Unsupervised Learning

PCA and PCoA

Richard Röttger

University of Southern Denmark

February 8, 2017

Objectives for Today

- Last lecture, we have learned about density estimators
- They only work in low-dimensional cases satisfactory
- This lecture, we will tackle the problem of the dimensionality
- More specifically:
  - Curse of dimensionality
  - PCA (Principal Component Analysis)
  - Multidimensional Scaling
- The introduction is again based on Ricardo Gutierrez-Osuna's class on Pattern Analysis

The Curse of Dimensionality

The curse of dimensionality

- A term coined by Bellman in 1961
- Refers to the problems associated with multivariate data analysis as the dimensionality increases

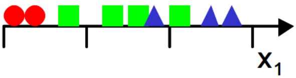
Consider a 3-class pattern recognition problem

- A simple approach would be to
  - Divide the feature space into uniform bins
  - Compute the ratio of examples for each class at each bin and,
  - For a new example, choose the predominant class in its bin
- In our toy problem we decide to start with one single feature and divide the real line into 3 segments
- After doing this, we notice that there exists too much overlap among the classes, so we decide to incorporate a second feature to try and improve separability

The Curse of Dimensionality

The curse of dimensionality

- A term coined by Bellman in 1961
- Refers to the problems associated with multivariate data analysis as the dimensionality increases

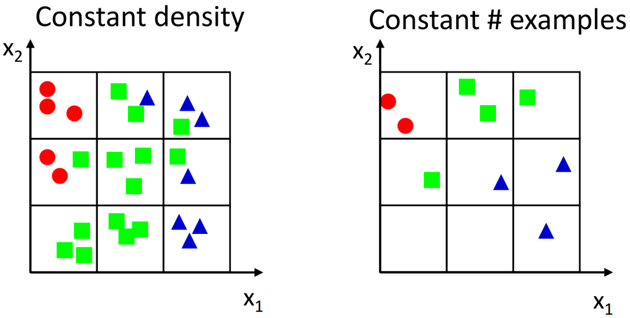


Consider a 3-class pattern recognition problem

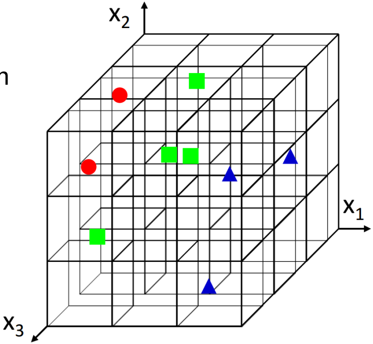
- A simple approach would be to
  - Divide the feature space into uniform bins
  - Compute the ratio of examples for each class at each bin and,
  - For a new example, choose the predominant class in its bin
- In our toy problem we decide to start with one single feature and divide the real line into 3 segments
- After doing this, we notice that there exists too much overlap among the classes, so we decide to incorporate a second feature to try and improve separability

The Curse of Dimensionality

- Preserving the granularity of each axis results $3^2 = 9$ bins
- 2 possibilities: maintain the density or maintain the number of examples?
  - Density: increases the number of examples from 9 to 27
  - Examples: results in a 2D scatter plot that is very sparse



More than two Dimensions

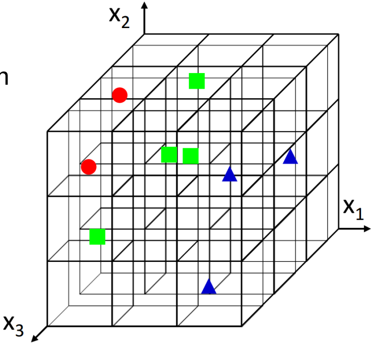


- Moving to three features makes the problem worse:
  - The number of bins grows to $3^3 = 27$
  - Maintaining the density would require 81 samples
  - For the same number of examples, the 3D scatter plot is almost empty
- Obviously, our approach to divide the sample space into equally spaced bins was quite inefficient
- There are other approaches that are much less susceptible to the curse of dimensionality, but the problem still exists

How do we beat the curse of dimensionality?

- In practice, our number of samples and dimensions is fixed
- This is only to demonstrate the problems with dimensionality
- Bottom-line: Reducing the dimensionality can be beneficial

More than two Dimensions



- Moving to three features makes the problem worse:
  - The number of bins grows to $3^3 = 27$
  - Maintaining the density would require 81 samples
  - For the same number of examples, the 3D scatter plot is almost empty
- Obviously, our approach to divide the sample space into equally spaced bins was quite inefficient
- There are other approaches that are much less susceptible to the curse of dimensionality, but the problem still exists

How do we beat the curse of dimensionality?

- In practice, our number of samples and dimensions is fixed
- This is only to demonstrate the problems with dimensionality
- Bottom-line: Reducing the dimensionality can be beneficial

Implications

- Maintaining density requires exponential growth in the sample size
- Exponential growth in the complexity of the target function (a density estimate) with increasing dimensionality
  - "A function defined in high-dimensional space is likely to be much more complex than a function defined in a lower-dimensional space, and those complications are harder to discern" - J. Friedman
  - This means that, in order to learn it well, a more complex target function requires denser sample points!
- What to do if it ain't Gaussian?
  - For 1D a large number of density functions can be found in textbooks, but for high-dimensions only the multivariate Gaussian density is available.
  - Moreover, for large d the Gaussian can only be handled in a simplified form!
- Humans have an extraordinary capacity to discern patterns and clusters in 1D, 2D and 3D, but these capabilities break down for $d \geq 4$

Lower-Dimensional Projections
Introduction

A new approach

- The methods we have seen are only feasible for univariate or bivariate data
- For multivariate data, some tricks exist but normally the methods lose a lot of their power
- $\Rightarrow$ We need a different way to cope with multivariate data

