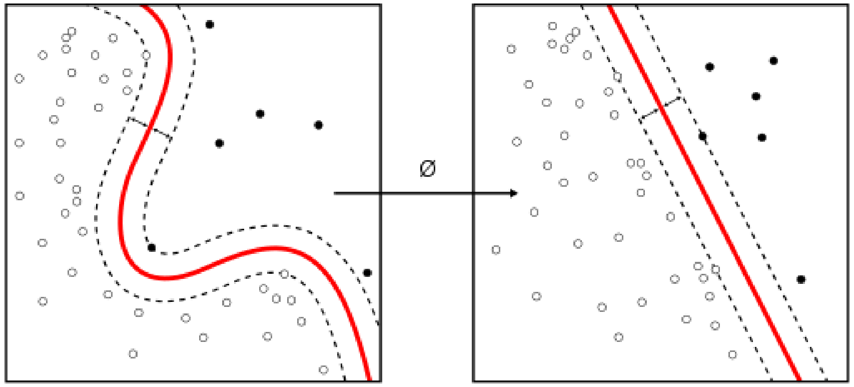
Best-fit subspaces and Singular Value Decomposition

longhuan@sjtu.edu.cn



Mathematical background

Singular vector decomposition

Principle component analysis

Eigenvalues & Eigenvectors

- **Eigenvectors** (for a square $m \times m$ matrix $\mathbf{S}$)

$$\boxed{\mathbf{S}\mathbf{v} = \lambda \mathbf{v}}$$

(right) eigenvector   eigenvalue

$$\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0} \qquad \lambda \in \mathbb{R}$$

Example

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\mathbf{S}\mathbf{v} = \lambda \mathbf{v} \iff (\mathbf{S} - \lambda \mathbf{I})\,\mathbf{v} = \mathbf{0}$$

- How many eigenvalues are there at most?

only has a non-zero solution if $|\mathbf{S} - \lambda \mathbf{I}| = 0$

this is a m-th order equation in λ which can have **at most m distinct solutions** (roots of the characteristic polynomial) – <u>can be complex even though S is real.</u>

Matrix-vector multiplication

$$S = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

has eigenvalues 3, 2, 0 with corresponding eigenvectors

$$v_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad v_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

On each eigenvector, S acts as a multiple of the identity matrix: but as a different multiple on each.

Any vector (say $x = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix}$) can be viewed as a combination of the eigenvectors: $x = 2v_1 + 4v_2 + 6v_3$

Matrix vector multiplication

- Thus a matrix-vector multiplication such as Sx (S, x as in the previous slide) can be rewritten in terms of the eigenvalues/vectors:

$$Sx = S(2v_1 + 4v_2 + 6v_3)$$

$$Sx = 2Sv_1 + 4Sv_2 + 6Sv_3 = 2\lambda_1 v_1 + 4\lambda_2 v_2 + 6\lambda_3 v_3$$

- Even though x is an arbitrary vector, the action of S on x is determined by the eigenvalues/vectors.
- **Suggestion:** the effect of "small" eigenvalues is small.

Eigenvalues & Eigenvectors

For symmetric matrices, eigenvectors for distinct eigenvalues are **orthogonal**

$$Sv_{\{1,2\}} = \lambda_{\{1,2\}} v_{\{1,2\}}, \text{ and } \lambda_1 \neq \lambda_2 \Rightarrow v_1 \bullet v_2 = 0$$

All eigenvalues of a real symmetric matrix are **real**.

$$\text{for complex } \lambda, \text{ if } |S - \lambda I| = 0 \text{ and } \mathrm{S} = \mathrm{S}^{\mathrm{T}} \Rightarrow \lambda \in \Re$$

All eigenvalues of a positive semidefinite matrix are **non-negative**

$$\overbrace{\forall w \in \Re^n, w^T S w \geq 0}, \text{ then if } \mathrm{Sv} = \lambda \mathrm{v} \Rightarrow \lambda \geq 0$$

Example

- Let $S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \longleftarrow \boxed{\text{Real, symmetric.}}$

- Then
$$S - \lambda I = \begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} \Rightarrow (2-\lambda)^2 - 1 = 0.$$

- The eigenvalues are 1 and 3 (nonnegative, real).
- The eigenvectors are orthogonal (and real):

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Plug in these values and solve for eigenvectors.

Eigen/diagonal Decomposition

- Let $\mathbf{S} \in \mathbb{R}^{m \times m}$ be a **square** matrix with ***m*** **linearly independent eigenvectors** (a "non-defective" matrix)
- **Theorem**: Exists an **eigen decomposition**

$$\mathbf{S} = \mathbf{U} \underset{}{\overset{\text{diagonal}}{\downarrow}}\!\!\mathbf{\Lambda} \mathbf{U}^{-1}$$

- Columns of ***U*** are **eigenvectors** of ***S***
- Diagonal elements of $\mathbf{\Lambda}$ are **eigenvalues** $\mathbf{S}$ of

$$\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

Diagonal decomposition: why/how

Let ***U*** have the eigenvectors as columns: $U = \begin{bmatrix} \\ v_1 & \ldots & v_n \\ \\ \end{bmatrix}$

Then, ***SU*** can be written

$$SU = S\begin{bmatrix} \\ v_1 & \ldots & v_n \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \lambda_1 v_1 & \ldots & \lambda_n v_n \\ \\ \end{bmatrix} = \begin{bmatrix} \\ v_1 & \ldots & v_n \\ \\ \end{bmatrix}\begin{bmatrix} \lambda_1 & & \\ & \ldots & \\ & & \lambda_n \end{bmatrix}$$

Thus $\boldsymbol{SU=U\Lambda}$, or $\boldsymbol{U^{-1}SU=\Lambda}$

And $\boldsymbol{S=U\Lambda U^{-1}}$.

Diagonal decomposition - example

Recall $S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \qquad \lambda_1 = 1, \lambda_2 = 3.$

The eigenvectors $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ form $U = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$

Inverting, we have $U^{-1} = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$

Recall UU^{-1} =I.

