Felix Reinbott

Dimension reduction, clustering and more

An overview of some unsupervised learning techniques

Warnemünde, 26.09.2023

Institute for Mathematical Stochastics
Otto-von-Guericke-University Magdeburg

A few examples of typical problems from applications

Detecting insurance fraud

Assume we have requests for insurance offers based on user criteria like age, height, etc.
Some people do multiple requests by changing parameters slightly to obtain better offers.
Can we find the requests that belong to one person?

A few examples of typical problems from applications

Detecting insurance fraud

Assume we have requests for insurance offers based on user criteria like age, height, etc. Some people do multiple requests by changing parameters slightly to obtain better offers. Can we find the requests that belong to one person?

Image compression

Assume we have 256×256 pixel images of a certain handwritten number. Can we compress these images in a memory-efficient way? It might be plausible that these images lie in some low dimensional subspace of $\mathbb{R}^{256 \cdot 256}$.

A few examples of typical problems from applications

Detecting insurance fraud

Assume we have requests for insurance offers based on user criteria like age, height, etc. Some people do multiple requests by changing parameters slightly to obtain better offers. Can we find the requests that belong to one person?

Image compression

Assume we have 256×256 pixel images of a certain handwritten number. Can we compress these images in a memory-efficient way? It might be plausible that these images lie in some low dimensional subspace of $\mathbb{R}^{256 \cdot 256}$.

Detecting prototypical customers

A supermarket chain wants to identify $k \in \mathbb{N}$ archetypes of customers to optimize their market layout. Is it possible to find k archetypes of customers in a reasonable manner?

A general outline of unsupervised learning techniques

Assume that we have data $x_1, \dots, x_n \in \mathbb{R}^d$, d "large", that are realizations of an i.i.d. sample from some probability measure $\mathbb{P}^X$ that is unknown.

A general outline of unsupervised learning techniques

Assume that we have data $x_1, \dots, x_n \in \mathbb{R}^d$, d "large", that are realizations of an i.i.d. sample from some probability measure $\mathbb{P}^X$ that is unknown.

Usual task: Infer properties of interest about $\mathbb{P}^X$ from data

Usually we care about things like

- Does the data concentrate around some lower dimensional space or manifold?
- If $\mathbb{P}^X$ has a density f, does f have multiple local maxima?

A general outline of unsupervised learning techniques

Assume that we have data $x_1, \dots, x_n \in \mathbb{R}^d$, d "large", that are realizations of an i.i.d. sample from some probability measure $\mathbb{P}^X$ that is unknown.

Usual task: Infer properties of interest about $\mathbb{P}^X$ from data

Usually we care about things like

- Does the data concentrate around some lower dimensional space or manifold?
- If $\mathbb{P}^X$ has a density f, does f have multiple local maxima?

Immediate problem compared to linear regression

We do not have a reference variable to check how good our model is.

Principal Component Analysis (PCA)

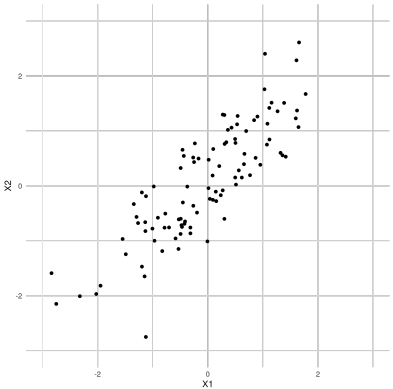


Figure: An i.i.d. sample of 100 bivariate gaussians.

Sample shows most variance along line with positive slope.

Principal Component Analysis (PCA)

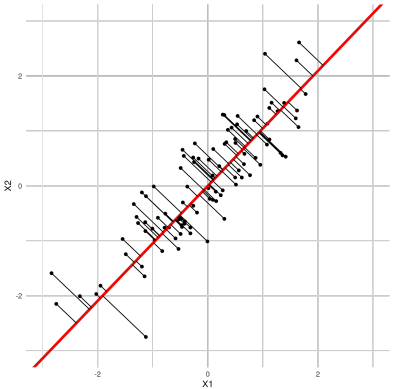


Figure: An i.i.d. sample of 100 bivariate gaussians.

Sample shows most variance along line with positive slope.

If we project on optimal line, we obtain new points which are one-dimensional and lie "close" to the original data.

Principal Component Analysis (PCA)

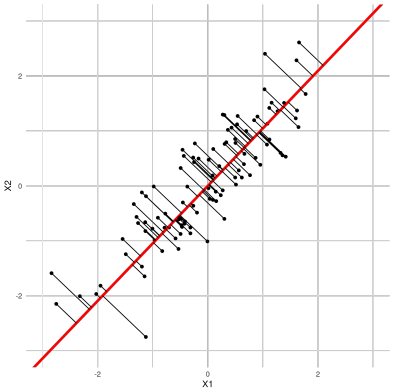


Figure: An i.i.d. sample of 100 bivariate gaussians.

