Dimensionality Reduction

COMP5318/COMP4318 Machine Learning and Data Mining

semester 1, 2023, week 6b

Irena Koprinska

Reference: Müller and Guido ch. 3.4.1: 142-157, Geron ch.8: 219-226, Witten ch.8.3: 305-307

Outline

- Motivation
- Principal component analysis (PCA)
- Singular value decomposition
- Examples
  - PCA for feature extraction
  - PCA for compression

Motivation

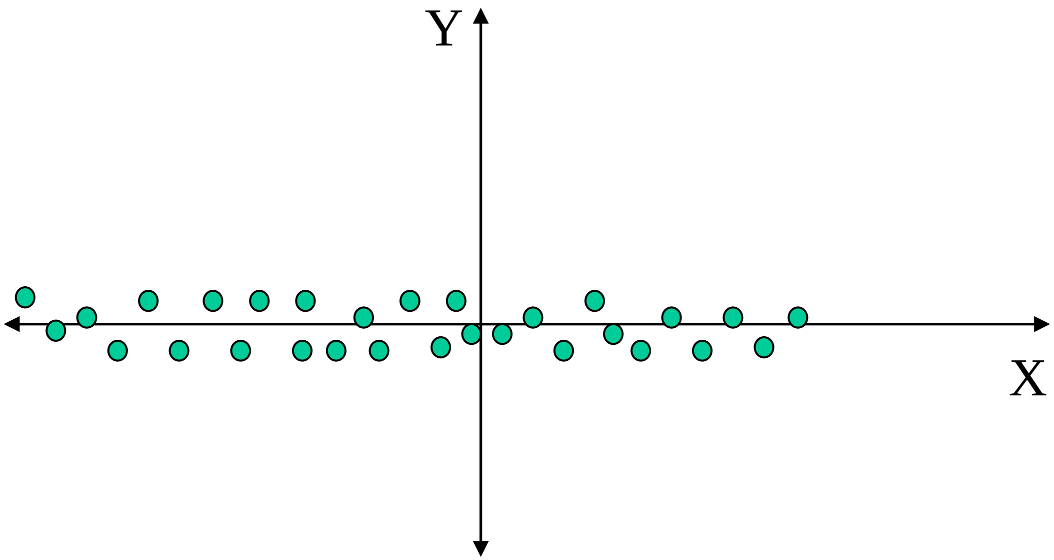
- Some ML problems involve thousands of features
- Problems with high dimensional data
  - Slower training
  - Unreliable classification – examples are far away from each other; high dimensional data is very sparse
  - Overfitting is more likely in high dimensional data
  - Building interpretable models is not possible - we would like to build compact and easier to interpret classification models
  - Visualizing – humans can only interpret low dimensional data, e.g. max 3 dimensions
  - Not all features are important – it is desirable to find a smaller set of features that are necessary and sufficient for good classification
- Dimensionality reduction removes redundant and highly correlated features and reduces noise in the data

Principal Component Analysis

- PCA is the most popular dimensionality reduction method
- It is often called a feature projection method
- The main idea is to find a new set of dimensions (axes) and project the data into it
  - The dimensionality of the new space is smaller than the dimensionality of the original space
  - The new axes capture the essence of the data (the variability of the data)
- The resulting dataset (the projection) can be used as an input to train a ML algorithm
- In summary, we construct new features; the number of new features is smaller than the number of the original features

Example 1

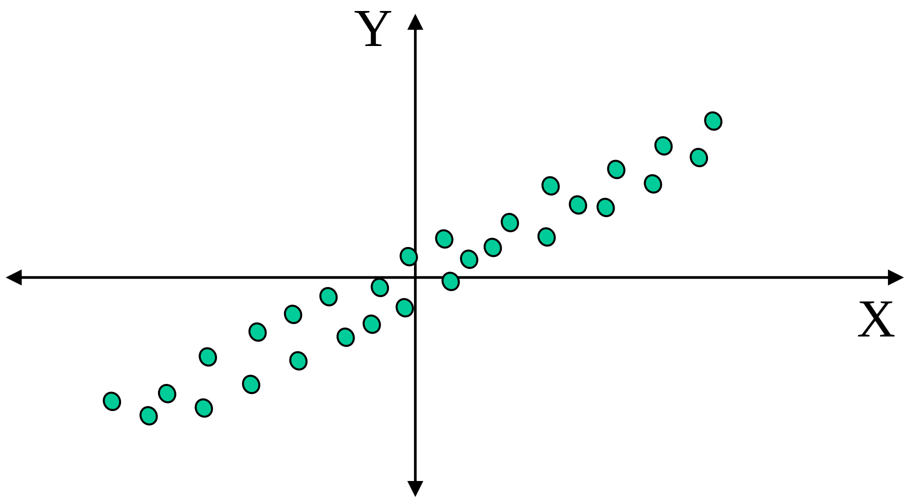
- Where is the maximum variability of the data – along which axis – X or Y?



