


# **Data and Data Exploration**


Xiaochun MAI

Shenzhen University

# Outline

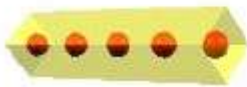
1. Data Attribute types
2. Types of Data Sets
3. Characteristics of Structured Data
4. Data Preprocessing 

# Data Preprocessing

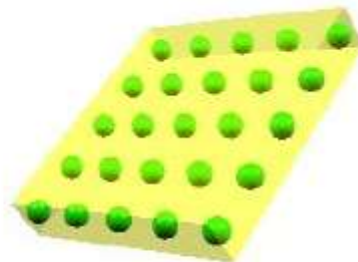
- 1) Data Quality Issues
- 2) Data Preprocessing to address quality issues
  - 1) Data Cleaning
  - 2) Aggregation
  - 3) Sampling
  - 4) Dimensionality Reduction 
  - 5) Feature Subset Selection
  - 6) Feature Generation/Creation
  - 7) Discretization and Binarization
  - 8) Attribute Transformation

# Curse of Dimensionality

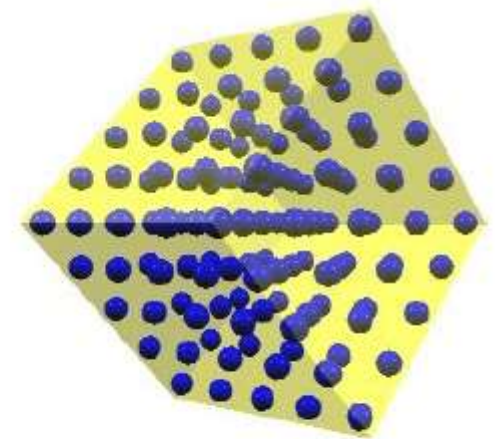
- One challenge of mining high-dimensional data is **insufficient data samples**
- Suppose 5 samples/objects is considered enough in 1-D
  - 1D : 5 points ( $5^1$ )
  - 2D : 25 points ( $5^2$ )
  - 3D : 125 points ( $5^3$ )
  - 10D : 9,765,625 points ( $5^{10}$ )



5 points



25 points



125 points

# 4) Dimensionality Reduction

## Simply our data

- **Purposes:**

- Avoid curse of dimensionality
- Reduce amount of time and memory required by machine learning algorithms
- Allow data to be more easily visualized
- May help to eliminate irrelevant features or reduce noise

- **Techniques**

- Principle Component Analysis (PCA).
- Singular Value Decomposition (SVD)

# Philosophy of PCA

- When a data set has too many variables that are **correlated**, how do you want to handle it?
- If we directly construct a model using all the correlated variables, then we could get low prediction results
- We need to think some strategic method to **find few important uncorrelated variables** (in form of components) from a large set of original correlated variables available in a data set.
- PCA helps to overcome such challenges, which was by Pearson (1901) and Hotelling (1933).

# Philosophy of PCA

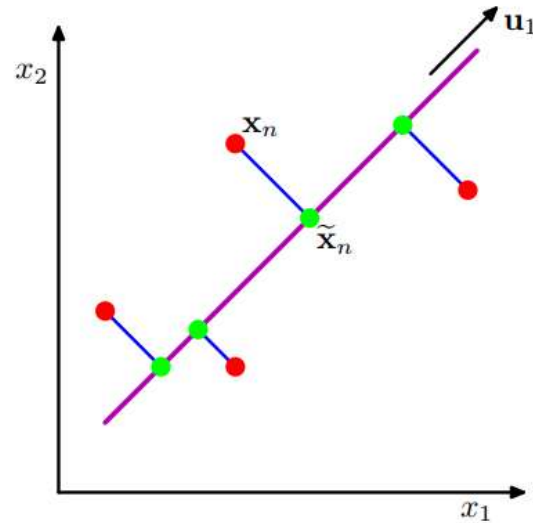
- Key idea is to extract low dimensional set of features from a high dimensional data set ( $\geq 3$ ) with a motivation *to capture as much information as possible*.
- Assume that a data set has dimension  $1000 (n) \times 100 (p)$ , where  $n$  represents the number of observations/objects and  $p$  represents number of variables.
- One straightforward way to analyse the *correlation between variables* is to construct *scatter plots* for each pair of variable. Unfortunately, we will have  $p(p-1)/2$  (499,500) variable pairs.

# Philosophy of PCA

- Principal Component Analysis (PCA): Find a (linear) projection that
  - Minimize reconstruction error (Pearson, 1901)
  - Maximize the variance (signal) of the projected data (Hotelling, 1933)



# Philosophy of PCA



From PRML (Bishop, 2006)

- ▶ Two-dimensional data  $\mathbf{x} = [x_1, x_2]^\top$  projected onto a one-dimensional linear manifold (affine subspace) with direction  $\mathbf{u}_1$ .
- ▶ **Red:** Original data, **Green:** Projected data

# Eigenvalues & Eigenvectors

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

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

(right) eigenvector  $\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0}$       eigenvalue  $\lambda \in \mathbb{R}$

*Example*

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

- How many eigenvalues are there at most?

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

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

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

# 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

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

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:

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

$$Sx = 2Sv_1 + 4Sv_2 + 6Sv_3 = 2\lambda_1v_1 + 4\lambda_2v_2 + 6\lambda_3v_3$$

- 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**.

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

$$\underbrace{\forall w \in \mathbb{R}^n, w^T S w \geq 0}_{\text{positive semidefinite}}, \text{ then if } Sv = \lambda v \Rightarrow \lambda \geq 0$$

# Example

- Let

$$S = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \leftarrow \text{Real, symmetric.}$$

- 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  $S \in \mathbb{R}^{m \times m}$  be a **square** matrix with  **$m$  linearly independent eigenvectors** (a “non-defective” matrix)

- Theorem:** Exists an **eigen decomposition**

$$S = U \Lambda U^{-1}$$

*diagonal*

Unique  
for  
distinct  
eigen-  
values

– (cf. matrix diagonalization theorem)

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

$$\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

# Diagonal decomposition: why/how

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

Then,  $\mathbf{SU}$  can be written

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

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

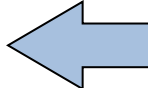
And  $\mathbf{S}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ .



# Diagonal decomposition - example

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

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,  $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{bmatrix}$

# Example continued

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

Then,  $\mathbf{S} = \underbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}}_{\mathbf{\Lambda}} \underbrace{\begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}_{(\mathbf{Q}^{-1} = \mathbf{Q}^T)}$

Why? Stay tuned ...