Sample shows most variance along line with positive slope.

If we project on optimal line, we obtain new points which are one-dimensional and lie "close" to the original data.

→ Let's make that precise.

How to compute PCA

Goal: Find linear subspace that lies close to data

Assume that $X \in L^2$ is a d-dimensional random vector and we want to map X to some $p << d$ dimensional subspace that is best in the sense that

$$\mathbb{E}\left[\|X - PX\|_2^2\right]$$

is minimal and the orthogonal projection matrix P is a projection to the subspace.

How to compute PCA

Goal: Find linear subspace that lies close to data

Assume that $X \in L^2$ is a d-dimensional random vector and we want to map X to some $p << d$ dimensional subspace that is best in the sense that

$$\mathbb{E}\left[\|X - PX\|_2^2\right]$$

is minimal and the orthogonal projection matrix P is a projection to the subspace.

→ We can simplify that expression drastically!

A more convenient reformulation

Using standard linear algebra, it can be shown that minimizing the distance of X to some p dimensional subspace is the same as teratively solving

$$\max_{v_i \in \mathbb{R}^d : \|v_i\|_2 = 1} v_i^T \Sigma v_i, \quad \langle v_i, v_j \rangle = 0, i \neq j, i = 1, \ldots, p.$$

Solving this gives us that we choose v_i to be an eigenvector belonging to the i-th largest eigenvalue of Σ.

A more convenient reformulation

Using standard linear algebra, it can be shown that minimizing the distance of X to some p dimensional subspace is the same as teratively solving

$$\max_{v_i \in \mathbb{R}^d : \|v_i\|_2 = 1} v_i^T \Sigma v_i, \quad \langle v_i, v_j \rangle = 0, i \neq j, i = 1, \ldots, p.$$

Solving this gives us that we choose v_i to be an eigenvector belonging to the i-th largest eigenvalue of Σ.

Main takeaway: Projecting data on linear subspace is solving an eigenvalue problem!

→ We can easily apply this to data by estimating the covariance of the data!

An application: Hand written digits

We will work through a coded example using *R*.

A model based approach to dimension reduction: Factor analysis

Assume that we have centered data d-variate data X_i, $i = 1, \ldots, n$ generated from

$$X_i = AZ_i + \epsilon_i,$$

Where

- $A \in \mathbb{R}^{d \times p}$ is called the factor loading matrix and $p < d$,
- Z_i independent p-dimensional RV's standardized with uncorrelated components,
- ϵ_i are independent, centered d-dimensional RV's,
- each Z_i is independent of every ϵ_j, $j = 1, \ldots, n$.

A model based approach to dimension reduction: Factor analysis

Assume that we have centered data d-variate data X_i, $i = 1, \ldots, n$ generated from

$$X_i = AZ_i + \epsilon_i,$$

Where

- $A \in \mathbb{R}^{d \times p}$ is called the factor loading matrix and $p < d$,
- Z_i independent p-dimensional RV's standardized with uncorrelated components,
- ϵ_i are independent, centered d-dimensional RV's,
- each Z_i is independent of every ϵ_j, $j = 1, \ldots, n$.
- USUALLY assumed that Z_i and ϵ_i are gaussian with diagonal covariance matrix.

A model based approach to dimension reduction: Factor analysis

From these assumptions we can calculate

$$\mathbb{E}X_i = \mathbb{E}\left[AZ_i + \epsilon_i\right] = A\mathbb{E}Z_i + \mathbb{E}\epsilon_i = 0.$$

A model based approach to dimension reduction: Factor analysis

From these assumptions we can calculate

$$\mathbb{E}X_i = \mathbb{E}[AZ_i + \epsilon_i] = A\mathbb{E}Z_i + \mathbb{E}\epsilon_i = 0.$$

This allows us to calculate the covariance easily by

$$Cov(X_i, X_i) = \mathbb{E}[(AZ_i + \epsilon_i)(Z_i^T A^T + \epsilon_i^T)] = AA^T + D_\epsilon$$

A model based approach to dimension reduction: Factor analysis

From these assumptions we can calculate

$$\mathbb{E}X_i = \mathbb{E}[AZ_i + \epsilon_i] = A\mathbb{E}Z_i + \mathbb{E}\epsilon_i = 0.$$

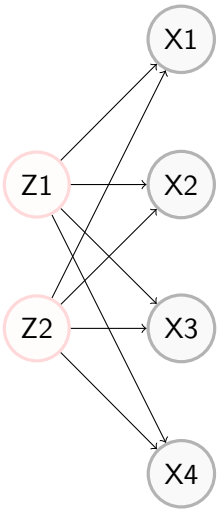
This allows us to calculate the covariance easily by

$$Cov(X_i, X_i) = \mathbb{E}[(AZ_i + \epsilon_i)(Z_i^T A^T + \epsilon_i^T)] = AA^T + D_\epsilon$$

→ Many possiblites to fit model, assume normality and fit with maximum-likelihood or use distance minimization approach from theoretical covariance to empirical covariance.