Dimensionality reduction

Two approaches are available to reduce dimensionality

- Feature extraction creating a subset of new features by combinations of the existing features
- Feature selection choosing a subset of all the features

$$\begin{bmatrix} x_1 \\ x_2 \\ \\ \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \\ x_{i_M} \end{bmatrix} \qquad f\left(\begin{bmatrix} x_1 \\ x_2 \\ \\ \\ x_N \end{bmatrix}\right) : \begin{bmatrix} x_1 \\ x_2 \\ \\ \\ x_N \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \\ y_m \end{bmatrix}$$

Feature Extraction

Given a vector x in feature space $\mathbb{R}^N$ find a mapping function $y = f(x) : \mathbb{R}^N \rightarrow \mathbb{R}^M$ with $M < N$ such that the transformed feature vector $y \in \mathbb{R}$ preserves most of the information

Dimensionality reduction

Two approaches are available to reduce dimensionality

- Feature extraction creating a subset of new features by combinations of the existing features
- Feature selection choosing a subset of all the features

$$\begin{bmatrix} x_1 \\ x_2 \\ \\ x_N \end{bmatrix} \to \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \\ x_{i_M} \end{bmatrix} \qquad f\left(\begin{bmatrix} x_1 \\ x_2 \\ \\ x_N \end{bmatrix}\right) : \begin{bmatrix} x_1 \\ x_2 \\ \\ x_N \end{bmatrix} \to \begin{bmatrix} y_1 \\ y_2 \\ \\ y_m \end{bmatrix}$$

Feature Extraction

Given a vector x in feature space $\mathbb{R}^N$ find a mapping function $y = f(x) : \mathbb{R}^N \to \mathbb{R}^M$ with $M < N$ such that the transformed feature vector $y \in \mathbb{R}$ preserves most of the information

Linear Dimensionality Reduction

- In general, the optimal mapping $y = f(x)$ will be a non-linear function
    - However, there is no systematic way to generate non-linear transforms
    - The selection of a particular subset of transforms is problem dependent
- For these reasons, feature extraction is commonly based on linear transforms, of the form $y = Wx$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{1N} \\ \vdots & \ddots & \vdots \\ w_{M1} & \cdots & w_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_N \end{bmatrix}$$

Lower-Dimensional Projections

Principal Component Analysis

Introduction to PCA

- PCA is a very complex and large topic which can basically fill entire lecture series
- Furthermore, there are many interpretations and different applications for a PCA[1]
- Here, we limit ourselfs to the usage of PCA in clustering:
  - Project data to a lower dimensional space
  - Hopefully provides a better means for visual inspection

Generally Speaking

The task of a PCA is to perform a dimensionality reduction in such a way that most of the variance in the original data is preserved

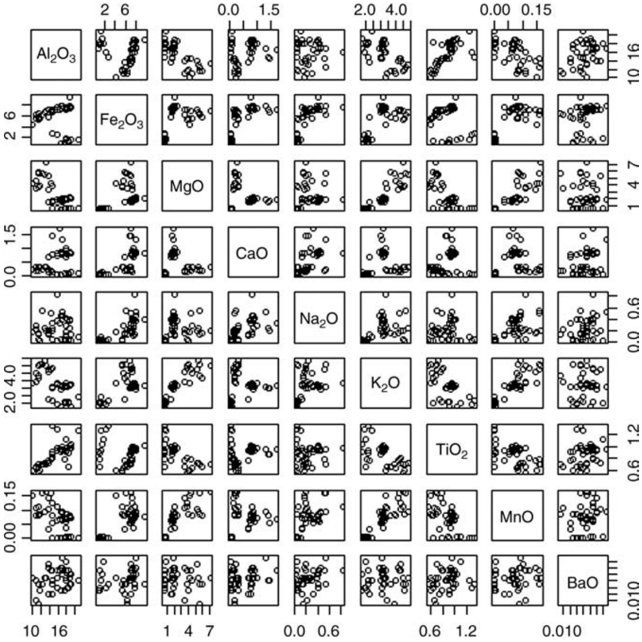
[1] see for example: https://liorpachter.wordpress.com/2014/05/26/what-is-principal-component-analysis/

Example: Pottery Data

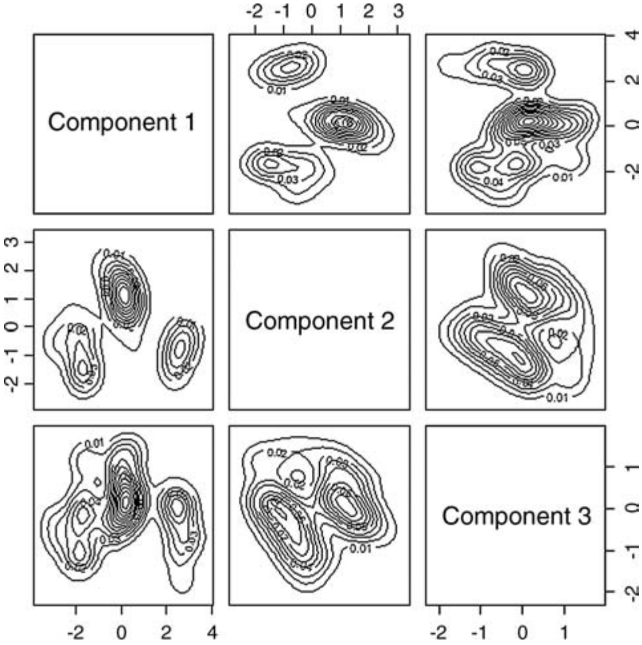
Sample number	Chemical component								
	Al_2O_3	Fe_2O_3	MgO	CaO	Na_2O	K_2O	TiO_2	MnO	BaO
1	18.8	9.52	2.00	0.79	0.40	3.20	1.01	0.077	0.015
2	16.9	7.33	1.65	0.84	0.40	3.05	0.99	0.067	0.018
3	18.2	7.64	1.82	0.77	0.40	3.07	0.98	0.087	0.014
4	16.9	7.29	1.56	0.76	0.40	3.05	1.00	0.063	0.019
5	17.8	7.24	1.83	0.92	0.43	3.12	0.93	0.061	0.019
6	18.8	7.45	2.06	0.87	0.25	3.26	0.98	0.072	0.017
7	16.5	7.05	1.81	1.73	0.33	3.20	0.95	0.066	0.019
8	18.0	7.42	2.06	1.00	0.28	3.37	0.96	0.072	0.017
9	15.8	7.15	1.62	0.71	0.38	3.25	0.93	0.062	0.017

- The data show the chemical compounds of ancient pottery
- The table is very unintuitive
- Let's look at the scatter-plots

Pottery Data: Scatter-Plots



Pottery Data: PCA



How it works?