Then, ***S=UΛU***$^{-1}$ **=** $\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$

Example continued

Let's divide ***U*** (and multiply $\boldsymbol{U^{-1}}$) by $\sqrt{2}$

Then, ***S*=** $$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Q $\qquad$ Λ $\qquad$ ***(Q^{-1}= Q^T)***

Why? Stay tuned …

Symmetric Eigen Decomposition

- If $\mathbf{S} \in \mathbb{R}^{m \times m}$ is a **symmetric** matrix:
- **Theorem**: Exists a (unique) **eigen decomposition** $S = Q\Lambda Q^T$
- where ***Q*** is **orthogonal:**
  - $\boldsymbol{Q^{-1} = Q^T}$
  - Columns of ***Q*** are normalized eigenvectors
  - Columns are orthogonal.
  - (everything is real)

Mathematical background

Singular vector decomposition

Principle component analysis

Singular Value Decomposition

For an $m \times n$ matrix $\boldsymbol{A}$ of rank r there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$A = \underset{m\times m}{U}\,\underset{m\times n}{\Sigma}\,\underset{V \text{ is } n\times n}{V^T}$$

The columns of ***U*** are orthogonal eigenvectors of $\boldsymbol{AA^T}$.

The columns of ***V*** are orthogonal eigenvectors of $\boldsymbol{A^TA}$.

Eigenvalues $\lambda_1 \ldots \lambda_r$ of $\boldsymbol{AA^T}$ are the eigenvalues of $\boldsymbol{A^TA}$.

$$\sigma_i = \sqrt{\lambda_i}$$

$$\Sigma = diag(\sigma_1 \ldots \sigma_r)$$

Singular values.

AA^T , A^TA

Rank(***A***)=Rank(***AA^T***)=Rank(***A^TA***).

Eigenvalues $\lambda_1 \ldots \lambda_r$ of ***AA^T*** are the eigenvalues of ***A^TA***.

Lemma. $\lambda \neq 0$ be the eigenvalue of AA^T, $\alpha_1, \ldots, \alpha_r$ are pairwise orthogonal unit eigenvectors corresponds to λ, then $A^T\alpha_1, A^T\alpha_2, \ldots, A^T\alpha_r$ are pairwise orthogonal eigenvectors of A^TA, and $\sqrt{(A^T\alpha_j, A^T\alpha_j)} = \sqrt{\lambda_j}$.

Theorem. $\mathrm{A} = [\mathrm{a_{ij}}]_{\mathrm{m\times n}}$ and $\sqrt{\lambda}_1, \sqrt{\lambda}_2, \ldots,$ $\sqrt{\lambda}_r$ be the singular value of A, then $\mathrm{A} = \mathrm{U}\Sigma V^T$,

where $UU^T = I, VV^T = I,\ \Sigma = \begin{bmatrix} \sqrt{\lambda}_1 & & & & \\ & \sqrt{\lambda}_2 & & \cdots & \\ & & \sqrt{\lambda}_r & & \\ & \vdots & & \ddots & \vdots \\ & & & \cdots & \end{bmatrix}$.