Answer: X

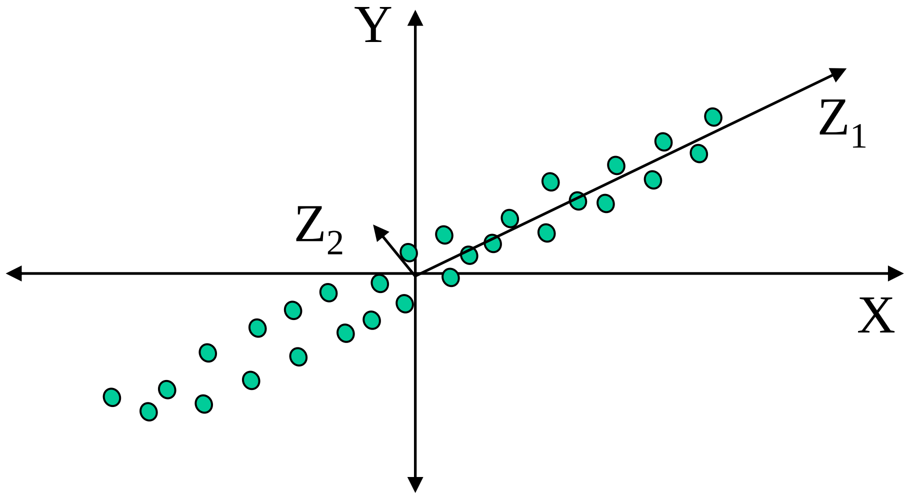
Example 2

- Where is the maximum variability of the data? Is it along X or Y or another axis?



Example 2 - Answer

- Where is the maximum variability of the data? Is it along X or Y or another axis?

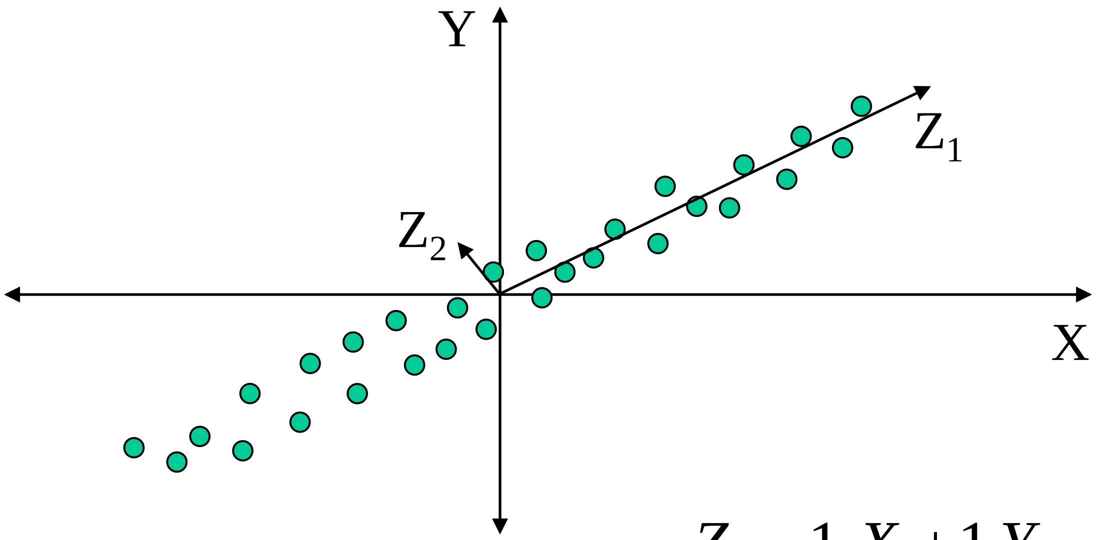


Answer: Along another axis - Z_1

Max variability along Z_1, some variability along Z_2

Example 2 again

- Two axes: Z_1 and Z_2
- $Var(Z_1) > Var(Z_2)$
- Z_1 and Z_2 are linear combination of X and Y