Observation

- In the example some structure became suddenly visible
- Now the data suggest to contain 3 clusters
- BUT: How do we derive these components?

General Approach

- The PCA performs a basis transformation, in which the first basis vector is the vector accounting for most of the variance in the dataset, the second for the most of the remaining variance and so on ...
- These basis vectors can be found by the eigenvalue decomposition of the covariance matrix Q or the sample correlation matrix R.
- The eigenvalues $\lambda_1, \ldots, \lambda_d$ indicate the variance of the eigenvectors $y_1, \ldots, y_d$

How it works?

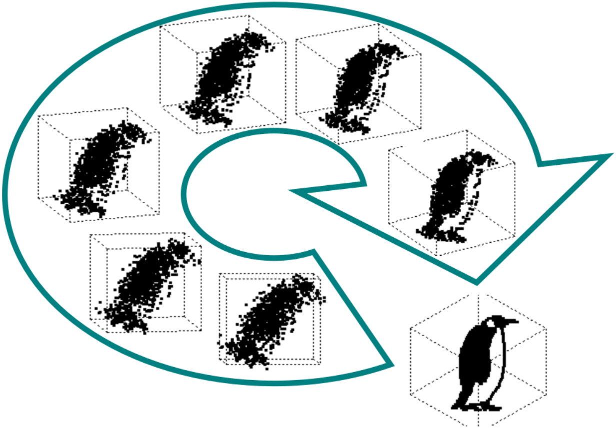
Observation

- In the example some structure became suddenly visible
- Now the data suggest to contain 3 clusters
- BUT: How do we derive these components?

General Approach

- The PCA performs a basis transformation, in which the first basis vector is the vector accounting for most of the variance in the dataset, the second for the most of the remaining variance and so on ...
- These basis vectors can be found by the eigenvalue decomposition of the covariance matrix Q or the sample correlation matrix R.
- The eigenvalues $\lambda_1, \ldots, \lambda_d$ indicate the variance of the eigenvectors $y_1, \ldots, y_d$

Example for a PCA



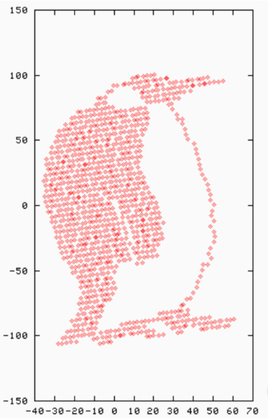


Image taken from Ricardo Gutierrez-Osuna's class on Pattern Analysis

Let's look into details

- Don't panic, we won't go too much into details
- We will perform a small PCA step by step
- Many of the following slides are taken and adapted from: `http://www.cse.psu.edu/~rcollins/CSE586Spring2010/`
    - this is a computer vision class ... just so you can see how often you will be faced with a PCA

The Covariance

- Variance and Covariance are a measure of the “spread“ of a set of points around their center of mass (mean)
- Variance – measure of the deviation from the mean for points in one dimension e.g. heights
- Covariance as a measure of how much each of the dimensions vary from the mean with respect to each other
- 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

- The covariance is defined as

$$\mathrm{Cov}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})$$

- This is the observed covariance for n observations (x_i, y_i)
- $\overline{X}$ and $\overline{Y}$ are the observed mean of the two given dimensions, i.e.,

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Covariance

- The covariance is defined as

$$\mathrm{Cov}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})$$

- This is the <u>observed</u> covariance for n observations (x_i, y_i)
- $\overline{X}$ and $\overline{Y}$ are the <u>observed</u> mean of the two given dimensions, i.e.,

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The Covariance Matrix

Definition

- The covariance can be calculated between each pair of dimensions
- We can put all of the in a Matrix, e.g., for three dimensions X, Y, Z:

$$C = \begin{pmatrix} \mathrm{Cov}(X,X) & \mathrm{Cov}(X,Y) & \mathrm{Cov}(X,Z) \\ \mathrm{Cov}(Y,X) & \mathrm{Cov}(Y,Y) & \mathrm{Cov}(Y,Z) \\ \mathrm{Cov}(Z,X) & \mathrm{Cov}(Z,Y) & \mathrm{Cov}(Z,Z) \end{pmatrix}$$

Properties

- Diagonal is the variances of X, Y and Z
- $\mathrm{Cov}(X,Y) = \mathrm{Cov}(Y,X)$ hence matrix is symmetrical about the diagonal
- d-dimensional data will result in $d \times d$ covariance matrix

The Covariance Matrix

Definition

- The covariance can be calculated between each pair of dimensions
- We can put all of the in a Matrix, e.g., for three dimensions X, Y, Z:

$$C = \begin{pmatrix} \mathrm{Cov}(X,X) & \mathrm{Cov}(X,Y) & \mathrm{Cov}(X,Z) \\ \mathrm{Cov}(Y,X) & \mathrm{Cov}(Y,Y) & \mathrm{Cov}(Y,Z) \\ \mathrm{Cov}(Z,X) & \mathrm{Cov}(Z,Y) & \mathrm{Cov}(Z,Z) \end{pmatrix}$$

Properties

- Diagonal is the variances of X, Y and Z
- $\mathrm{Cov}(X,Y) = \mathrm{Cov}(Y,X)$ hence matrix is symmetrical about the diagonal
- d-dimensional data will result in $d \times d$ covariance matrix

What does a Covariance mean in the first place?