Singular Value Decomposition

- Illustration of SVD dimensions and sparseness

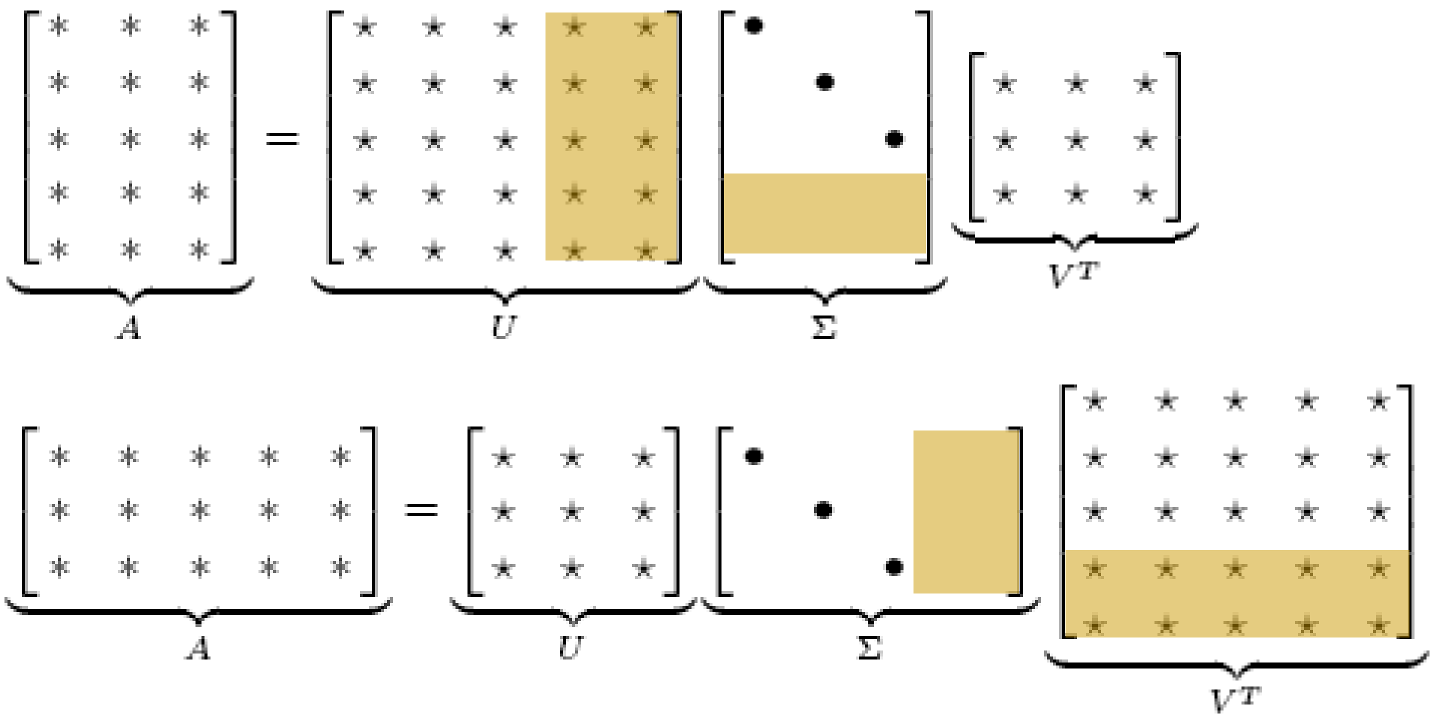


Image of unit ball under linear transformation

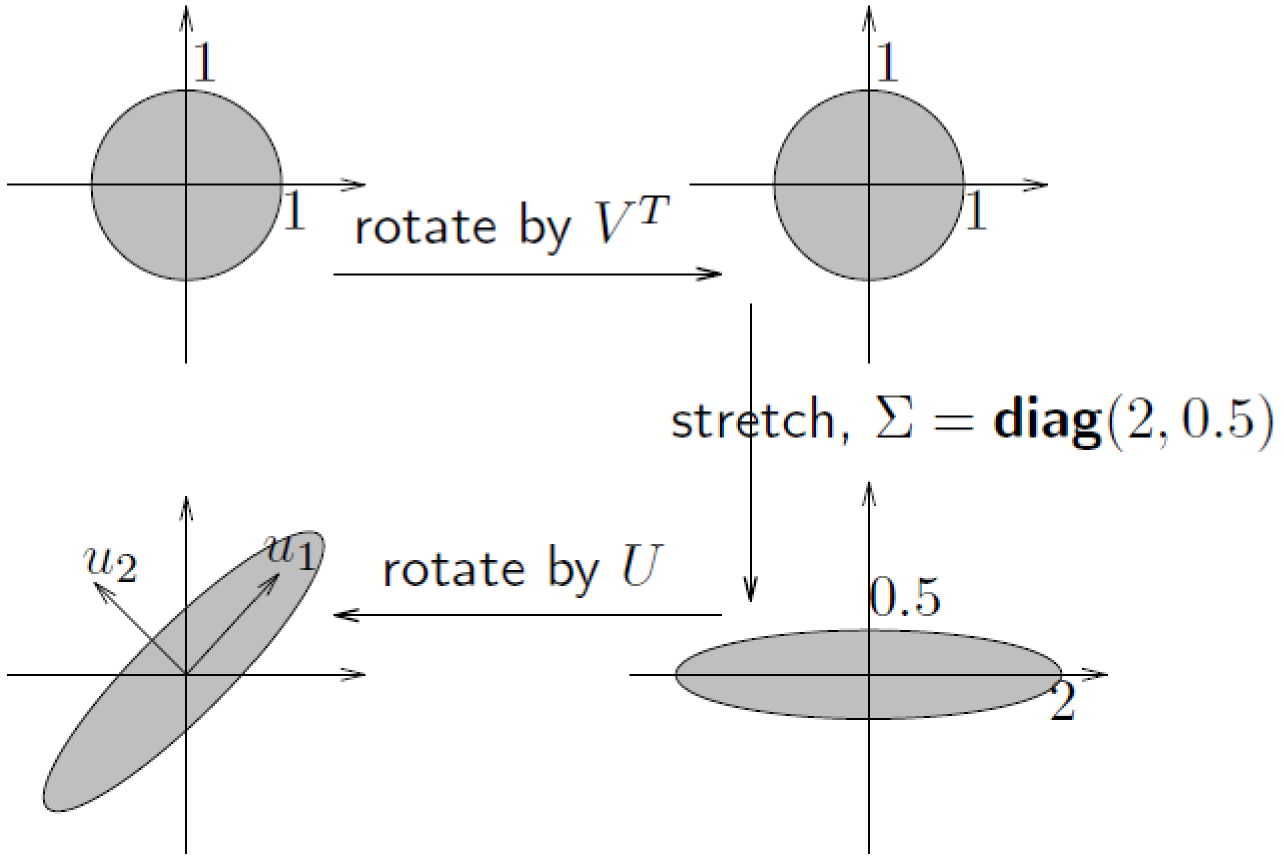
full SVD:

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

gives intepretation of $y = Ax$:

- rotate (by V^T)
- stretch along axes by σ_i ($\sigma_i = 0$ for $i > r$)
- zero-pad (if $m > n$) or truncate (if $m < n$) to get m-vector
- rotate (by U)

Image of unit ball under A



$\{Ax \mid \|x\| \leq 1\}$ is *ellipsoid* with principal axes $\sigma_i u_i$.

SVD example

Let $$A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Thus *m*=3, *n*=2. Its SVD is

$$\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Typically, the singular values arranged in decreasing order.

Low-rank Approximation

- SVD can be used to compute optimal **low-rank approximations**.
- Approximation problem: Find ***A*_*k*** of rank ***k*** such that

$$A_k = \min_{X:rank(X)=k} \|A - X\|_F \longleftarrow \textit{Frobenius norm}$$

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}.$$

A_k and X are both $m \times n$ matrices.

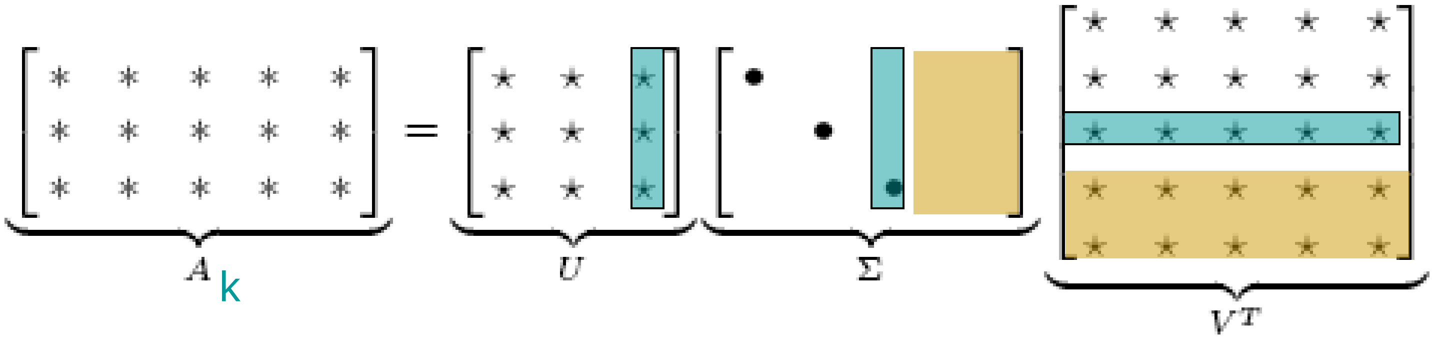
Typically, want $k << r$.

Low-rank Approximation

- Solution via SVD

$$A_k = U\,\mathrm{diag}(\sigma_1,\ldots,\sigma_k,\underbrace{0,\ldots,0}_{})V^T$$

set smallest r-k
singular values to zero



$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \longleftarrow$$

column notation: sum of rank 1 matrices

Approximation error

- How good (bad) is this approximation?
- It's the best possible, measured by the Frobenius norm of the error:

$$\min_{X:rank(X)=k} \|A - X\|_F = \|A - A_k\|_F$$

where the σ_i are ordered such that $\sigma_i \geq \sigma_{i+1}$.

Suggests why Frobenius error drops as *k* increased.