$$Z_1 = 1.X + 1.Y$$

$$Z_2 = -1.X + 1.Y$$

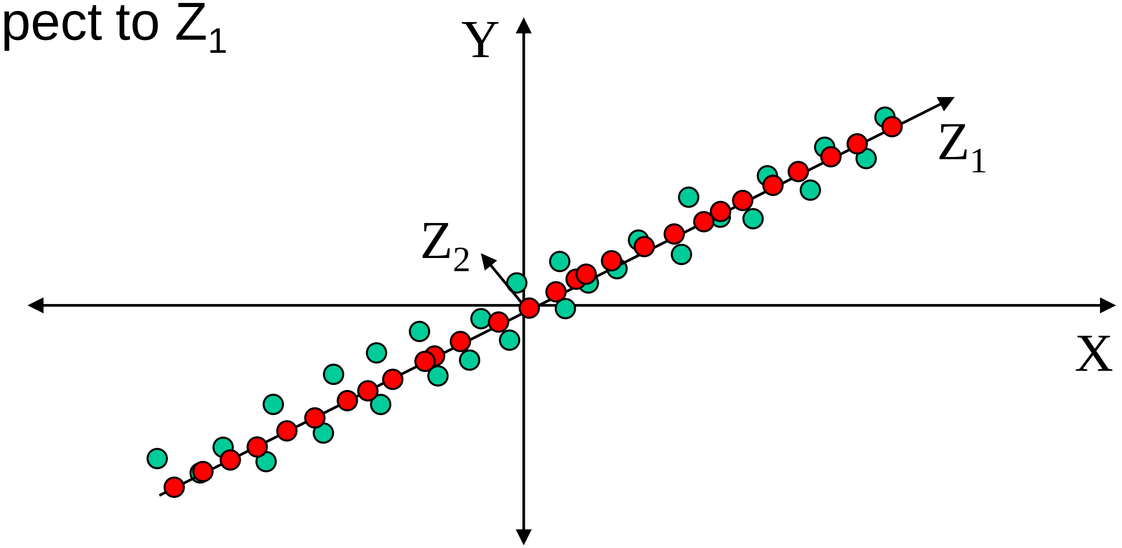
PCA – main idea

- Given: *N* examples with dimensionality *m* (i.e. *m* features)
- Find: *m* new axes $Z_1,\ldots, Z_m$ orthogonal to each other such that
  $Var(Z_1) > Var(Z_2) \ldots > Var(Z_m)$
- Z1,…, Zm are called principal components

- The principal components are vectors that define a new coordinate system
- They are ordered based on how much variance they capture
  - The first axis goes in the direction of the highest variance in the data
  - The second axis is orthogonal to the first one and goes in the direction of the second highest variance
  - The third one is orthogonal to both the first and second and goes in the direction of the third highest variance, and so on

PCA – how to reduce data dimensionality?

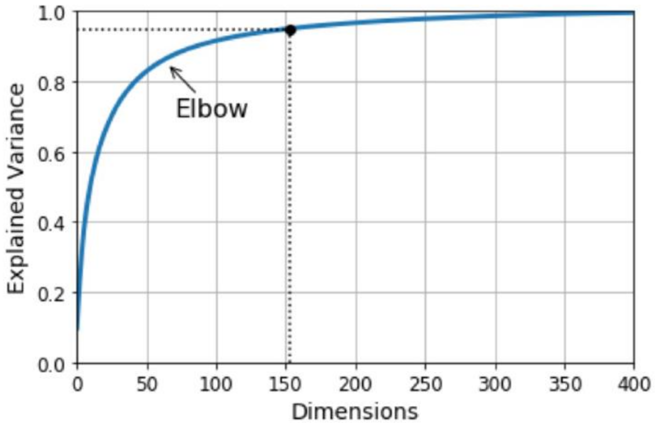
- Select the *k* largest principal components $Z_1, Z_2, \ldots Z_k$ and project our data points on them ($k<m$)
- For our 2-dim data in Example 2, we can select only Z_1 ->1-dim data
- The red points are projections of the original green points on the first principal component Z_1
- To describe the red points we need only one coordinate instead of two – the coordinate with respect to Z_1



How many principal components (dimensions) to select?