- The covariance is not bound to a certain range of values (e.g., between $[0, 1]$)
- So, what does a covariance of 37.6 mean?
  - In terms of covariance, the sign is more important than the value
  - Positive: Both dimensions increase/decrease together
  - Negative: While one dimension increases the other decreases
  - Zero: The two dimensions are independent of each other

Remember:

- 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

What does a Covariance mean in the first place?

- The covariance is not bound to a certain range of values (e.g., between $[0, 1]$)
- So, what does a covariance of 37.6 mean?
  - In terms of covariance, the sign is more important than the value
  - Positive: Both dimensions increase/decrease together
  - Negative: While one dimension increases the other decreases
  - Zero: The two dimensions are independent of each other

Remember:

- 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

What does a Covariance mean in the first place?

- The covariance is not bound to a certain range of values (e.g., between $[0, 1]$)
- So, what does a covariance of 37.6 mean?
  - In terms of covariance, the sign is more important than the value
  - Positive: Both dimensions increase/decrease together
  - Negative: While one dimension increases the other decreases
  - Zero: The two dimensions are independent of each other

Remember:

- 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

Back to PCA

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be a set of d dimensional vectors. Further, let $\overline{\mathbf{x}}$ be the mean-vector:

$$\mathbf{x}_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{pmatrix} \qquad \overline{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{pmatrix}$$

Furthermore, let X be the $d \times n$-matrix of the form

$$X = \begin{pmatrix} \mathbf{x}_1 - \overline{\mathbf{x}} & \mathbf{x}_2 - \overline{\mathbf{x}} & \ldots & \mathbf{x}_n - \overline{\mathbf{x}} \end{pmatrix}$$

Which is basically just the dataset centered around the mean

Back to PCA

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be a set of d dimensional vectors. Further, let $\overline{\mathbf{x}}$ be the mean-vector:

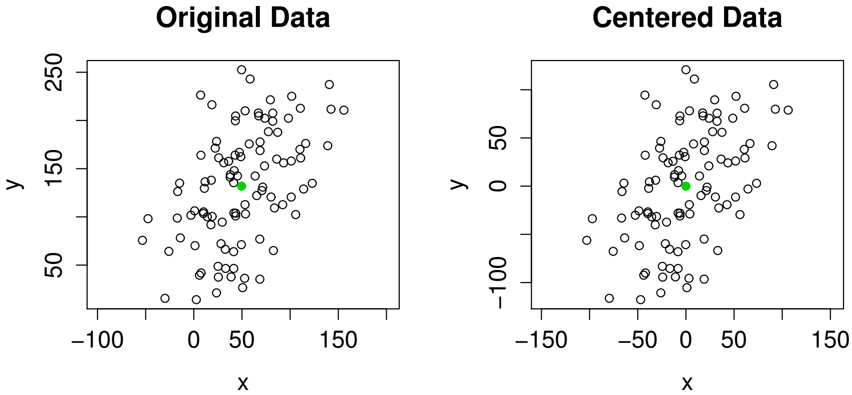
$$\mathbf{x}_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{pmatrix} \qquad \overline{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{pmatrix}$$

Furthermore, let X be the $d \times n$-matrix of the form

$$X = \begin{pmatrix} \mathbf{x}_1 - \overline{\mathbf{x}} & \mathbf{x}_2 - \overline{\mathbf{x}} & \ldots & \mathbf{x}_n - \overline{\mathbf{x}} \end{pmatrix}$$

Which is basically just the dataset centered around the mean

What we have so far:



The Covariance Matrix

Now, the Covariance matrix Q can easily be calculated as $Q = XX^\top$ which is

$$Q = \frac{1}{n-1} XX^\top = \frac{1}{n-1} \begin{pmatrix} \mathbf{x}_1 - \overline{\mathbf{x}} & \mathbf{x}_2 - \overline{\mathbf{x}} & \dots & \mathbf{x}_n - \overline{\mathbf{x}} \end{pmatrix} \begin{pmatrix} (\mathbf{x}_1 - \overline{\mathbf{x}})^\top \\ (\mathbf{x}_2 - \overline{\mathbf{x}})^\top \\ \vdots \\ (\mathbf{x}_n - \overline{\mathbf{x}})^\top \end{pmatrix}$$

Note:

- Q is square
- Q is symmetric
- Q is a $d \times d$ matrix

Perform the PCA

- Now, the Eigenvectors $e_1, \ldots, e_d$ of Q give us the principal components
- How does us help that?

- Each $\mathbf{x}_j$ can now be written as

$$\mathbf{x}_j = \overline{\mathbf{x}} + \sum_{i=1}^{d} g_{ij} e_i$$

where e_i are the eigenvectors of Q with non-zero eigenvalues

- The eigenvectors span the eigenspace
- The scalars g_{ij} are the coordinates of x_i in the space

Perform the PCA

- Now, the Eigenvectors $e_1, \ldots, e_d$ of Q give us the principal components
- How does us help that?

- Each $\mathbf{x}_j$ can now be written as

$$\mathbf{x}_j = \overline{\mathbf{x}} + \sum_{i=1}^{d} g_{ij} e_i$$

where e_i are the eigenvectors of Q with non-zero eigenvalues

- The eigenvectors span the eigenspace
- The scalars g_{ij} are the coordinates of x_i in the space

Perform the PCA

- Now, the Eigenvectors $e_1, \ldots, e_d$ of Q give us the principal components
- How does us help that?