Recall random projection

- Completely different method for low-rank approximation
- Was *data-oblivious*
  - SVD-based approximation is *data-dependent*
- Error for random projection depended only on start/finish dimensionality
  - For every distance
- Error for SVD-based approximation is for the Frobenius norm, not for individual distances

SVD Low-rank approximation

- Whereas the term-doc matrix A may have m=50000, n=10 million (and rank close to 50000)
- We can construct an approximation A_{100} with rank 100.
  - <u>Of all rank 100 matrices, it would have the lowest Frobenius error.</u>

C. Eckart, G. Young, *The approximation of a matrix by another of lower rank*. Psychometrika, 1, 211-218, 1936.

Power Method for SVD

- $A = \sum_i \delta_i u_i v_i^T$
- $B = A^T A$

  $= (\sum_i \delta_i v_i u_i^T)\,(\sum_j \delta_j u_j v_j^T) = \sum_i \delta_i^2 v_i v_i^T$

  ...
- $B^k = \sum_i \delta_i^{2k} v_i v_i^T$
- If $\delta_1 > \delta_2$ then the first term in the summation dominates. i.e., $B^k \to \delta_1^{2k} v_1 v_1^T$
- To estimate v_1: take the first column of B^k and normalize it to an unit vector.

Power Method for SVD

- $A = \sum_i \delta_i u_i v_i^T$, A is large and sparse
- $B^k = \sum_i \delta_i^{2k} v_i v_i^T$ could be very cost.
- Improve:
  - Randomly select $x = \sum_{i=1}^{d} c_i v_i$
  - $B^k \cdot x \approx (\delta_1^{2k} v_1 v_1^T)x = c_1 \delta_1^{2k} v_1 = y$
  - Normalize y will get v_1.
  - Trick: $B^k \cdot x = A^T A \cdots A^T Ax$

Matrix Approximation

$A_i = U\Sigma_i V^T$

Σ_i : the rank i version of Σ (by setting last $m-i$ σ's to zero)

A_i : the best rank i approximation to A in the sence of Euclidean distance

$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_m u_m v_m^T$

Storage save : rank one matrix $(m+n)$ numbers

Operation save : $m+n$

making small σ's to zero and back substitute
(see next page for application in image compression)

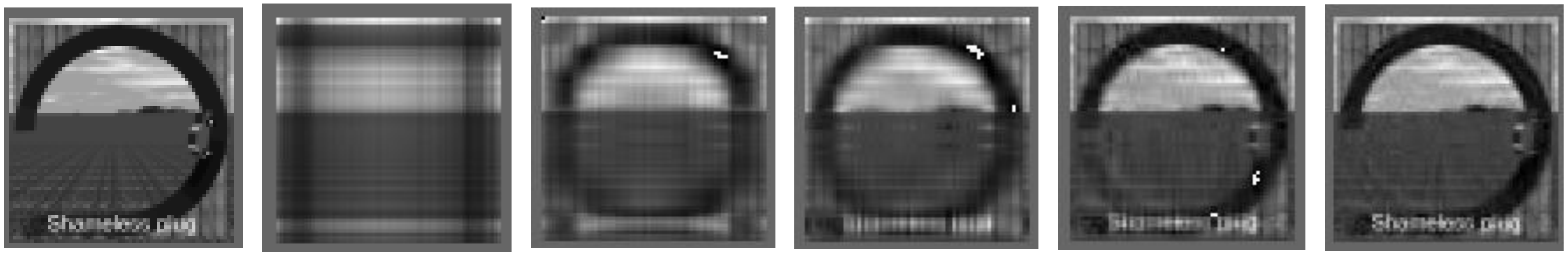
Image Compression

- For grey scale images: m×n bytes

After SVD, taking the most significan t r terms:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

- Only need to store r×(m+n+1)



Original

64×64

r = 1,3,5,10,16 (no perceivable difference afterwards)

Mathematical background

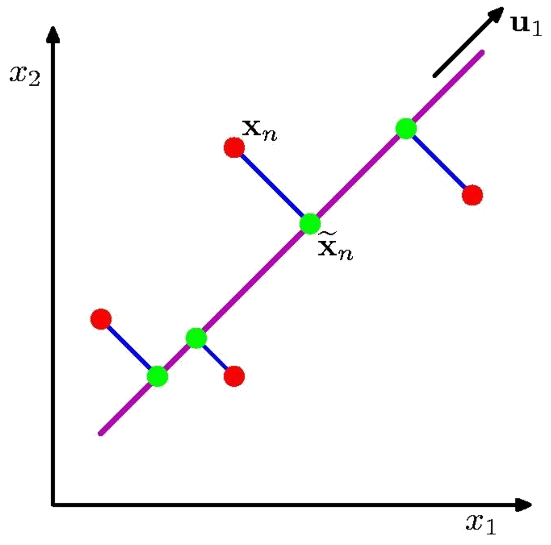
Singular vector decomposition

Principle component analysis

PCA Applications

- Data Visualization
- Data Compression
- Noise Reduction
- Data Classification
- Trend Analysis
- Factor Analysis

Principle Component Analysis



PCA:

Orthogonal projection of data onto lower-dimension linear space that...

– maximizes variance of projected data (purple line)

– minimizes mean squared distance between

  • data point and

  • projections (sum of blue lines)

Principle Components Analysis

Idea:

- Given data points in a d-dimensional space, project into lower dimensional space while preserving as much information as possible
  - Eg, find best planar approximation to 3D data
  - Eg, find best 12-D approximation to 10^4-D data
- In particular, choose projection that minimizes *squared error* in reconstructing original data

Covariance

- **Variance** – measure of the deviation from the mean for points in one dimension e.g. heights

$$Var(X) = E((x - \mu)^2)$$

- **Covariance** as a measure of how much each of the dimensions vary from the mean with respect to each other.

$$Cov(X,Y) = \boldsymbol{E}[((X - \boldsymbol{E}(X)(Y - \boldsymbol{E}(Y))]$$

- Covariance is measured between 2 dimensions to see if there is a *relationship between the 2 dimensions* e.g. number of hours studied & marks obtained.
- The covariance between one dimension and itself is the variance

Covariance Matrix

- Representing Covariance between dimensions as a matrix e.g. for 3 dimensions:

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

- Diagonal is the variances of x, y and z
- $cov(x, y) \ = \ cov(y, x)$ hence matrix is symmetrical about the diagonal
- n-dimensional data will result in $n \times n$ covariance matrix.