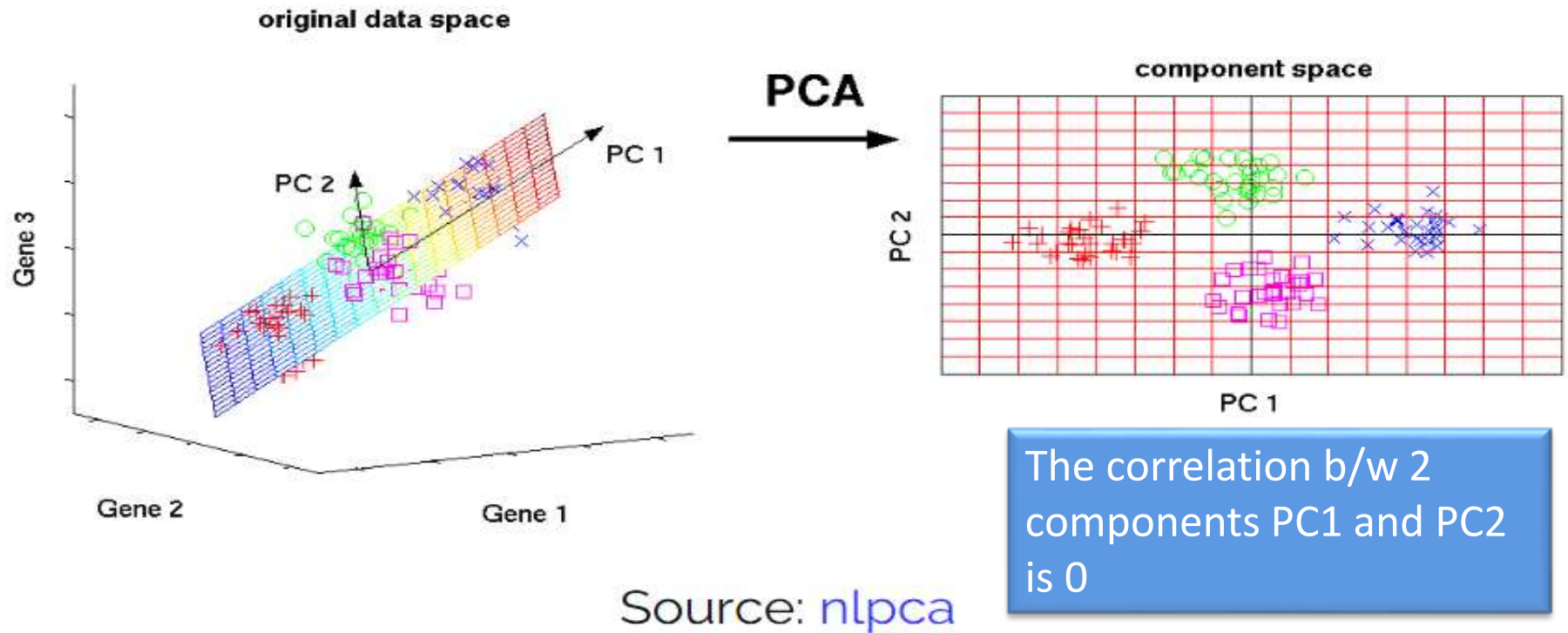
# Symmetric Eigen Decomposition

- If  $S \in \mathbb{R}^{m \times m}$  is a **symmetric** matrix:
- **Theorem**: Exists a (unique) **eigen decomposition**

$$S = Q\Lambda Q^T$$

- where  $Q$  is **orthogonal**:
  - $Q^{-1} = Q^T$
  - Columns of  $Q$  are normalized eigenvectors
  - Columns are orthogonal.
  - (everything is real)

# An Illustrative Example of PCA



- The transformation of 3D data to 2D (lower dimension) using PCA. We observe that similar data in 3D are still similar in 2D.
- Note each resultant (new) dimension, namely, PC1 and PC2, is a *linear combination* of original  $p$  features ( $p=3$  here). In addition, they (PC1 and PC2) are *orthogonal* (perpendicular, not correlated)

# What are the principal components ?

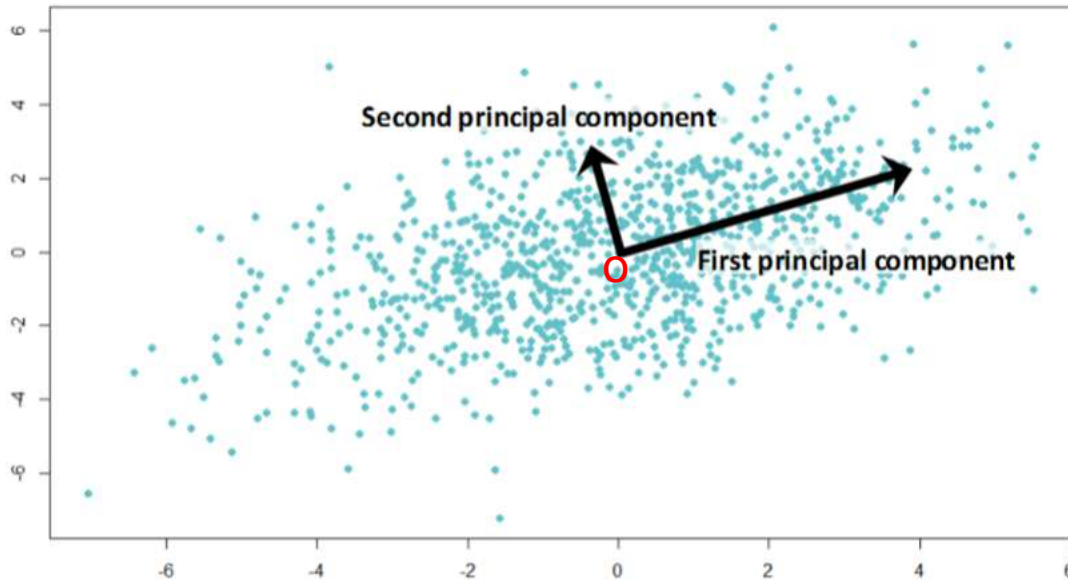
- A principal component is a **normalized linear combination of the original variables**/predictors in a data set. In our figure, **PC1** and **PC2** are the principal components. Let's say a set of variables as  $X_1, X_2, \dots, X_p$  (e.g., Gene 1, Gene 2, Gene 3) are given in our data, the principal component can be written as:

- **PC1** =  $\Phi_{11}X_1 + \Phi_{21}X_2 + \Phi_{31}X_3 + \dots + \Phi_{p1}X_p$
- **PC2** =  $\Phi_{12}X_1 + \Phi_{22}X_2 + \Phi_{32}X_3 + \dots + \Phi_{p2}X_p$
- Loading vector  $(\Phi_{11} \ \Phi_{21} \ \Phi_{31} \ \dots \ \Phi_{p1})$  satisfies
$$(\Phi_{11})^2 + (\Phi_{21})^2 + (\Phi_{31})^2 + \dots + (\Phi_{p1})^2 = 1$$

# Loading vector

- loading vector ( $\Phi_{11} \Phi_{21} \Phi_{31} \dots \Phi_{p1}$ ) defines the direction of PC1 along which *data varies the most* (maximum variance, deviating from data center). It results in *a line* in  $p$  dimensional space which *is closest to the  $n$  observations in our given data*, by minimizing the sum of squared distance between a data point and the line (like linear regression).
- Larger the variability captured in first component, larger the information captured by component.
- PC2 is also a linear combination of original predictors which capture the remaining variance in the data set and is uncorrelated with PC1.

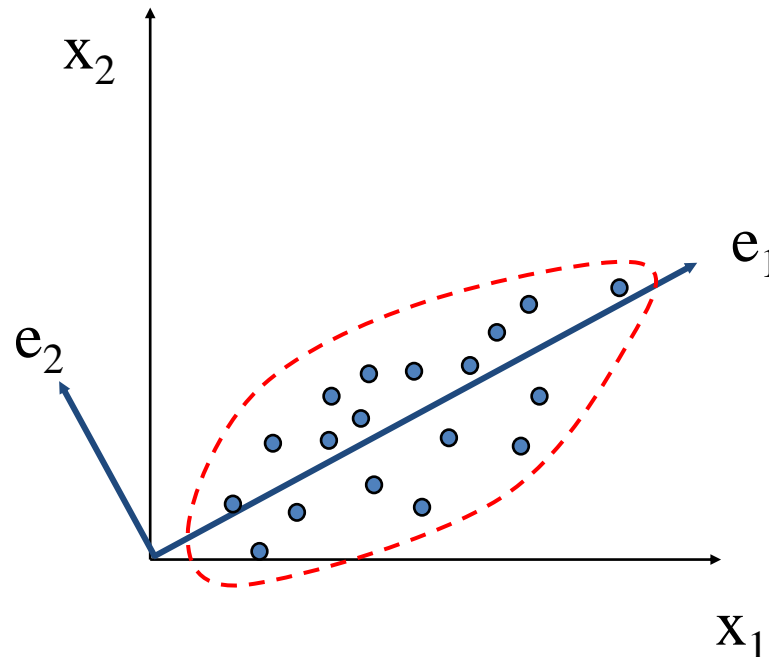
# Illustration of the PC1 and PC2



- All succeeding principal components (e.g. PC3) capture the remaining variation without being correlated with the previous components (PC1, PC2)
- PCA can be applied only on numerical data. Therefore, if the data has categorical variables they must be converted to numerical.

# Mathematically, PCA

- Approach:
  - Find the eigenvectors of the covariance matrix
- The eigenvectors define the new space
- Rank Eigen values and get PC1 (with the largest Eigen values), PC2 (with the second largest Eigen values), ....