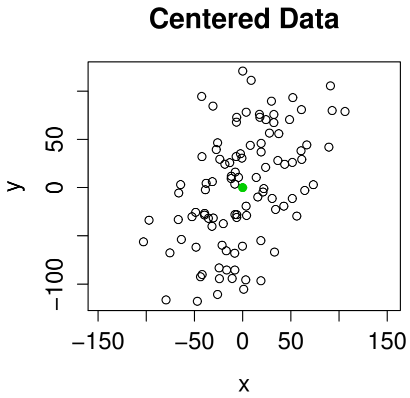
- Each $\mathbf{x}_j$ can now be written as

$$\mathbf{x}_j = \overline{\mathbf{x}} + \sum_{i=1}^{d} g_{ij} e_i$$

where e_i are the eigenvectors of Q with non-zero eigenvalues

- The eigenvectors span the eigenspace
- The scalars g_{ij} are the coordinates of x_i in the space

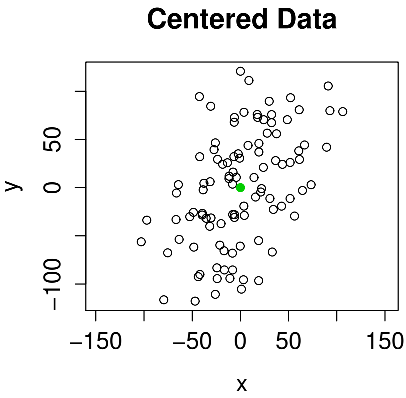
Our Small Example

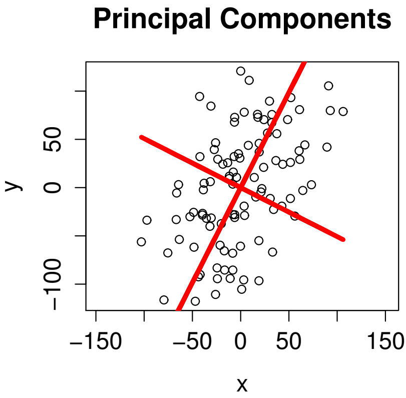


- The basis vectors are

$$e_1 = \begin{pmatrix} 0.45 \\ 0.89 \end{pmatrix} \qquad e_2 = \begin{pmatrix} -0.89 \\ 0.454 \end{pmatrix}$$

Our Small Example

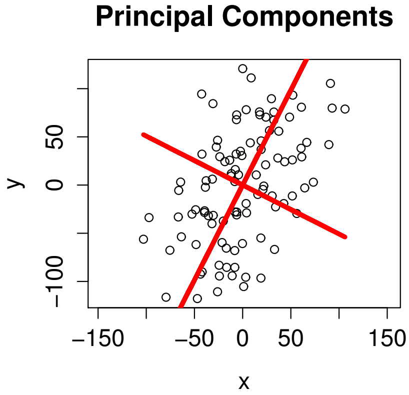




- The basis vectors are

$$e_1 = \begin{pmatrix} 0.45 \\ 0.89 \end{pmatrix} \qquad e_2 = \begin{pmatrix} -0.89 \\ 0.454 \end{pmatrix}$$

We also can rotate the data

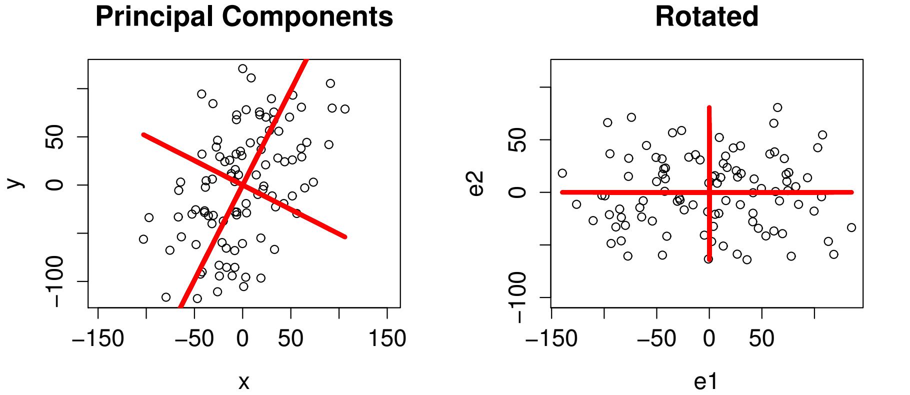


- The rotation points can be calculated by

$$R = E^\top \cdot X^\top$$

with E being the matrix of the eigenvectors $E = (\mathbf{e_1} \quad \mathbf{e_2} \quad \dots \quad \mathbf{e_d})$ and X the mean-corrected dataset

We also can rotate the data



- The rotation points can be calculated by

$$R = E^{\top} \cdot X^{\top}$$

with E being the matrix of the eigenvectors $E = (\mathbf{e_1} \quad \mathbf{e_2} \quad \ldots \quad \mathbf{e_d})$ and X the mean-corrected dataset

What now? - Using the PCA for Dimensionality Reduction

- Fantastic, all that stuff in order to rotate some data?
- Expressing X in terms of $\mathbf{e_1}, \ldots, \mathbf{e_1}$ has not changed the size of the data at all, we just performed a basis transformation
- **BUT:** Hopefully, most of the new coordinates have values close to zero (as there is almost no variance "left" when calculating the PCAs)
- That means in turn, the data lie in a lower-dimensional linear subspace ...
- Thus, we don't use all of the eigenvectors to transform the data
- Which ones should we take?

What now? - Using the PCA for Dimensionality Reduction

- Fantastic, all that stuff in order to rotate some data?
- Expressing X in terms of $\mathbf{e_1}, \ldots, \mathbf{e_1}$ has not changed the size of the data at all, we just performed a basis transformation

- **BUT:** Hopefully, most of the new coordinates have values close to zero (as there is almost no variance "left" when calculating the PCAs)
- That means in turn, the data lie in a lower-dimensional linear subspace ...
- Thus, we don't use all of the eigenvectors to transform the data
- Which ones should we take?