Covariance

- Exact value is not as important as it’s sign.
- A <u>positive value</u> of covariance indicates both dimensions increase or decrease together
  - e.g. as the number of hours studied increases, the marks in that subject increase.
- A <u>negative value</u> indicates while one increases the other decreases, or vice-versa
  - e.g. active social life at WoL vs performance in CS dept.
- If <u>covariance is zero</u>: the two dimensions are independent of each other
  - e.g. heights of students vs the marks obtained in a subject.

PCA

- **Principal Components Analysis (PCA)** is a technique that can be used to simplify a dataset.
- It is a linear transformation that chooses a new coordinate system for the data set such that
  - greatest variance by any projection of the data set comes to lie on the first axis (then called the first principal component),
  - the second greatest variance on the second axis, and so on.
- PCA can be used for reducing dimensionality by eliminating the later principal components.

PCA

- By finding the eigenvalues and eigenvectors of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to *the dimensions that have the strongest correlation in the dataset*.
- This is the **principal component**.
- PCA is a useful statistical technique that has found application in:
  - fields such as face recognition and image compression
  - finding patterns in data of high dimension

PCA process –STEP 1

- Subtract the mean

  from each of the data dimensions. All the x values have $\bar{x}$ subtracted and y values have $\bar{y}$ subtracted from them. This produces a data set whose mean is zero.

Subtracting the mean makes variance and covariance calculation easier by simplifying their equations. The variance and co-variance values are not affected by the mean value.

PCA process –STEP 1

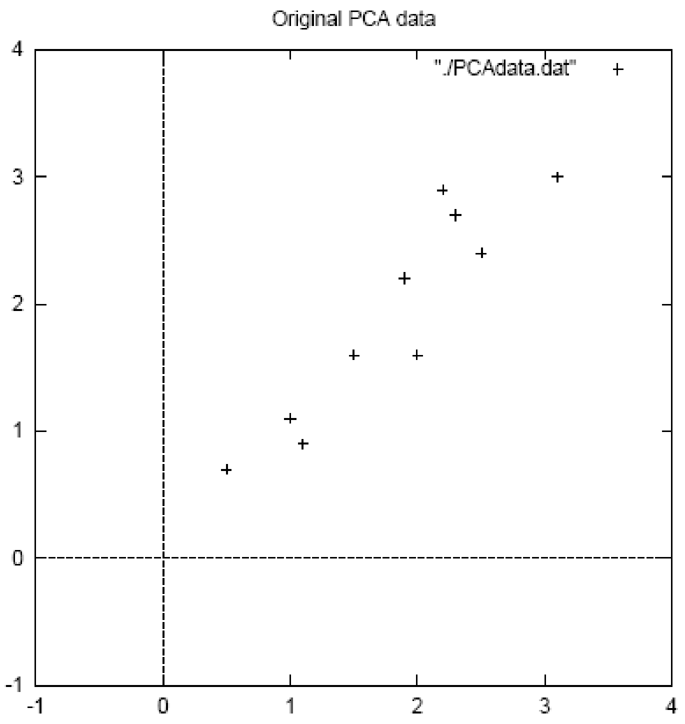
DATA:

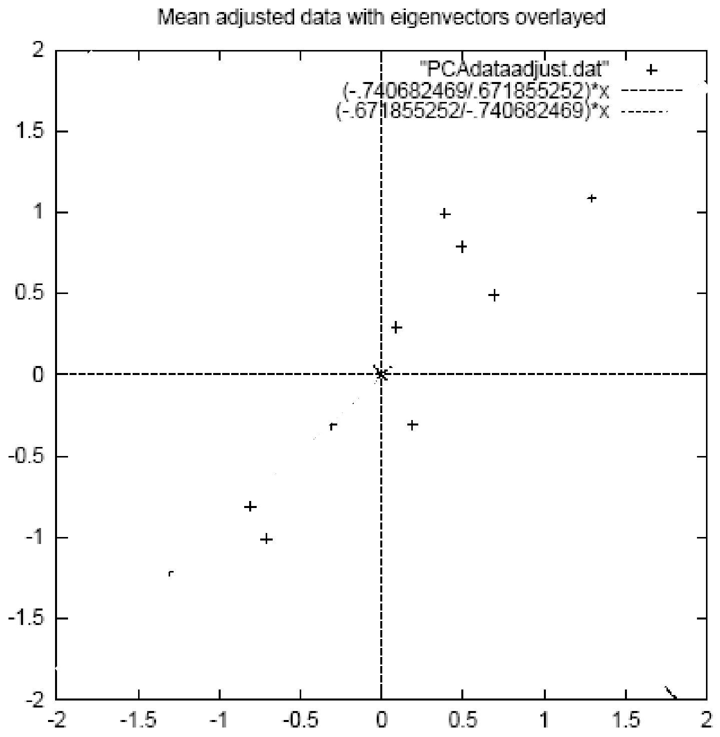
x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

ZERO MEAN DATA:

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

PCA process –STEP 1





PCA process –STEP 2

- Calculate the covariance matrix

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

Since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

PCA process –STEP 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

PCA process –STEP 3

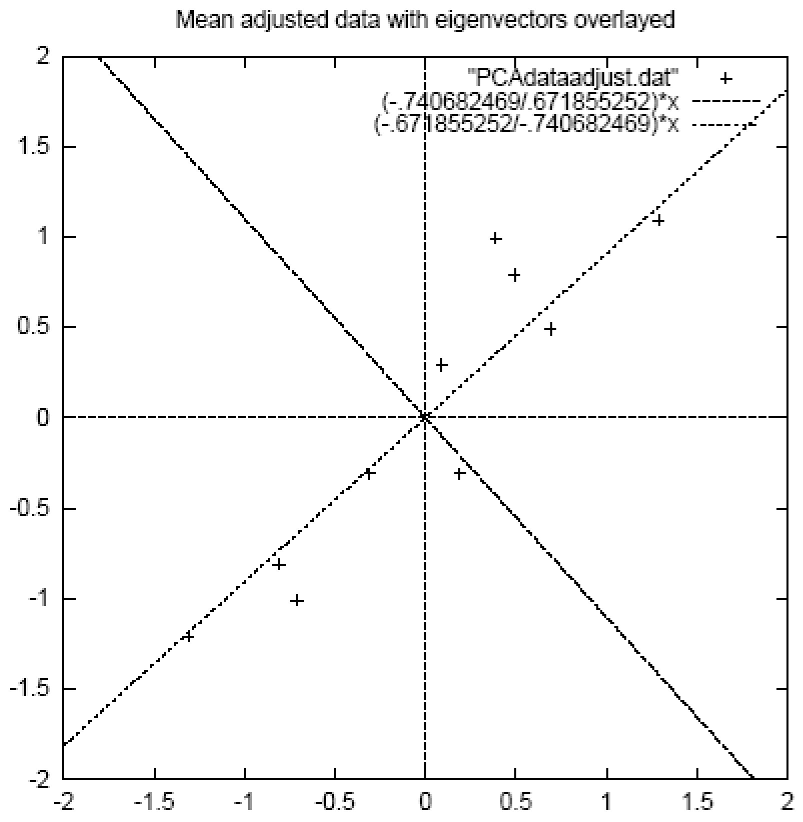


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

•eigenvectors are plotted as diagonal dotted lines on the plot.