- Method 1: Set min % of variance that should be preserved, e.g. 95%
  - Choose *k* such that $Z_1, Z_2, \ldots Z_k$ capture 95% of the variance

- Method 2: (Elbow method)
  - Plot the number of dimensions as a function of variance
  - There is usually an elbow in the curve where the variance stops growing fast

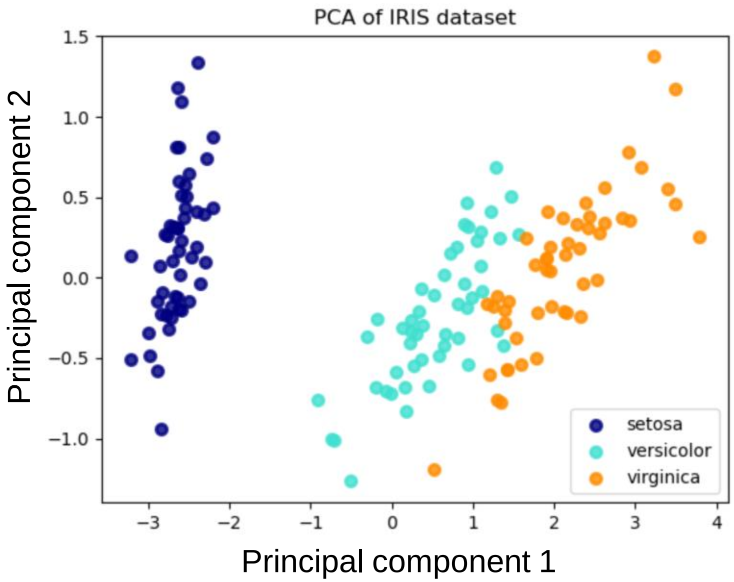


- 95% variance is at 153 dimensions
- Elbow (subjective) - e.g. 100 dimensions

PCA on iris data

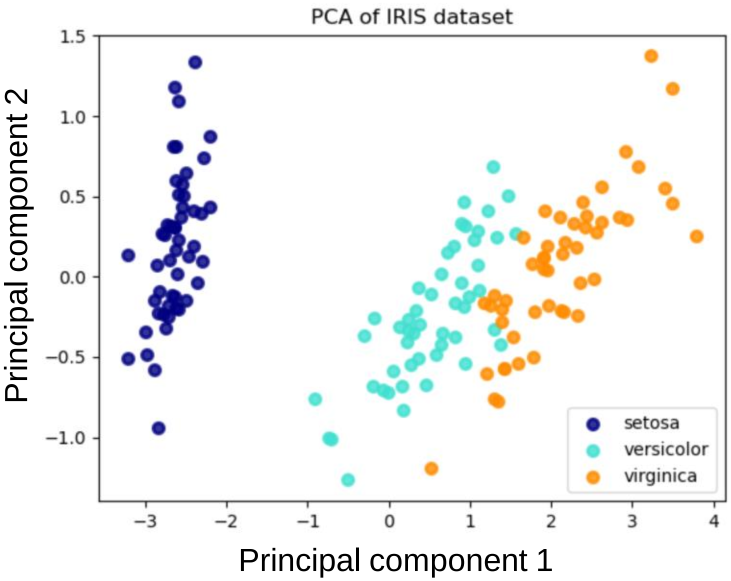
- 150 flowers, 3 classes (setosa, versicolor and virginica)
- 4 original features; 2 new features using PCA
- PC1 captures 92.5% of the variance, PC2 - 5.3%

Image from https://archive.ics.uci.edu/ml/datasets/iris



- setosa is well separated from the other 2 classes
- versicolor and virginica are close to each other but also relatively well separated

PCA on iris data (2)



- The dimensionality of the data is reduced from 4 to 2 while preserving the variance of the data
- Most of the variance can be captured by a small fraction of the original dimensions!

- We can use the new features to train a classifier (k nearest neighbor, NB etc.) - the accuracy may even improve if the new representation is better
- => This is an application of PCA for feature extraction – find a lower dimensional representation that is better suited than the original representation

How to find the principal components?

- Using a standard matrix factorization method, called Singular Value Decomposition (SVD)
- Theorem: Any n x m matrix X ($n \geq m$) can be written as the product of 3 matrices

$$X = U \times \Lambda \times V^T$$

$\mathbf{U}$ - n x m orthogonal matrix

$\mathbf{V}^{\mathbf{T}}$ - the transpose of an m x m orthogonal matrix

$\mathbf{\Lambda}$ - m x m diagonal matrix containing the singular values (positive or zero elements)

- $\mathbf{V}$ defines the new set of axes (principal components)
  - Provides important information about the variance in data – the 1st axis goes in the direction with highest variance, 2nd – 2nd highest variance and so on
- $\mathbf{X}$ is the original data
- $\mathbf{U}$ is the transformed data, i.e. the i-th row of $\mathbf{U}$ contains the coordinates of the i-th row of $\mathbf{X}$ in the new coordinate system