Dimensionality Reduction with PCA

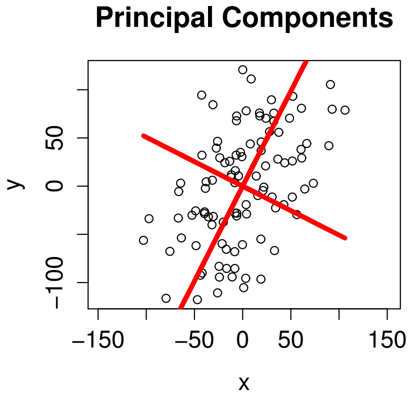
- Let λ_i be the eigenvalue belonging to the eigenvector $\mathbf{e_i}$
- Assume, the list of eigenvectors is sorted such that the according eigenvalues fulfill

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$$

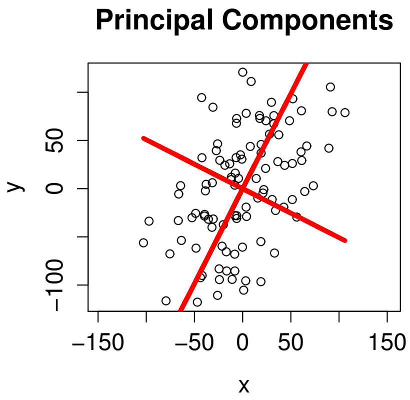
- Assume $\lambda_i \approx 0$ when $i > k$
- Then

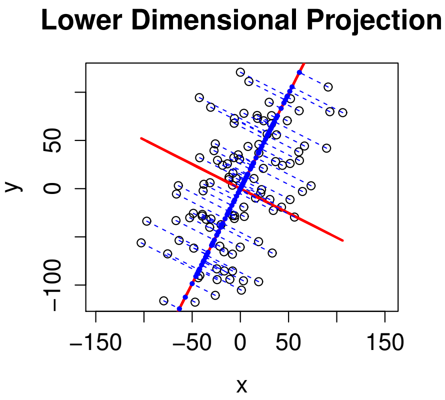
$$\mathbf{x_j} \approx \overline{\mathbf{x}} + \sum_{i=1}^{k} g_{ij} e_i$$

In Our Example

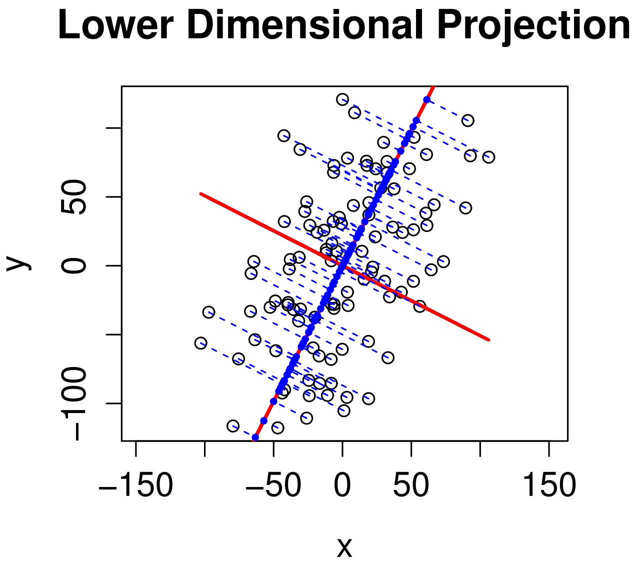


In Our Example





Zoom in



Some Final Remarks

- Obviously, a PCA makes more sense, when performed on higher dimensional data
- We have not looked at any proofs (i.e., do the eigenvectors actually give us the vectors with the most variance?)
- We haven't discussed how to find eigenvectors efficiently
  - Most programming languages have implementations available
  - For example `eigen(X)` in R
- The lost of variance can actually be calculated by

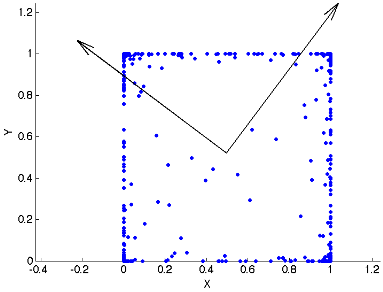
$$\frac{\sum_{i=0}^{k} \mathbf{e_i}}{\sum_{i=0}^{d} \mathbf{e_i}}$$

where k is the number of "used" coordinates

Problems with PCA

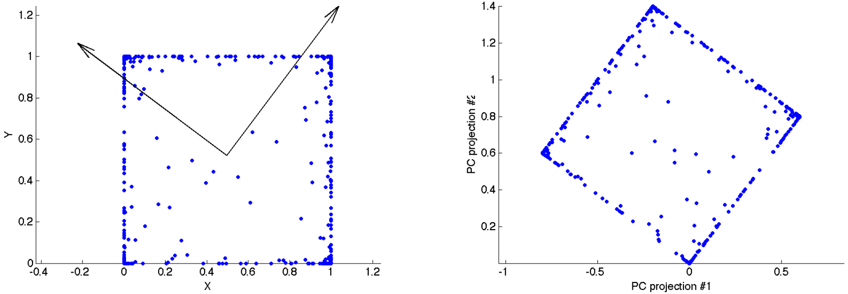
- PCA is not without its problems and limitations
- PCA assumes approximate normality of the input space distribution
- PCA may still be able to produce a "good" low dimensional projection of the data even if the data isn't normally distributed
- PCA may "fail" if the data lies on a "complicated" manifold
- PCA assumes that the input data is real and continuous

Problems with PCA



Images taken from Frank Wood's lecture slides of his "Linear Regression Models" course, `http://www.robots.ox.ac.uk/~fwood`

Problems with PCA



Images taken from Frank Wood's lecture slides of his "Linear Regression Models" course, `http://www.robots.ox.ac.uk/~fwood`

Lower-Dimensional Projections

Multidimensional Scaling

Goal of Multidimensional Scaling

- Detecting underlying structure
- Represent data in lower dimensional space so that distances are preserved
  - Distances between data points are mapped to a reduced space
  - In other words, we are looking for a projection of the data measured with some distance into an Euclidean space
- Typically displayed on a 2-d plot
- We will briefly discuss
  - Metric (or classic) multidimensional scaling
  - Non-metric multidimensional scaling

The following slides are based on `http://www.cs.toronto.edu/~bonner/courses/2007s/csc411/lectures/16mds.pdf` and `http://www.cedar.buffalo.edu/~srihari/CSE626/Lecture-Slides/Ch3-part2-PCA.pdf`