They are perpendicular to each other.

•Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.

The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

PCA process –STEP 4

- Reduce dimensionality and form *feature vector* the eigenvector with the *highest* eigenvalue is the *principle component* of the data set.

- In our example, the eigenvector with the larges eigenvalue was the one that pointed down the middle of the data.

- Once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance.

PCA process –STEP 4

- Now, if you like, you can decide to *ignore* the components of lesser significance

- You do lose some information, but if the eigenvalues are small, you don’t lose much

  – n dimensions in your data
  – calculate n eigenvectors and eigenvalues
  – choose only the first p eigenvectors
  – final data set has only p dimensions.

PCA process –STEP 4

- Feature Vector

  FeatureVector = (eig1 eig2 eig3 … eign)

  We can either form a feature vector with both of the eigenvectors:

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

  or, we can choose to leave out the smaller, less significant component and only have a single column:

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

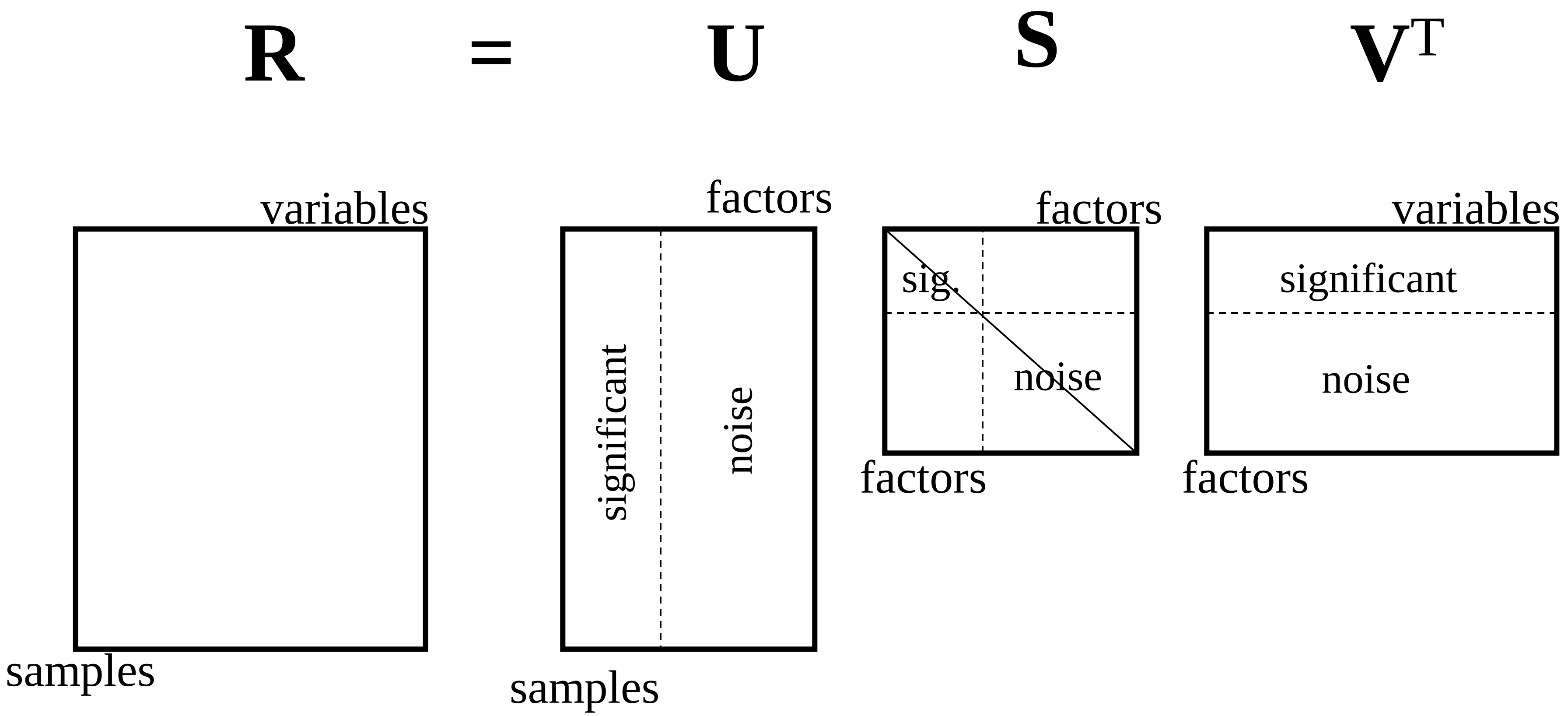
PCA process –STEP 5

- Deriving the new data

  FinalData = RowFeatureVector x RowZeroMeanData

- RowFeatureVector is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top.
- RowZeroMeanData is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension.

PCA process –STEP 5



PCA process –STEP 5

- FinalData is the final data set, <u>with data items in columns, and dimensions along rows.</u>
- <u>What will this give us?</u>
  - It will give us the original data *solely in terms of the vectors we chose*.
- We have changed our data from being in terms of the axes x and y , and now they are in terms of our 2 eigenvectors.