Data reduction using SVD

- $\mathbf{X}$ can be re-written as:

$$X = \lambda_1 u_1 v_1^T + \lambda_2 u_2 v_2^T + \cdots \lambda_m u_m v_m^T$$

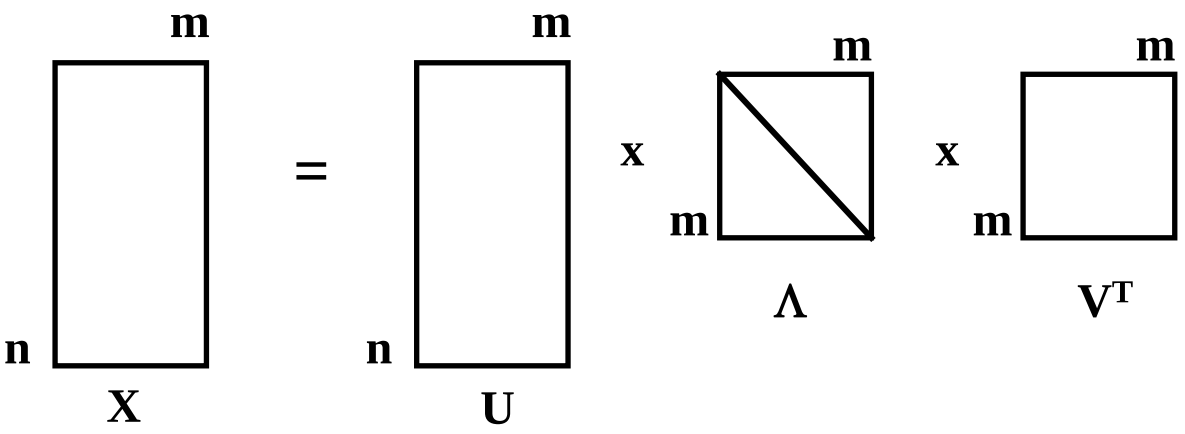
where λ are sorted in decreasing order

- Data reduction comes from taking only the first *k* components (*k*<*m*)

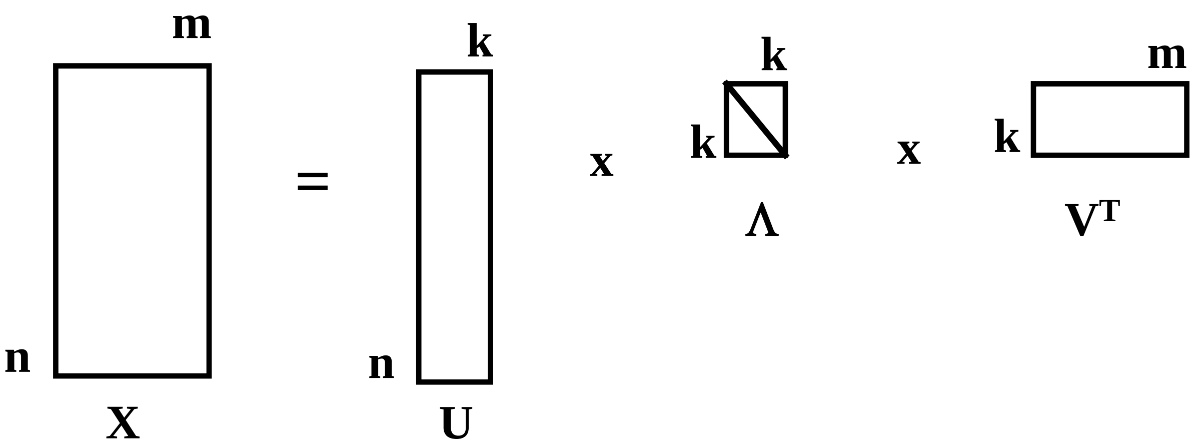
$$X_{reduced} = \lambda_1 u_1 v_1^T + \lambda_2 u_2 v_2^T + \cdots \lambda_k u_k v_k^T$$

- => The size of the data can be reduced by eliminating the weaker components (the ones with low variance)
- Using only the strongest components, it is possible to get a good approximation of the original data

Without data reduction:



With data reduction:



SVD example

original data

$$\mathbf{X} = \begin{pmatrix} -149 & -50 & -154 \\ 534 & 181 & 542 \\ -29 & -10 & -27 \end{pmatrix}$$

transformed data (the projection)

$$\mathbf{U} = \begin{pmatrix} -0.27 & -0.68 & 0.68 \\ 0.96 & -0.16 & 0.22 \\ -0.05 & 0.72 & 0.70 \end{pmatrix}$$

singular values

$$\Lambda = \begin{pmatrix} 818 & 0 & 0 \\ 0 & 2.48 & 0 \\ 0 & 0 & 0.003 \end{pmatrix}$$

new set of axes (principal components)

$$\mathbf{V} = \begin{pmatrix} 0.68 & -0.67 & 0.3 \\ 0.23 & -0.19 & -0.95 \\ 0.69 & 0.72 & 0.02 \end{pmatrix}$$