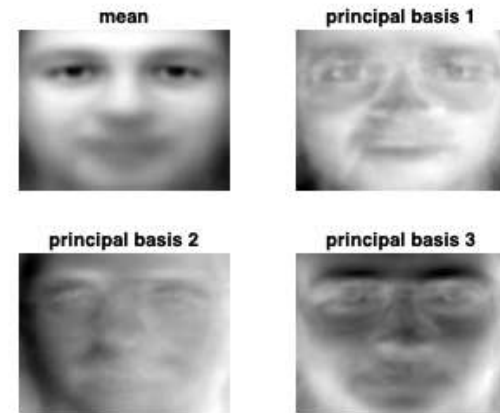




# Principal components analysis (PCA)



(a)



(b)

Figure 20.3: a) Some randomly chosen  $64 \times 64$  pixel images from the Olivetti face database. (b) The mean and the first three PCA components represented as images. Generated by [pcaImageDemo.ipynb](#).

# Principal components analysis (PCA)

The simplest and most widely used form of dimensionality reduction is **principal components analysis** or **PCA**. The basic idea is to find a linear and orthogonal projection of the high dimensional data  $\mathbf{x} \in \mathbb{R}^D$  to a low dimensional subspace  $\mathbf{z} \in \mathbb{R}^L$ , such that the low dimensional representation is a “good approximation” to the original data, in the following sense: if we project or **encode**  $\mathbf{x}$  to get  $\mathbf{z} = \mathbf{W}^\top \mathbf{x}$ , and then unproject or **decode**  $\mathbf{z}$  to get  $\hat{\mathbf{x}} = \mathbf{W}\mathbf{z}$ , then we want  $\hat{\mathbf{x}}$  to be close to  $\mathbf{x}$  in  $\ell_2$  distance. In particular, we can define the following **reconstruction error** or **distortion**:

$$\mathcal{L}(\mathbf{W}) \triangleq \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \text{decode}(\text{encode}(\mathbf{x}_n; \mathbf{W}); \mathbf{W})\|_2^2 \quad (20.1)$$

where the encode and decoding stages are both linear maps, as we explain below.

In Section 20.1.2, we show that we can minimize this objective by setting  $\hat{\mathbf{W}} = \mathbf{U}_L$ , where  $\mathbf{U}_L$  contains the  $L$  eigenvectors with largest eigenvalues of the empirical covariance matrix

$$\hat{\Sigma} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top = \frac{1}{N} \mathbf{X}_c^\top \mathbf{X}_c \quad (20.2)$$

where  $\mathbf{X}_c$  is a centered version of the  $N \times D$  design matrix. In Section 20.2.2, we show that this is equivalent to maximizing the likelihood of a latent linear Gaussian model known as probabilistic PCA.

# Derivation of the algorithm

Suppose we have an (unlabeled) dataset  $\mathcal{D} = \{\mathbf{x}_n : n = 1 : N\}$ , where  $\mathbf{x}_n \in \mathbb{R}^D$ . We can represent this as an  $N \times D$  data matrix  $\mathbf{X}$ . We will assume  $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n = \mathbf{0}$ , which we can ensure by centering the data.

We would like to approximate each  $\mathbf{x}_n$  by a low dimensional representation,  $\mathbf{z}_n \in \mathbb{R}^L$ . We assume that each  $\mathbf{x}_n$  can be “explained” in terms of a weighted combination of basis functions  $\mathbf{w}_1, \dots, \mathbf{w}_L$ , where each  $\mathbf{w}_k \in \mathbb{R}^D$ , and where the weights are given by  $\mathbf{z}_n \in \mathbb{R}^L$ , i.e., we assume  $\mathbf{x}_n \approx \sum_{k=1}^L z_{nk} \mathbf{w}_k$ . The vector  $\mathbf{z}_n$  is the low dimensional representation of  $\mathbf{x}_n$ , and is known as the **latent vector**, since it consists of latent or “hidden” values that are not observed in the data. The collection of these latent variables are called the **latent factors**.

We can measure the error produced by this approximation as follows:

$$\mathcal{L}(\mathbf{W}, \mathbf{Z}) = \frac{1}{N} \|\mathbf{X} - \mathbf{Z}\mathbf{W}^\top\|_F^2 = \frac{1}{N} \|\mathbf{X}^\top - \mathbf{W}\mathbf{Z}^\top\|_F^2 = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{W}\mathbf{z}_n\|^2 \quad (20.3)$$

where the rows of  $\mathbf{Z}$  contain the low dimension versions of the rows of  $\mathbf{X}$ . This is known as the (average) **reconstruction error**, since we are approximating each  $\mathbf{x}_n$  by  $\hat{\mathbf{x}}_n = \mathbf{W}\mathbf{z}_n$ .

We want to minimize this subject to the constraint that  $\mathbf{W}$  is an orthogonal matrix. Below we show that the optimal solution is obtained by setting  $\hat{\mathbf{W}} = \mathbf{U}_L$ , where  $\mathbf{U}_L$  contains the  $L$  eigenvectors with largest eigenvalues of the empirical covariance matrix.

# Base case

Let us start by estimating the best 1d solution,  $\mathbf{w}_1 \in \mathbb{R}^D$ . We will find the remaining basis vectors  $\mathbf{w}_2, \mathbf{w}_3$ , etc. later.

Let the coefficients for each of the data points associated with the first basis vector be denoted by  $\tilde{\mathbf{z}}_1 = [z_{11}, \dots, z_{N1}] \in \mathbb{R}^N$ . The reconstruction error is given by

$$\mathcal{L}(\mathbf{w}_1, \tilde{\mathbf{z}}_1) = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - z_{n1} \mathbf{w}_1\|^2 = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - z_{n1} \mathbf{w}_1)^\top (\mathbf{x}_n - z_{n1} \mathbf{w}_1) \quad (20.4)$$

$$= \frac{1}{N} \sum_{n=1}^N [\mathbf{x}_n^\top \mathbf{x}_n - 2z_{n1} \mathbf{w}_1^\top \mathbf{x}_n + z_{n1}^2 \mathbf{w}_1^\top \mathbf{w}_1] \quad (20.5)$$

$$= \frac{1}{N} \sum_{n=1}^N [\mathbf{x}_n^\top \mathbf{x}_n - 2z_{n1} \mathbf{w}_1^\top \mathbf{x}_n + z_{n1}^2] \quad (20.6)$$

since  $\mathbf{w}_1^\top \mathbf{w}_1 = 1$  (by the orthonormality assumption). Taking derivatives wrt  $z_{n1}$  and equating to zero gives

$$\frac{\partial}{\partial z_{n1}} \mathcal{L}(\mathbf{w}_1, \tilde{\mathbf{z}}_1) = \frac{1}{N} [-2\mathbf{w}_1^\top \mathbf{x}_n + 2z_{n1}] = 0 \Rightarrow z_{n1} = \mathbf{w}_1^\top \mathbf{x}_n \quad (20.7)$$

# Base case

So the optimal embedding is obtained by orthogonally projecting the data onto  $\mathbf{w}_1$  (see Figure 20.1). Plugging this back in gives the loss for the weights:

$$\mathcal{L}(\mathbf{w}_1) = \mathcal{L}(\mathbf{w}_1, \tilde{\mathbf{z}}_1^*(\mathbf{w}_1)) = \frac{1}{N} \sum_{n=1}^N [\mathbf{x}_n^\top \mathbf{x}_n - z_{n1}^2] = \text{const} - \frac{1}{N} \sum_{n=1}^N z_{n1}^2 \quad (20.8)$$

To solve for  $\mathbf{w}_1$ , note that

$$\mathcal{L}(\mathbf{w}_1) = -\frac{1}{N} \sum_{n=1}^N z_{n1}^2 = -\frac{1}{N} \sum_{n=1}^N \mathbf{w}_1^\top \mathbf{x}_n \mathbf{x}_n^\top \mathbf{w}_1 = -\mathbf{w}_1^\top \hat{\Sigma} \mathbf{w}_1 \quad (20.9)$$

where  $\Sigma$  is the empirical covariance matrix (since we assumed the data is centered). We can trivially optimize this by letting  $\|\mathbf{w}_1\| \rightarrow \infty$ , so we impose the constraint  $\|\mathbf{w}_1\| = 1$  and instead optimize

$$\tilde{\mathcal{L}}(\mathbf{w}_1) = \mathbf{w}_1^\top \hat{\Sigma} \mathbf{w}_1 - \lambda_1 (\mathbf{w}_1^\top \mathbf{w}_1 - 1) \quad (20.10)$$

where  $\lambda_1$  is a Lagrange multiplier (see Section 8.5.1). Taking derivatives and equating to zero we have

$$\frac{\partial}{\partial \mathbf{w}_1} \tilde{\mathcal{L}}(\mathbf{w}_1) = 2\hat{\Sigma} \mathbf{w}_1 - 2\lambda_1 \mathbf{w}_1 = 0 \quad (20.11)$$

$$\hat{\Sigma} \mathbf{w}_1 = \lambda_1 \mathbf{w}_1 \quad (20.12)$$

Hence the optimal direction onto which we should project the data is an eigenvector of the covariance matrix. Left multiplying by  $\mathbf{w}_1^\top$  (and using  $\mathbf{w}_1^\top \mathbf{w}_1 = 1$ ) we find

$$\mathbf{w}_1^\top \hat{\Sigma} \mathbf{w}_1 = \lambda_1 \quad (20.13)$$

Since we want to maximize this quantity (minimize the loss), we pick the eigenvector which corresponds to the largest eigenvalue.