PCA process –STEP 5

FinalData transpose:
dimensions along columns

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

PCA process –STEP 5

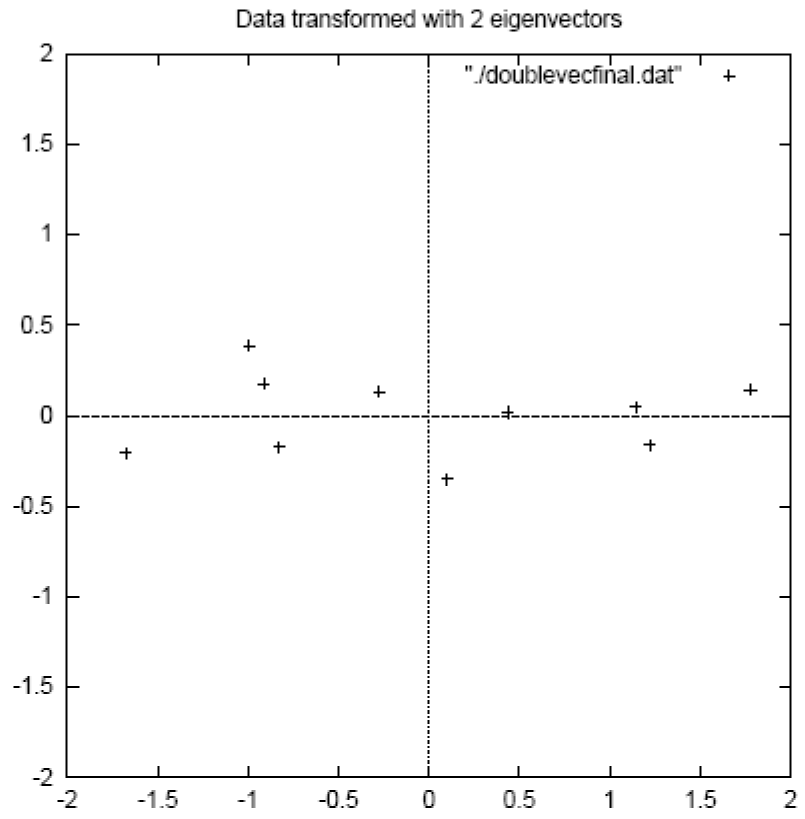


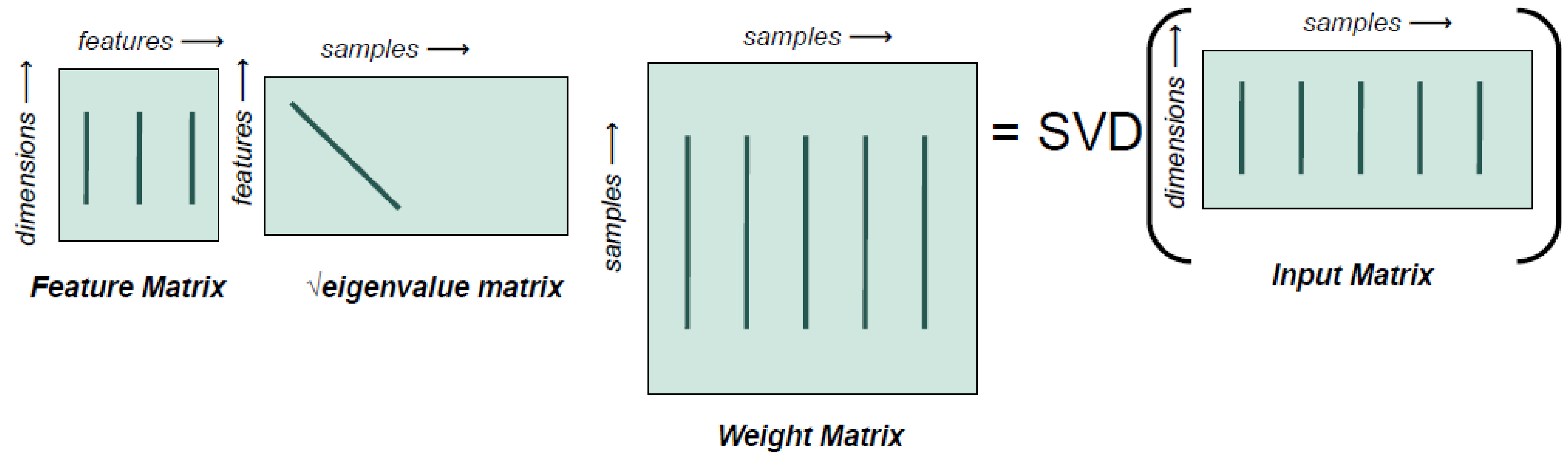
Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