- You can verify that:

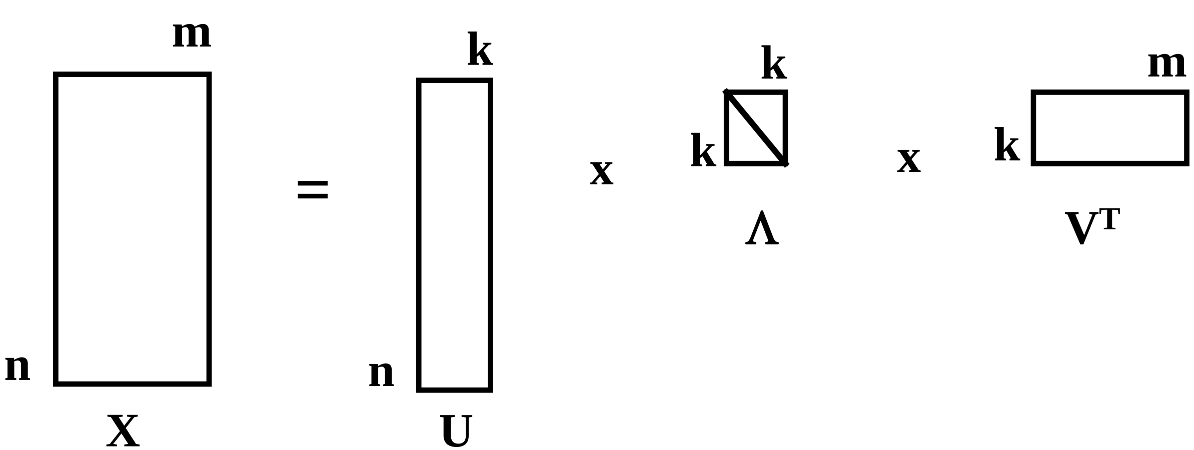
$$\mathbf{X} = \mathbf{U} \times \mathbf{\Lambda} \times \mathbf{V}^{\mathbf{T}}$$

- Most of the variance is captured in the first component
- => the original 3-dim data **X** can be reduced to 1-dim data in the new feature space = first column of **U**)

SVD for compression

- Consider image compression, e.g. grayscale image
- Uncompressed image: *n* x *m* pixels => we need to store *n* x *m* int numbers
- Compressed image using the first *k* components – we need to store:
  - *k* singular values from the $\mathbf{\Lambda}$ matrix
  - The first *k* columns of the $\mathbf{U}$ matrix (*k* columns x *n* rows)
  - The first *k* columns of the $\mathbf{V}$ matrix (*k* columns x *m* rows)
  - Total: *k* x (1+*n*+*m*)

With data reduction:

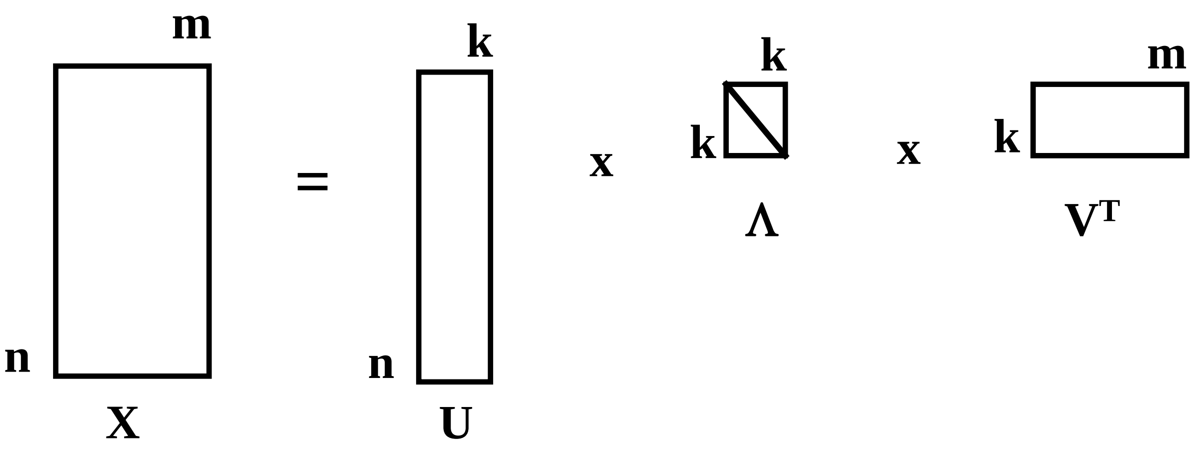


Compression ratio

- Compression ratio = after compression/ before compression

$$r = \frac{k(1+n+m)}{n \times m}$$

- For *n* >> *m* >*k*, this ratio is approximately *k*/*m*
- e.g. if *m* = 365 and *k* = 10 => *r* = 0.027 = 2.7%