# Optimal weight vector maximizes the variance of the projected data

Before continuing, we make an interesting observation. Since the data has been centered, we have

$$\mathbb{E}[z_{n1}] = \mathbb{E}[\mathbf{x}_n^\top \mathbf{w}_1] = \mathbb{E}[\mathbf{x}_n]^\top \mathbf{w}_1 = 0 \quad (20.14)$$

Hence variance of the projected data is given by

$$\mathbb{V}[\tilde{\mathbf{z}}_1] = \mathbb{E}[\tilde{\mathbf{z}}_1^2] - (\mathbb{E}[\tilde{\mathbf{z}}_1])^2 = \frac{1}{N} \sum_{n=1}^N z_{n1}^2 - 0 = -\mathcal{L}(\mathbf{w}_1) + \text{const} \quad (20.15)$$

From this, we see that *minimizing* the reconstruction error is equivalent to *maximizing* the variance of the projected data:

$$\arg \min_{\mathbf{w}_1} \mathcal{L}(\mathbf{w}_1) = \arg \max_{\mathbf{w}_1} \mathbb{V}[\tilde{\mathbf{z}}_1(\mathbf{w}_1)] \quad (20.16)$$

This is why it is often said that PCA finds the directions of maximal variance. (See Figure 20.4 for an illustration.) However, the minimum error formulation is easier to understand and is more general.

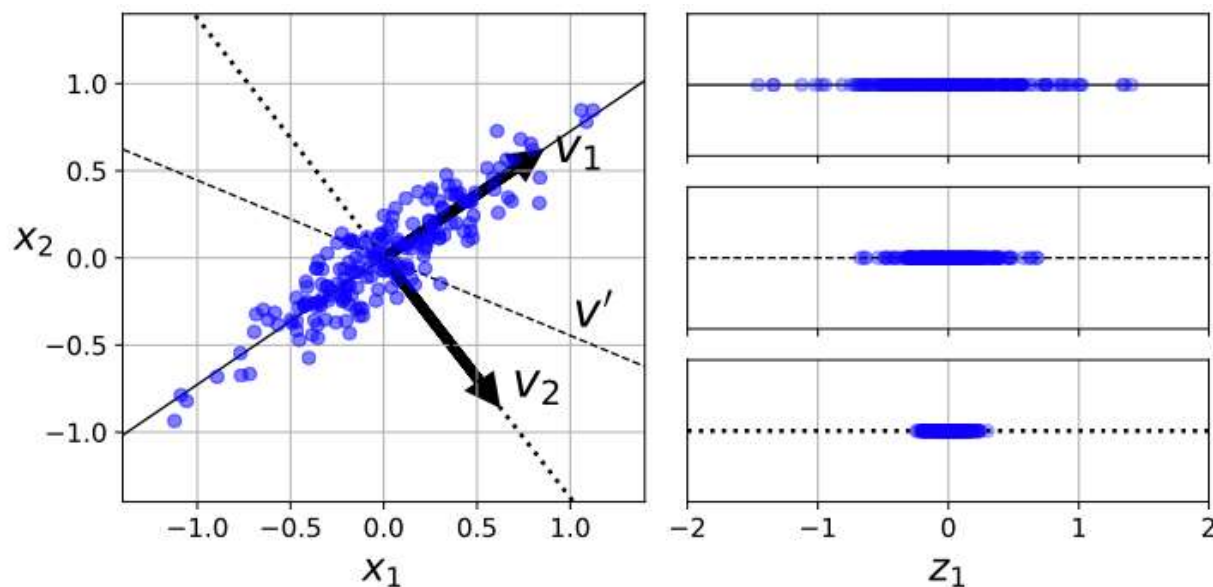


Figure 20.4: Illustration of the variance of the points projected onto different 1d vectors.  $v_1$  is the first principal component, which maximizes the variance of the projection.  $v_2$  is the second principal component which is direction orthogonal to  $v_1$ . Finally  $v'$  is some other vector in between  $v_1$  and  $v_2$ . Adapted from Figure 8.7 of [Gér19]. Generated by `pca_projected_variance.ipynb`

# Induction step

Now let us find another direction  $\mathbf{w}_2$  to further minimize the reconstruction error, subject to  $\mathbf{w}_1^\top \mathbf{w}_2 = 0$  and  $\mathbf{w}_2^\top \mathbf{w}_2 = 1$ . The error is

$$\mathcal{L}(\mathbf{w}_1, \tilde{\mathbf{z}}_1, \mathbf{w}_2, \tilde{\mathbf{z}}_2) = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - z_{n1} \mathbf{w}_1 - z_{n2} \mathbf{w}_2\|^2 \quad (20.17)$$

Optimizing wrt  $\mathbf{w}_1$  and  $\mathbf{z}_1$  gives the same solution as before. Exercise 20.3 asks you to show that  $\frac{\partial \mathcal{L}}{\partial \mathbf{z}_2} = 0$  yields  $z_{n2} = \mathbf{w}_2^\top \mathbf{x}_n$ . Substituting in yields

$$\mathcal{L}(\mathbf{w}_2) = \frac{1}{N} \sum_{n=1}^N [\mathbf{x}_n^\top \mathbf{x}_n - \mathbf{w}_1^\top \mathbf{x}_n \mathbf{x}_n^\top \mathbf{w}_1 - \mathbf{w}_2^\top \mathbf{x}_n \mathbf{x}_n^\top \mathbf{w}_2] = \text{const} - \mathbf{w}_2^\top \hat{\Sigma} \mathbf{w}_2 \quad (20.18)$$

Dropping the constant term, plugging in the optimal  $\mathbf{w}_1$  and adding the constraints yields

$$\tilde{\mathcal{L}}(\mathbf{w}_2) = -\mathbf{w}_2^\top \hat{\Sigma} \mathbf{w}_2 + \lambda_2 (\mathbf{w}_2^\top \mathbf{w}_2 - 1) + \lambda_{12} (\mathbf{w}_2^\top \mathbf{w}_1 - 0) \quad (20.19)$$

Exercise 20.3 asks you to show that the solution is given by the eigenvector with the second largest eigenvalue:

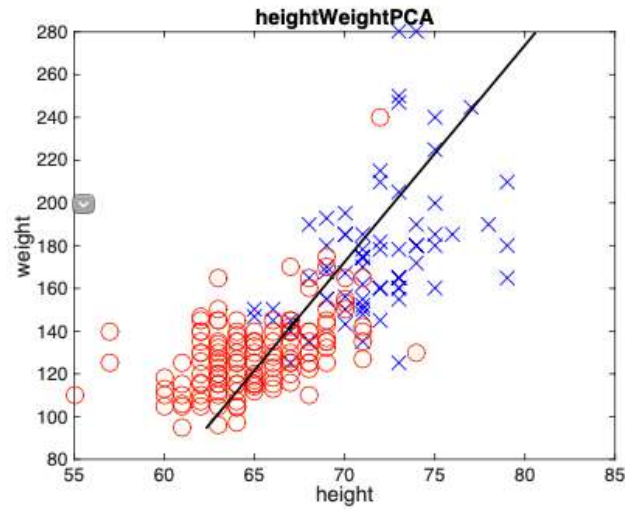
$$\hat{\Sigma} \mathbf{w}_2 = \lambda_2 \mathbf{w}_2 \quad (20.20)$$

The proof continues in this way to show that  $\hat{\mathbf{W}} = \mathbf{U}_L$ .