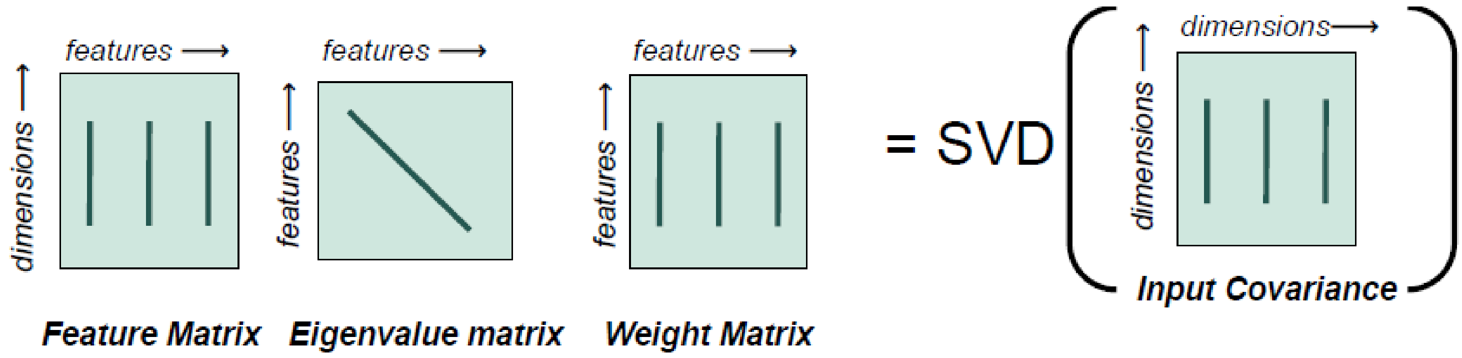
PCA vs SVD

A better way to compute PCA

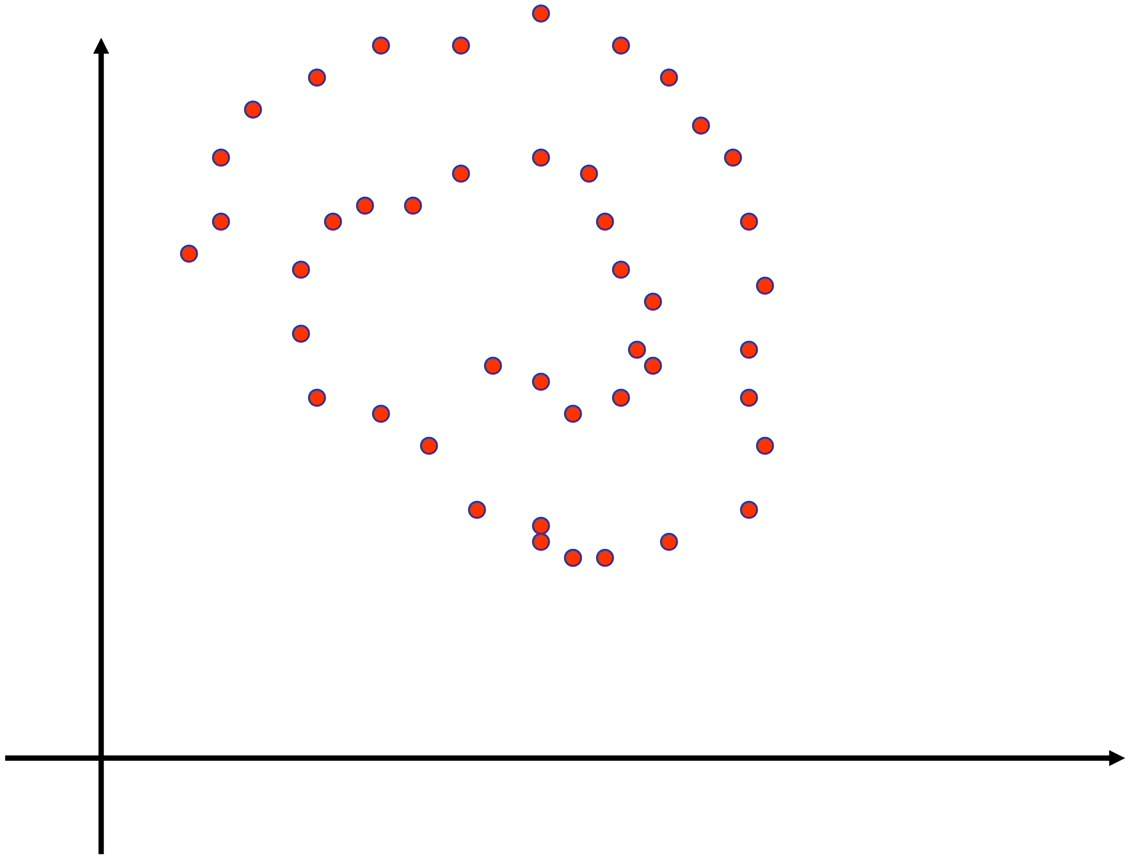
- SVD
- Why?
  - More stable, robust and fancy extensions!

PCA through the SVD



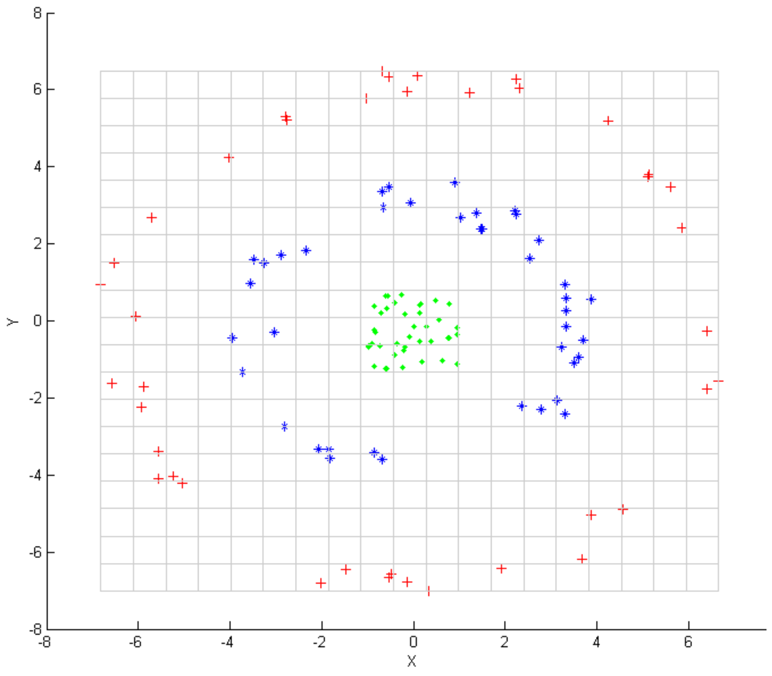


PCA, a Problematic Data Set

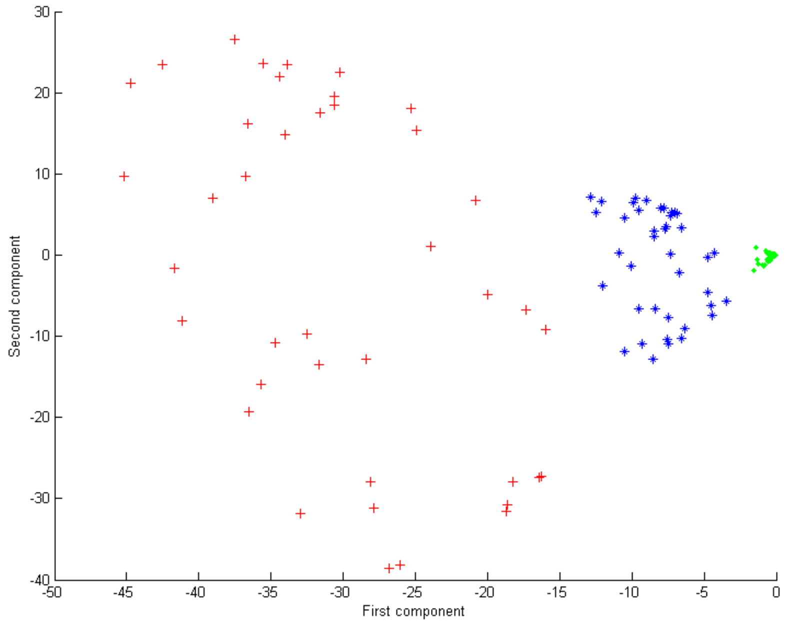


PCA cannot capture NON-LINEAR structure!

Kernel PCA



8
6
4
2
0
-2
-4
-6
-8
Y
-8
-6
-4
-2
0
2
4
6
X



30
20
10
0
-10
-20
-30
-40
Second component
-50
-45
-40
-35
-30
-25
-20
-15
-10
-5
0
First component

PCA Conclusions

- PCA
  - finds orthonormal basis for data
  - Sorts dimensions in order of "importance"
  - Discard low significance dimensions

- Uses:
  - Get compact description
  - Ignore noise
  - Improve classification (hopefully)

- Not magic:
  - Doesn't know class labels
  - Can only capture linear variations

- One of many tricks to reduce dimensionality!