Feature Extraction Example

PCA for feature extraction in images – face recognition

- Example from Müller and Guido, ch. 3.4.1
- Labeled Faces in the Wild (LFW) dataset: http://vis-www.cs.umass.edu/lfw/
- Contains images of celebrities – politicians, singers, actors, athletes, etc.
- 3,023 images of 62 different people; 1 image = 87 x 65 pixels

Some images from the LFW dataset

Face recognition – possible solutions

- Task: Determine if a new image (previously unseen) belongs to a known person from the database
- Applications: photo collection, social media, security

- Possible solution:
  - Build a separate classifier for each person
  - However, there are many different people in face datasets and very few images of the same person => many classifiers with very few examples per class – hard to train. Also, adding new face images for an existing person will require re-training of the classifier.
- Another solution: use nearest neighbor classifier
  - Look for the most similar face images to the new example
  - Can work with only 1 image per person
- Let's try 1-nearest neighbor and see how it will work!

Face recognition using nearest neighbor

- 62-class classification problem (62 people)
- Take max 50 images per person (some people have many)
- 5,655 features (87 x 65 pixels) - raw pixel representation

- 1-nearest neighbor: 27% accuracy on test set
- Not bad, better than a random guess (1/62=1.6% accuracy)

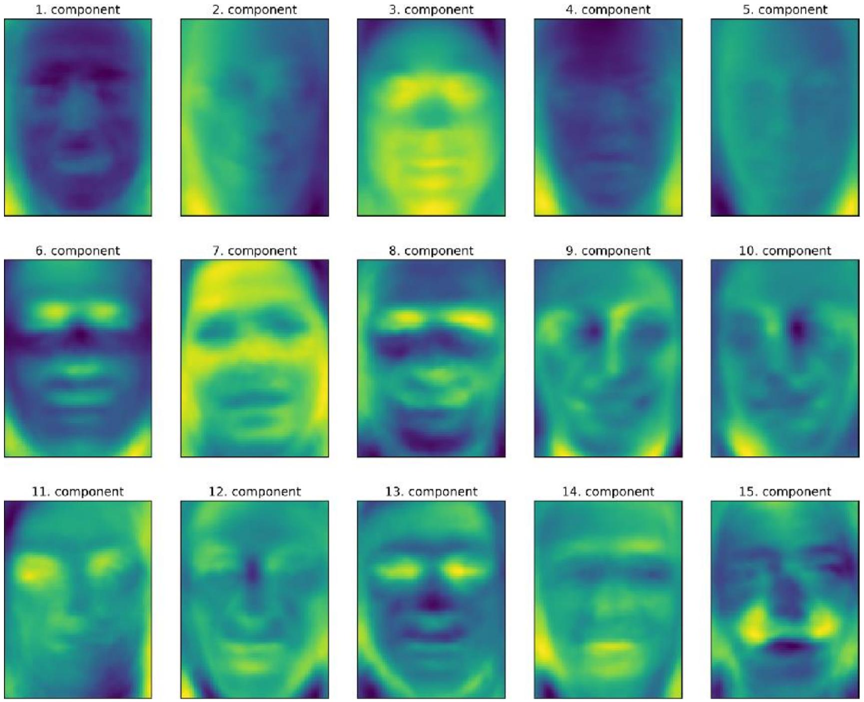
- We use the pixel representation and computed distance between grayscale values at the same position
- Not a good way to measure similarity between faces – hard to capture facial features; sensitive to shifts - shift in 1 pixel to the right -> big change

- Let’s try PCA to obtain a different representation!

Face recognition using PCA and nearest neighbor

- Using PCA with 100 features (the first 100 principal components)
- Accuracy on test set: 36% - improvement!
- => PCA provided a better representation

- We can also visualize the principal components:



- We can try to interpret which aspects of the face image are captured by the PCs (this is not always possible)
- It seems that PC1 encodes the contrast between the face and background, PC2 encodes the difference in lighting between the right and left part of the face, etc.