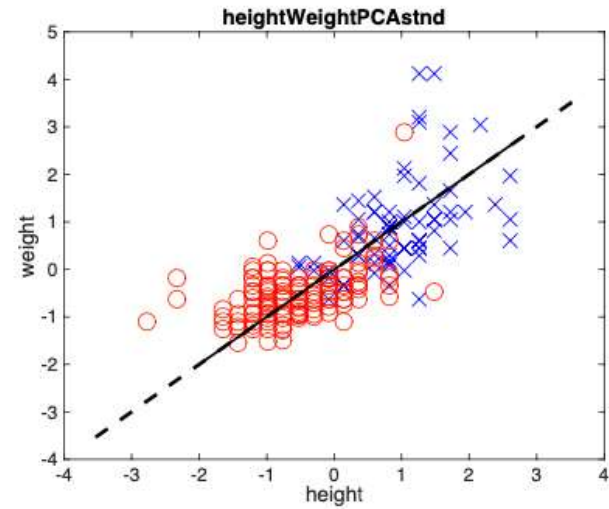


# Covariance matrix vs correlation matrix

- We have been working with the eigendecomposition of the covariance matrix. However, it is better to use the correlation matrix instead. The reason is that otherwise PCA can be “misled” by directions in which the variance is high merely because of the measurement scale. Figure 20.5 shows an example of this. On the left, we see that the vertical axis uses a larger range than the horizontal axis. This results in a first principal component that looks somewhat “unnatural”. On the right, we show the results of PCA after standardizing the data (which is equivalent to using the correlation matrix instead of the covariance matrix); the results look much better.



(a)



(b)

Figure 20.5: Effect of standardization on PCA applied to the height/weight dataset. (Red=female, blue=male.) Left: PCA of raw data. Right: PCA of standardized data. Generated by [pcaStandardization.ipynb](#).

# Dealing with high-dimensional data

We have presented PCA as the problem of finding the eigenvectors of the  $D \times D$  covariance matrix  $\mathbf{X}^T \mathbf{X}$ . If  $D > N$ , it is faster to work with the  $N \times N$  Gram matrix  $\mathbf{X} \mathbf{X}^T$ . We now show how to do this.

$\mathbf{X}$ :  $N \times D$

First, let  $\mathbf{U}$  be an orthogonal matrix containing the eigenvectors of  $\mathbf{X} \mathbf{X}^T$  with corresponding eigenvalues in  $\mathbf{\Lambda}$ . By definition we have  $(\mathbf{X} \mathbf{X}^T) \mathbf{U} = \mathbf{U} \mathbf{\Lambda}$ . Pre-multiplying by  $\mathbf{X}^T$  gives

$$(\mathbf{X}^T \mathbf{X})(\mathbf{X}^T \mathbf{U}) = (\mathbf{X}^T \mathbf{U}) \mathbf{\Lambda} \quad (20.21)$$

from which we see that the eigenvectors of  $\mathbf{X}^T \mathbf{X}$  are  $\mathbf{V} = \mathbf{X}^T \mathbf{U}$ , with eigenvalues given by  $\mathbf{\Lambda}$  as before. However, these eigenvectors are not normalized, since  $\|\mathbf{v}_j\|^2 = \mathbf{u}_j^T \mathbf{X} \mathbf{X}^T \mathbf{u}_j = \lambda_j \mathbf{u}_j^T \mathbf{u}_j = \lambda_j$ . The normalized eigenvectors are given by

$$\mathbf{V} = \mathbf{X}^T \mathbf{U} \mathbf{\Lambda}^{-\frac{1}{2}} \quad (20.22)$$

This provides an alternative way to compute the PCA basis. It also allows us to use the kernel trick, as we discuss in Section 20.4.6.

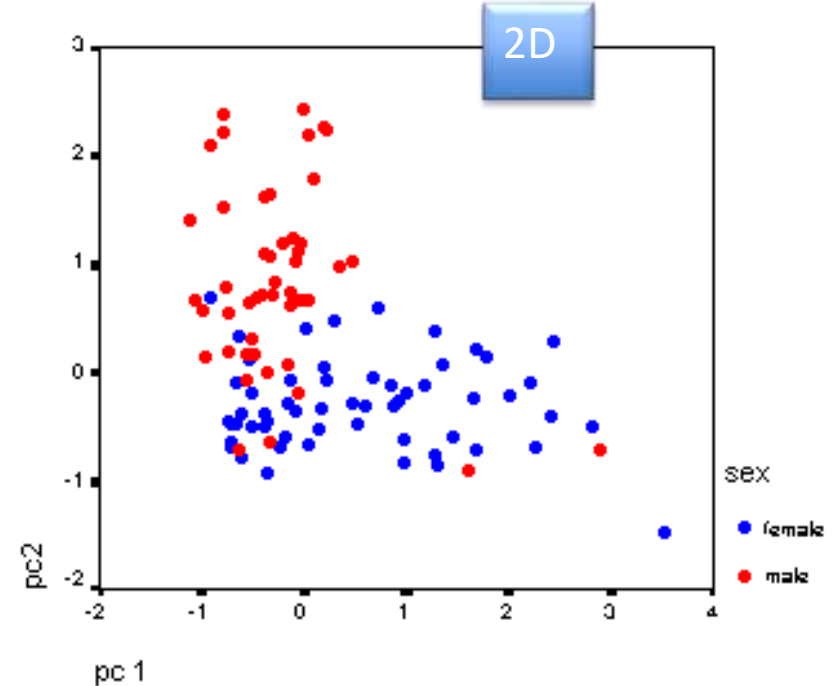
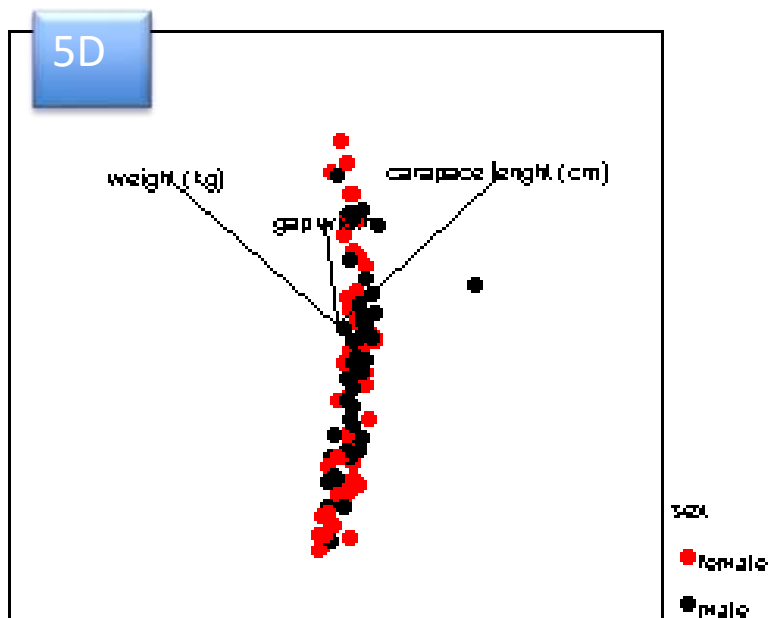
- Gram matrix captures the correlation between features.
- Given a real matrix  $\mathbf{X}$ , the matrix  $\mathbf{X}^T \mathbf{X}$  is the Gram matrix of the column vectors of  $\mathbf{X}$  and the matrix  $\mathbf{X} \mathbf{X}^T$  is the Gram matrix of the row vectors of  $\mathbf{X}$ .