Metric (or classic) multidimensional scaling

- Recall: given a $n \times d$ two-mode dataset containing the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.
- Assume the data is already centered around 0, i.e., $\overline{\mathbf{x}} = \mathbf{0}$
  - Just makes the rest easier, performing a shift as seen before is no problem!
- Now consider again the matrix $X = \{\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_n\}$
- Recall: $XX^\top$ gave us the $d \times d$ Covariance matrix
- The $n \times n$ Matrix $B = X^\top X$ is also very useful
- In fact, we will see on the next slide, that the Euclidean distance is given by

$$d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij}$$

Metric (or classic) multidimensional scaling

- Recall: given a $n \times d$ two-mode dataset containing the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.
- Assume the data is already centered around 0, i.e., $\overline{\mathbf{x}} = \mathbf{0}$
  - Just makes the rest easier, performing a shift as seen before is no problem!
- Now consider again the matrix $X = \{\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_n\}$
- Recall: $XX^\top$ gave us the $d \times d$ Covariance matrix
- The $n \times n$ Matrix $B = X^\top X$ is also very useful
- In fact, we will see on the next slide, that the Euclidean distance is given by

$$d_{ij}^2 = b_{ii} + b_{jj} - 2b_{ij}$$

Closer look at $X^\top X$

Let $n = 4, d = 3$, then

$$B = X^\top X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \end{pmatrix} \times \begin{pmatrix} x_{11} & x_{21} & x_{31} & x_{41} \\ x_{12} & x_{22} & x_{32} & x_{42} \\ x_{13} & x_{23} & x_{33} & x_{43} \end{pmatrix} =$$

$$= \begin{pmatrix} b_{11} = \sum_{d=1}^{3} x_{1d}^2 & b_{12} = \sum_{d=1}^{3} x_{1d} x_{2d} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix}$$

Remember the definition of the Euclidean distance, we get:

$$d_{ij}^2 = \sum_{d=1}^{3} (x_{id} - x_{jd})^2 = b_{ii} + b_{jj} - 2b_{ij}$$

* note the rather unusual indices in the matrix, because we wrote each vector $\mathbf{x}_i$ in a column. Sometimes, you find X being defined as having each vector in a row which only changes the position of the transposed sign.

Algorithm

1. Calculate the eigenvalues λ_i and eigenvectors γ_i of the matrix B
2. Scale the eigenvectors γ_i such that $\sum_{j=1}^{n} \gamma_{ij} = \lambda_i$
3. The d eigenvectors of the d largest eigenvalues give the d-dimensional coordinates

- Very often, we have only given a distance matrix D, thus a one-mode matrix with metric distances which can be converted into B by
  1. Define $A = (a_{ij})$ with $a_{ij} = -\frac{1}{2} d_{ij}^2$
  2. Now, you can define $B = (b_{ij})$ as

$$b_{ij} = a_{ij} - a_{i\bullet} - a_{\bullet j} + a_{\bullet\bullet}$$

  with $a_{i\bullet}$ being the average of row i, $a_{\bullet j}$ the average of column j and $a_{\bullet\bullet}$ the average of the matrix A
- Proceed as above

Algorithm

1. Calculate the eigenvalues λ_i and eigenvectors γ_i of the matrix B
2. Scale the eigenvectors γ_i such that $\sum_{j=1}^{n} \gamma_{ij} = \lambda_i$
3. The d eigenvectors of the d largest eigenvalues give the d-dimensional coordinates

- Very often, we have only given a distance matrix D, thus a one-mode matrix with metric distances which can be converted into B by
  1. Define $A = (a_{ij})$ with $a_{ij} = -\frac{1}{2}d_{ij}^2$
  2. Now, you can define $B = (b_{ij})$ as

  $$b_{ij} = a_{ij} - a_{i\bullet} - a_{\bullet j} + a_{\bullet\bullet}$$

  with $a_{i\bullet}$ being the average of row i, $a_{\bullet j}$ the average of column j and $a_{\bullet\bullet}$ the average of the matrix A
- Proceed as above

Non-Metric Multidimensional Scaling

- We will just have a brief overview
- Sometimes, distances are not given with metric distance
    - We will learn more about distances later on
- Given: A dataset X and a distance matrix D containing pair-wise distances obtained in <u>any</u> fashion appropriate for the data
- Goal: A representation Y in 2-D or 3-D space preserving (as good as possible) the pair-wise distances given in D

Multidimensional Scaling

To write the last slide a little bit more formally

- Let δ_{ij} denote the distance between object x_i and x_j given in the distance matrix D
- Let $y_i = (y_{i1}, \ldots, y_{id})$ the representation of x_i in a d-dimensional euclidean space
- Let d_{ij} denote the Euclidean distance between y_i and y_j
- Goal: $\forall i, j : \quad d_{ij} \approx \delta_{ij}$
- Exact equality generally not possible, so minimize an error function J

Multidimensional Scaling

To write the last slide a little bit more formally

- Let δ_{ij} denote the distance between object x_i and x_j given in the distance matrix D
- Let $y_i = (y_{i1}, \ldots, y_{id})$ the representation of x_i in a d-dimensional euclidean space
- Let d_{ij} denote the Euclidean distance between y_i and y_j
- Goal: $\forall i, j : \quad d_{ij} \approx \delta_{ij}$
- Exact equality generally not possible, so minimize an error function J

Multidimensional Scaling

To write the last slide a little bit more formally

- Let δ_{ij} denote the distance between object x_i and x_j given in the distance matrix D
- Let $y_i = (y_{i1}, \ldots, y_{id})$ the representation of x_i in a d-dimensional euclidean space
- Let d_{ij} denote the Euclidean distance between y_i and y_j
- Goal: $\forall i, j : \quad d_{ij} \approx \delta_{ij}$
- Exact equality generally not possible, so minimize an error function J

Multidimensional Scaling

To write the last slide a little bit more formally

- Let δ_{ij} denote the distance between object x_i and x_j given in the distance matrix D
- Let $y_i = (y_{i1}, \ldots, y_{id})$ the representation of x_i in a d-dimensional euclidean space
- Let d_{ij} denote the Euclidean distance between y_i and y_j
- Goal: $\forall i, j : \quad d_{ij} \approx \delta_{ij}$
- Exact equality generally not possible, so minimize an error function J