Image Compression Examples

Image compression example 1

- Example by Jonathan Bernick
- Image of rocks photographed by the Sojourner robot on Mars

- Before compression: 256 × 264 pixels grayscale bitmap
- => X is 256 × 264 matrix (i.e. contains 67,584 integer numbers from 0 to 255 that we need to store)

- After SVD compression, k=81 was selected based on captured variance
- => X= 81 × (1+256+264) = 42,201 numbers
- => 62% compression ratio (new/old storage)

Yogi

Rank:81, 62.4% data

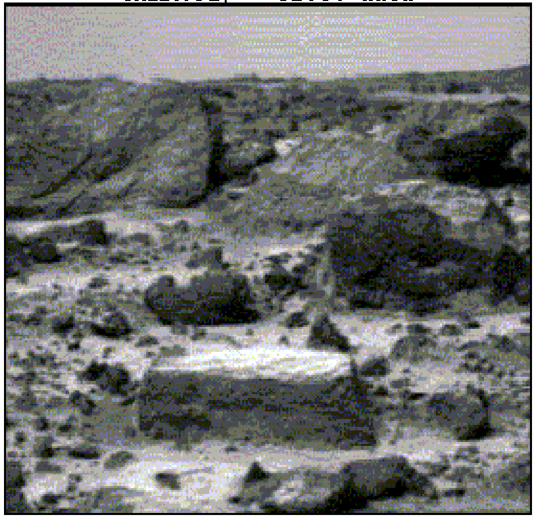
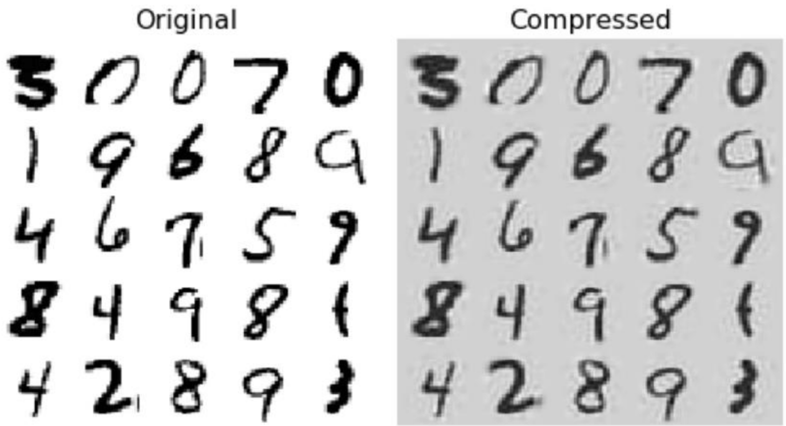


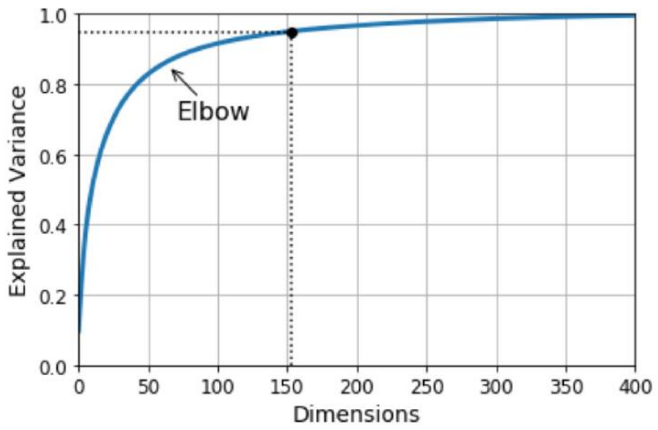
Image compression example 2

- MNIST dataset – contains hand-written digits
- http://yann.lecun.com/exdb/mnist/

- Example from Geron, ch.8; also in the tutorial exercises

- Original: 784 features
- PCA compressed: 153 features (95% preserved variance)





Summary

- PCA is a method for dimensionality reduction
- It is an unsupervised method – it doesn't use the class information
- It projects the data into a lower dimensional space that still captures the important information
- The new axes are ordered based on how much variance they capture
  - The first axis goes in the direction of the highest variance in the data
  - The second axis is orthogonal to the first one and goes in the direction of the second highest variance and so on
- The new axes are called principal components and represent patterns in data
- The data dimensionality is reduced by eliminating the weakest axes (the ones with low variance) => we use only the first principal components
- The resulting dataset (the projection) can be used as an input to train a ML algorithm
- PCA can also be used for compression