# Additional remarks

- How many PC?
  - We want to retain as much information as possible using these components. So, higher is the explained variance, higher will be the information contained in those components.
  - We can compute each PC explains how much variance and then makes decision (still a parameter)
- Build analytics models
  - Transform training data into component space
  - Build our prediction model using new training data
  - Transform test set into the same component space and make prediction

# Example: The data matrix

case	ht ( $x_1$ )	wt( $x_2$ )	age( $x_3$ )	sbp( $x_4$ )	heart rate ( $x_5$ )
1	175	1225	25	117	56
2	156	1050	31	122	63
n	202	1350	58	154	67



Allow us choose small number of uncorrelated varies to perform machine learning tasks

# Dimensionality Reduction: PCA

Dimensions = 206



# Example of a data:

## Iris Flower Data Set

- Many of the exploratory data techniques are illustrated with the famous ***Iris Flower*** data set (a.k.a. “Iris”).
  - Available at the UCI Machine Learning Repository  
<http://www.ics.uci.edu/~mlearn/MLRepository.html>
  - From the statistician R.A. Fisher
  - Three flower types (**classes**):
    - Iris Setosa
    - Iris Versicolour
    - Iris Virginica
  - Four (**non-class**) attributes
    - Sepal width
    - Sepal length
    - Petal width
    - Petal length
  - Total number Instances = 150



[https://en.wikipedia.org/wiki/Iris\\_flower\\_data\\_set](https://en.wikipedia.org/wiki/Iris_flower_data_set)



# R Example using Iris data

- Iris

```
> head(iris)
  Sepal.Length Sepal.width Petal.Length Petal.width Species
1          5.1          3.5          1.4          0.2   setosa
2          4.9          3.0          1.4          0.2   setosa
3          4.7          3.2          1.3          0.2   setosa
4          4.6          3.1          1.5          0.2   setosa
5          5.0          3.6          1.4          0.2   setosa
6          5.4          3.9          1.7          0.4   setosa
```

- irisPCA<-**princomp**(iris[-5]) # Exclude Species and perform PCA

- summary(irisPCA)

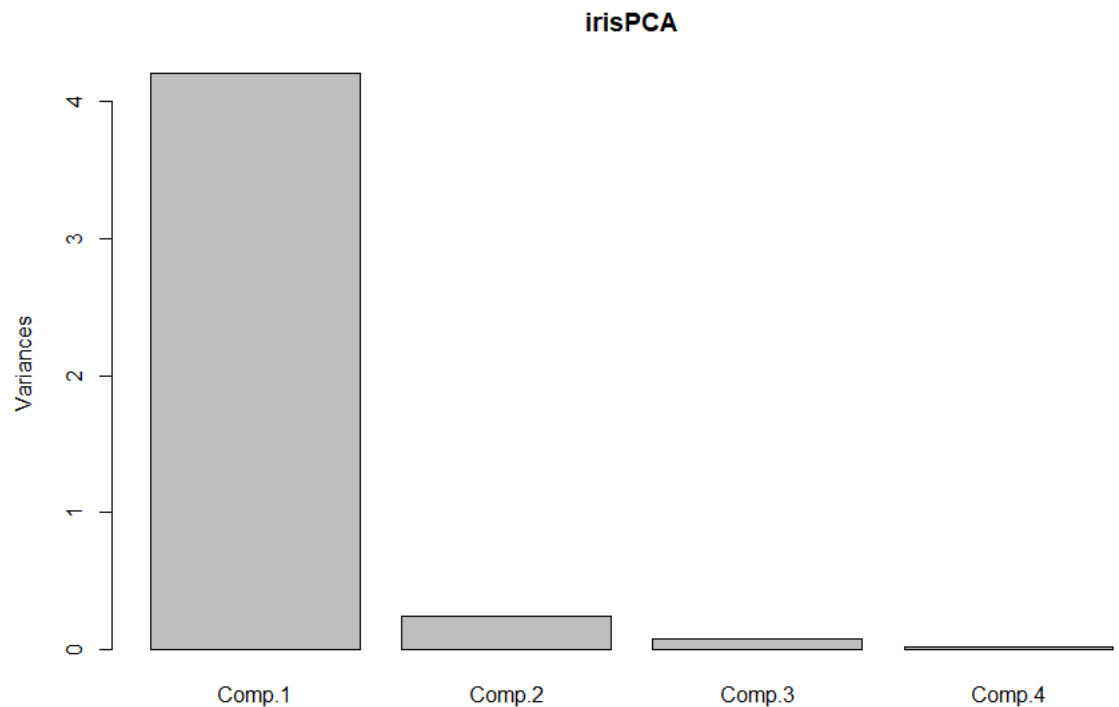
	PC1	PC2	PC3	PC4
> summary(irisPCA)				
Importance of components:				
	Comp.1	Comp.2	Comp.3	Comp.4
standard deviation	2.0494032	0.49097143	0.27872586	0.153870700
Proportion of Variance	0.9246187	0.05306648	0.01710261	0.005212184
Cumulative Proportion	0.9246187	0.97768521	0.99478782	1.000000000

92.5% of variation is explained by PC1 alone; 97.8% is explained by PC1 and PC2.



# Screen plot

- It shows the proportion of the total variation that is explained by each of the components. Perhaps 1 or 2 PC2 will be sufficient
- `screenplot(irisPCA)`





While **dimensionality reduction** is an important tool in machine learning/data mining, we must always be aware that it can *distort* the data in misleading ways.

Above is a two dimensional projection of an intrinsically three dimensional world....

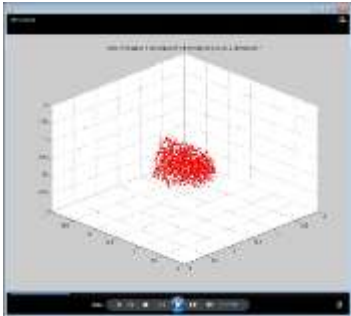


Original photographer unknown/  
See also [www.cs.gmu.edu/~jessica/DimReducDanger.htm](http://www.cs.gmu.edu/~jessica/DimReducDanger.htm)

© Eamonn Keogh

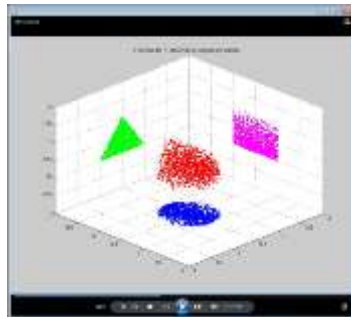
We may lose some important information when we perform feature selection.

A cloud of points in 3D

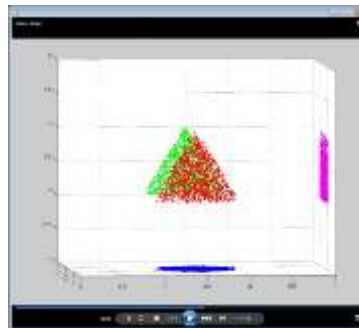


Can be projected into 2D

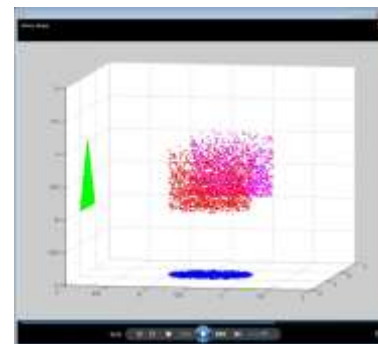
XY or XZ or YZ



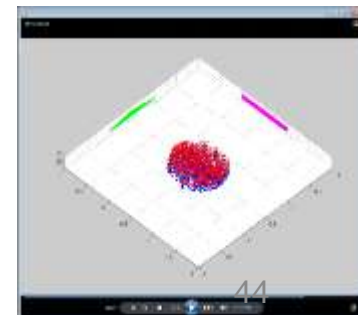
In 2D XZ we see a triangle



In 2D YZ we see a square



In 2D XY we see a circle



Screen dumps of a short video from  
[www.cs.gmu.edu/~jessica/DimReducDanger.htm](http://www.cs.gmu.edu/~jessica/DimReducDanger.htm)

# Singular Value Decomposition

The SVD is a factorization of a  $m \times n$  matrix into

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

where  $U$  is a  $m \times m$  orthogonal matrix,  $V^T$  is a  $n \times n$  orthogonal matrix and  $\Sigma$  is a  $m \times n$  diagonal matrix.