Error Functions

$$J_1 = \frac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$$

(penalizes large absolute errors)

$$J_2 = \sum_{i<j}\left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2$$

(penalizes large relative errors)

$$J_3 = \frac{1}{\sum_{i<j}\delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}$$

(a compromise between the two)

Error Functions

$$J_1 = \frac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$$

(penalizes large absolute errors)

$$J_2 = \sum_{i<j}\left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2$$

(penalizes large relative errors)

$$J_3 = \frac{1}{\sum_{i<j}\delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}$$

(a compromise between the two)

Error Functions

$$J_1 = \frac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j}\delta_{ij}^2}$$

(penalizes large absolute errors)

$$J_2 = \sum_{i<j}\left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}}\right)^2$$

(penalizes large relative errors)

$$J_3 = \frac{1}{\sum_{i<j}\delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}$$

(a compromise between the two)

Algorithms

- Normally, a iterative algorithm using some version of steepest descent is employed

- Genreal Procedure:
  1. Obtain the distances δ_{ij}
  2. Initialize Y (e.g., randomly)
  3. Calculate gradient updates and update point positions accordingly
  4. Repeat

- Grandient formula, e.g.:

$$\bigtriangledown J_1 = \frac{2}{\sum_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} - \delta_{kj}) \frac{y_k - y_j}{d_{kj}}$$

Algorithms

- Normally, a iterative algorithm using some version of steepest descent is employed

- Genreal Procedure:
  1. Obtain the distances δ_{ij}
  2. Initialize Y (e.g., randomly)
  3. Calculate gradient updates and update point positions accordingly
  4. Repeat

- Grandient formula, e.g.:

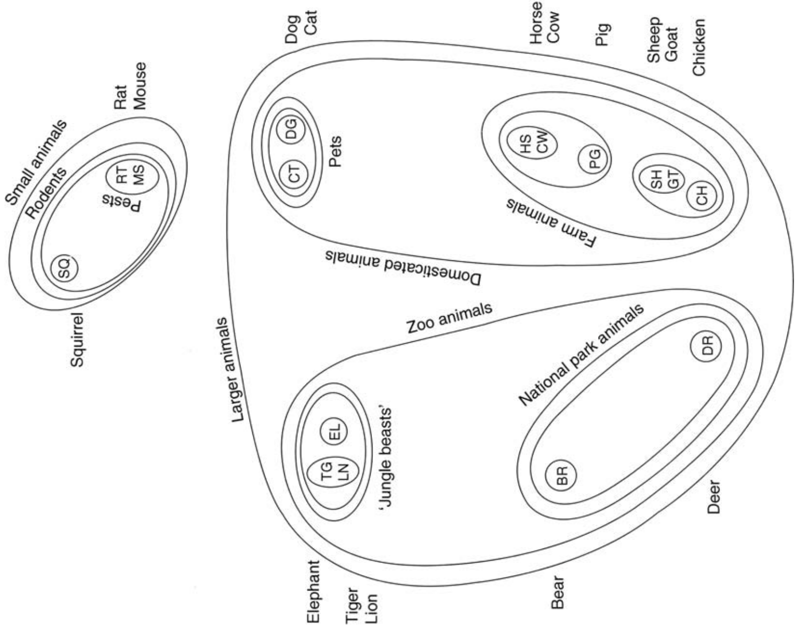
$$\nabla J_1 = \frac{2}{\sum_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} - \delta_{kj}) \frac{y_k - y_j}{d_{kj}}$$

Algorithms

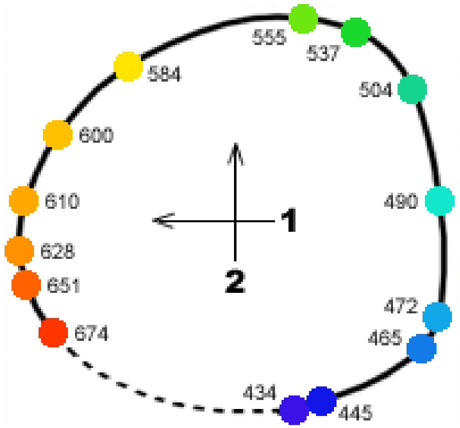
- Normally, a iterative algorithm using some version of steepest descent is employed
- Genreal Procedure:
  1. Obtain the distances δ_{ij}
  2. Initialize Y (e.g., randomly)
  3. Calculate gradient updates and update point positions accordingly
  4. Repeat
- Grandient formula, e.g.:

$$\bigtriangledown J_1 = \frac{2}{\sum_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} - \delta_{kj}) \frac{y_k - y_j}{d_{kj}}$$

Example


Small animals
Rodents
Pests
Rat
Mouse
Squirrel
RT
MS
SQ
Dog
Cat
Horse
Cow
Pig
Sheep
Goat
Chicken
DG
CT
Pets
HS
CW
PG
SH
GT
CH
Farm animals
Domesticated animals
Larger animals
Zoo animals
National park animals
DR
BR
EL
TG
LN
'Jungle beasts'
Elephant
Tiger
Lion
Bear
Deer

Example



- People were ask to judge the similarity between 14 colors
- Circle resembles the well-known color circle

Original Study: MLA Shepard, Roger N. ”The analysis of proximities: Multidimensional scaling with an unknown distance function. II.” Psychometrika 27.3 (1962): 219-246. Colorized picture taken from: `https://www.handprint.com/HP/WCL/color2.html`

Remarks & Conclusion

Remarks

- What we have seen here, is the so-called
  - Principal Coordinate Analysis (PCO) or metric multidimensional scaling (MDS) or classical scaling
  - Non-metric multidimensional scaling
- There are even more methods ...

Limitations

- When there are too many data points structure becomes obscured
- Highly sophisticated transformations of the data (compared to scatter lots and PCA)
  - E.g., introduction of artifacts