For a square matrix ( $m = n$ ):

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \dots$$

$$A = \begin{pmatrix} \vdots & \dots & \vdots \\ \mathbf{u}_1 & \dots & \mathbf{u}_n \\ \vdots & \dots & \vdots \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} \begin{pmatrix} \dots & \mathbf{v}_1^T & \dots \\ \vdots & \vdots & \vdots \\ \dots & \mathbf{v}_n^T & \dots \end{pmatrix}$$
$$A = \begin{pmatrix} \vdots & \dots & \vdots \\ \mathbf{u}_1 & \dots & \mathbf{u}_n \\ \vdots & \dots & \vdots \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} \begin{pmatrix} \vdots & \dots & \vdots \\ \mathbf{v}_1 & \dots & \mathbf{v}_n \\ \vdots & \dots & \vdots \end{pmatrix}^T$$



# Singular Value Decomposition

What happens when  $\mathbf{A}$  is not a square matrix?

1)  $m > n$

$$\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T = \underbrace{\begin{pmatrix} \vdots & \dots & \vdots & \dots & \vdots \\ \mathbf{u}_1 & \dots & \mathbf{u}_n & \dots & \mathbf{u}_m \\ \vdots & \dots & \vdots & \dots & \vdots \end{pmatrix}}_{m \times m} \underbrace{\begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_n & & \\ & & & 0 & \\ & & & \vdots & \\ & & & 0 & \end{pmatrix}}_{m \times n} \underbrace{\begin{pmatrix} \dots & \mathbf{v}_1^T & \dots \\ \vdots & \vdots & \vdots \\ \dots & \mathbf{v}_n^T & \dots \end{pmatrix}}_{n \times n}$$

We can instead re-write the above as:

$$\mathbf{A} = \mathbf{U}_R \mathbf{\Sigma}_R \mathbf{V}^T$$

Where  $\mathbf{U}_R$  is a  $m \times n$  matrix and  $\mathbf{\Sigma}_R$  is a  $n \times n$  matrix

# Singular Value Decomposition

2)  $n > m$

$$A = U \Sigma V^T = \underbrace{\begin{pmatrix} \vdots & \dots & \vdots \\ u_1 & \dots & u_m \\ \vdots & \dots & \vdots \end{pmatrix}}_{m \times m} \underbrace{\begin{pmatrix} \boxed{\begin{matrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \end{matrix}} & 0 & \dots \\ & \ddots & \\ & & 0 \end{pmatrix}}_{m \times n} \underbrace{\begin{pmatrix} \dots & \boxed{\begin{matrix} v_1^T \\ \vdots \\ v_m^T \end{matrix}} & \dots \\ \vdots & \vdots & \vdots \\ \dots & v_n^T & \dots \\ \vdots & \vdots & \vdots \\ \dots & v_n^T & \dots \end{pmatrix}}_{n \times n}$$

We can instead re-write the above as:

$$A = U \Sigma_R V_R^T$$

where  $V_R$  is a  $n \times m$  matrix and  $\Sigma_R$  is a  $m \times m$  matrix

In general:

$$A = U_R \Sigma_R V_R^T$$

$U_R$  is a  $m \times k$  matrix

$\Sigma_R$  is a  $k \times k$  matrix

$V_R$  is a  $n \times k$  matrix

$$k = \min(m, n)$$

# Singular Value Decomposition

Let's take a look at the product  $\mathbf{\Sigma}^T \mathbf{\Sigma}$ , where  $\mathbf{\Sigma}$  has the singular values of a  $\mathbf{A}$ , a  $m \times n$  matrix.

$$\begin{array}{c}
 \mathbf{\Sigma}^T \mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & 0 & \cdots \\ & \ddots & & \\ & & \sigma_n & \\ & & & \ddots \\ & & 0 & \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_n & \\ & & 0 & \\ & & \vdots & \\ & & 0 & \end{pmatrix} = \boxed{\begin{pmatrix} \sigma_1^2 & & & \\ & \ddots & & \\ & & \sigma_n^2 & \\ & & & \ddots \end{pmatrix}} \\
 m > n \quad \quad n \times m \quad \quad m \times n \quad \quad n \times n
 \end{array}$$

$$\begin{array}{c}
 \mathbf{\Sigma}^T \mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_m & \\ & & 0 & \\ & & \vdots & \\ & & 0 & \end{pmatrix} \begin{pmatrix} \sigma_1 & & 0 & & \\ & \ddots & & \ddots & \\ & & \sigma_m & & \\ & & & \ddots & \\ & & 0 & & \end{pmatrix} = \begin{pmatrix} \boxed{\begin{pmatrix} \sigma_1^2 & & & \\ & \ddots & & \\ & & \sigma_m^2 & \\ & & & \ddots \end{pmatrix}} & & 0 & & \\ & \ddots & & \ddots & \\ & & 0 & & 0 & \\ & & & \ddots & & \\ & & & & 0 & \end{pmatrix} \\
 n > m \quad \quad n \times m \quad \quad m \times n \quad \quad n \times n
 \end{array}$$



# Singular Value Decomposition

Assume  $\mathbf{A}$  with the singular value decomposition  $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$ . Let's take a look at the eigenpairs corresponding to  $\mathbf{A}^T \mathbf{A}$ :

$$\begin{aligned} \mathbf{A}^T \mathbf{A} &= (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T)^T (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T) \\ (\mathbf{V}^T)^T (\mathbf{\Sigma})^T \mathbf{U}^T (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T) &= \mathbf{V} \mathbf{\Sigma}^T \mathbf{U}^T \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T = \mathbf{V} \mathbf{\Sigma}^T \mathbf{\Sigma} \mathbf{V}^T \end{aligned}$$

Hence  $\mathbf{A}^T \mathbf{A} = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^T$

Recall that columns of  $\mathbf{V}$  are all linear independent (orthogonal matrix), then from diagonalization ( $\mathbf{B} = \mathbf{X} \mathbf{D} \mathbf{X}^{-1}$ ), we get:

- the columns of  $\mathbf{V}$  are the eigenvectors of the matrix  $\mathbf{A}^T \mathbf{A}$
- The diagonal entries of  $\mathbf{\Sigma}^2$  are the eigenvalues of  $\mathbf{A}^T \mathbf{A}$

Let's call  $\lambda$  the eigenvalues of  $\mathbf{A}^T \mathbf{A}$ , then  $\sigma_i^2 = \lambda_i$

# Singular Value Decomposition

In a similar way,

$$\begin{aligned} \mathbf{A}\mathbf{A}^T &= (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T) (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T)^T \\ (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T) (\mathbf{V}^T)^T (\mathbf{\Sigma})^T \mathbf{U}^T &= \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \mathbf{V} \mathbf{\Sigma}^T \mathbf{U}^T = \mathbf{U} \mathbf{\Sigma} \mathbf{\Sigma}^T \mathbf{U}^T \end{aligned}$$

Hence  $\mathbf{A}\mathbf{A}^T = \mathbf{U} \mathbf{\Sigma}^2 \mathbf{U}^T$

Recall that columns of  $\mathbf{U}$  are all linear independent (orthogonal matrices), then from diagonalization ( $\mathbf{B} = \mathbf{X}\mathbf{D}\mathbf{X}^{-1}$ ), we get:

- The columns of  $\mathbf{U}$  are the eigenvectors of the matrix  $\mathbf{A}\mathbf{A}^T$

# How can we compute an SVD of a matrix $A$ ?

1. Evaluate the  $n$  eigenvectors  $\mathbf{v}_i$  and eigenvalues  $\lambda_i$  of  $\mathbf{A}^T \mathbf{A}$
2. Make a matrix  $\mathbf{V}$  from the normalized vectors  $\mathbf{v}_i$ . The columns are called “right singular vectors”.

$$\mathbf{V} = \begin{pmatrix} \vdots & \dots & \vdots \\ \mathbf{v}_1 & \dots & \mathbf{v}_n \\ \vdots & \dots & \vdots \end{pmatrix}$$

3. Make a diagonal matrix from the square roots of the eigenvalues.

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} \quad \sigma_i = \sqrt{\lambda_i} \quad \text{and} \quad \sigma_1 \geq \sigma_2 \geq \sigma_3 \dots$$

4. Find  $\mathbf{U}$ :  $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \Rightarrow \mathbf{U} \mathbf{\Sigma} = \mathbf{A} \mathbf{V} \Rightarrow \mathbf{U} = \mathbf{A} \mathbf{V} \mathbf{\Sigma}^{-1}$ . The columns are called the “left singular vectors”.

# Singular Value Decomposition

Singular values cannot be negative since  $\mathbf{A}^T \mathbf{A}$  is a **positive semi-definite matrix** (for real matrices  $\mathbf{A}$ )

---

- A matrix is positive definite if  $\mathbf{x}^T \mathbf{B} \mathbf{x} > 0$  for  $\forall \mathbf{x} \neq \mathbf{0}$
- A matrix is positive semi-definite if  $\mathbf{x}^T \mathbf{B} \mathbf{x} \geq 0$  for  $\forall \mathbf{x} \neq \mathbf{0}$

- What do we know about the matrix  $\mathbf{A}^T \mathbf{A}$ ?

$$\mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} = (\mathbf{A} \mathbf{x})^T \mathbf{A} \mathbf{x} = \|\mathbf{A} \mathbf{x}\|_2^2 \geq 0$$

- Hence we know that  $\mathbf{A}^T \mathbf{A}$  is a positive semi-definite matrix
- A positive semi-definite matrix has non-negative eigenvalues

$$\mathbf{B} \mathbf{x} = \lambda \mathbf{x} \Rightarrow \mathbf{x}^T \mathbf{B} \mathbf{x} = \mathbf{x}^T \lambda \mathbf{x} = \lambda \|\mathbf{x}\|_2^2 \geq 0 \Rightarrow \lambda \geq 0$$

# Singular Value Decomposition

- The SVD is a factorization of a  $m \times n$  matrix into  $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$  where  $\mathbf{U}$  is a  $m \times m$  orthogonal matrix,  $\mathbf{V}^T$  is a  $n \times n$  orthogonal matrix and  $\mathbf{\Sigma}$  is a  $m \times n$  diagonal matrix.
- In reduced form:  $\mathbf{A} = \mathbf{U}_R \mathbf{\Sigma}_R \mathbf{V}_R^T$ , where  $\mathbf{U}_R$  is a  $m \times k$  matrix,  $\mathbf{\Sigma}_R$  is a  $k \times k$  matrix, and  $\mathbf{V}_R$  is a  $n \times k$  matrix, and  $k = \min(m, n)$ .
- The columns of  $\mathbf{V}$  are the eigenvectors of the matrix  $\mathbf{A}^T \mathbf{A}$ , denoted the right singular vectors.
- The columns of  $\mathbf{U}$  are the eigenvectors of the matrix  $\mathbf{A} \mathbf{A}^T$ , denoted the left singular vectors.
- The diagonal entries of  $\mathbf{\Sigma}^2$  are the eigenvalues of  $\mathbf{A}^T \mathbf{A}$ .  $\sigma_i = \sqrt{\lambda_i}$  are called the singular values.
- The singular values are always non-negative (since  $\mathbf{A}^T \mathbf{A}$  is a positive semi-definite matrix, the eigenvalues are always  $\lambda \geq 0$ )



# Low-Rank Approximation

Another way to write the SVD (assuming for now  $m > n$  for simplicity)

$$\begin{aligned}
 A &= \begin{pmatrix} \vdots & & \vdots \\ \mathbf{u}_1 & & \vdots \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & & 0 \\ & & \vdots \\ & & 0 \end{pmatrix} \begin{pmatrix} \dots & \mathbf{v}_1^T & \dots \\ \vdots & \vdots & \vdots \\ \dots & \mathbf{v}_n^T & \dots \end{pmatrix} \\
 &= \begin{pmatrix} \vdots & \dots & \vdots \\ \mathbf{u}_1 & \dots & \mathbf{u}_n \\ \vdots & \dots & \vdots \end{pmatrix} \begin{pmatrix} \dots & \sigma_1 \mathbf{v}_1^T & \dots \\ \vdots & \vdots & \vdots \\ \dots & \sigma_n \mathbf{v}_n^T & \dots \end{pmatrix} \\
 &= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \dots + \sigma_n \mathbf{u}_n \mathbf{v}_n^T
 \end{aligned}$$

The SVD writes the matrix  $A$  as a sum of outer products (of left and right singular vectors).

# Low-Rank Approximation

The best **rank- $k$**  approximation for a  $m \times n$  matrix  $\mathbf{A}$ , (where  $k \leq \min(m, n)$ ) is the one that minimizes the following problem:

$$\begin{aligned} \min_{\mathbf{A}_k} \quad & \|\mathbf{A} - \mathbf{A}_k\| \\ \text{such that} \quad & \text{rank}(\mathbf{A}_k) \leq k. \end{aligned}$$

When using the induced 2-norm, the best **rank- $k$**  approximation is given by:

$$\mathbf{A}_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T$$

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \dots \geq 0$$

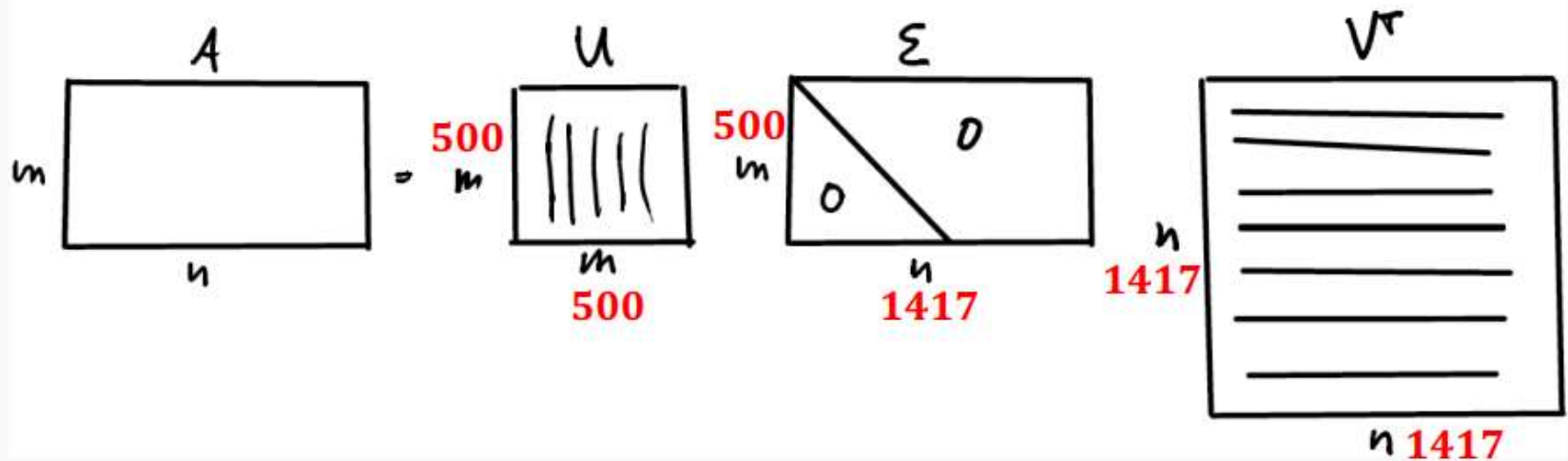
Note that  $\text{rank}(\mathbf{A}) = n$  and  $\text{rank}(\mathbf{A}_k) = k$  and the norm of the difference between the matrix and its approximation is

$$\|\mathbf{A} - \mathbf{A}_k\|_2 = \|\sigma_{k+1} \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T + \sigma_{k+2} \mathbf{u}_{k+2} \mathbf{v}_{k+2}^T + \cdots + \sigma_n \mathbf{u}_n \mathbf{v}_n^T\|_2 = \sigma_{k+1}$$

# Image compression



1417





# Image compression



Image using rank-50 approximation



# Summary

- What Kinds of Data to be Mined
- Data Attribute types
  - Nominal, Ordinal, Interval, Ratio
- Types of Data Sets
- Characteristics of Structured Data
- Data Preprocessing
  - Data Cleaning, Aggregation, Sampling, Dimensionality Reduction, Feature subset selection, Discretization and Binarization, Attribute Transformation. They *provide some options* for us to get better results.