Take care: $AA^T = ARR^TA^T$ for any orthonormal $R \in \mathbb{R}^{p\times p}$, hence, A is not unique.

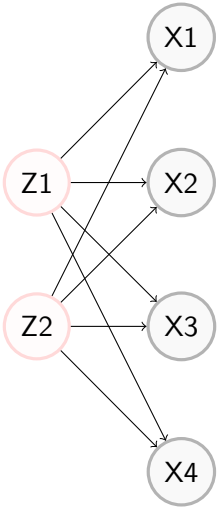
A nice visualization of factor models



We can interpret the factor model as a certain type of directed graph, where the observed data is obtained by adding the latent factors scaled with edge weights plus a random perturbation.

Why is this useful?

A nice visualization of factor models



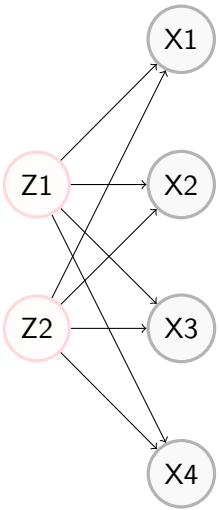
We can interpret the factor model as a certain type of directed graph, where the observed data is obtained by adding the latent factors scaled with edge weights plus a random perturbation.

Why is this useful?

We can validate hypothesised causal structures

For example in survey data , it is assumed that certain traits affect a particular set of questions only.

A nice visualization of factor models



We can interpret the factor model as a certain type of directed graph, where the observed data is obtained by adding the latent factors scaled with edge weights plus a random perturbation.

Why is this useful?

We can validate hypothesised causal structures

For example in survey data , it is assumed that certain traits affect a particular set of questions only.

In high dimensional settings, sparsity helps

Sparsity greatly reduces the number of parameters to estimate.

→ Frank will tell you more about graphs!

An application: Again hand written digits

We will work through a coded example using *R* to see how FA performs.

A general overview of clustering

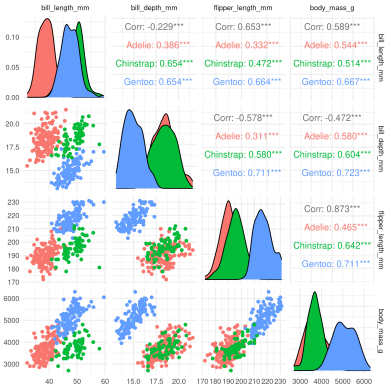


Figure: A dataset of different mesurements of penguins for three species.

Often multivariate data appears to have multiple "point clouds", for example induced by the species of the penguins in the plot.

A general overview of clustering

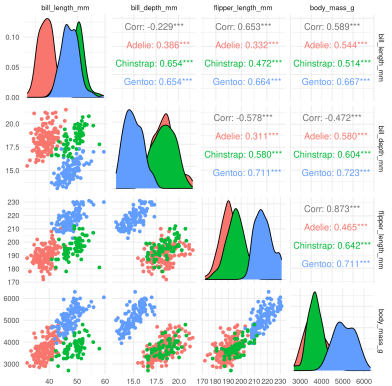


Figure: A dataset of different mesurements of penguins for three species.

Often multivariate data appears to have multiple "point clouds", for example induced by the species of the penguins in the plot.

Sometimes the driving factor for these "point clouds" is not known, but we still want to find groups in the data and obtain basic statistics about these potential groups.

A general overview of clustering

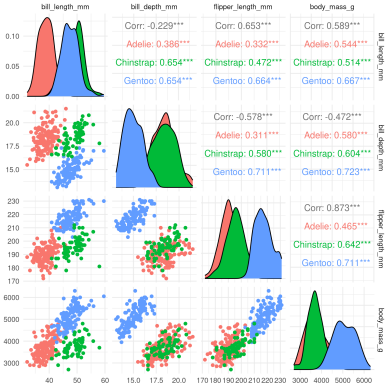


Figure: A dataset of different mesurements of penguins for three species.

Often multivariate data appears to have multiple "point clouds", for example induced by the species of the penguins in the plot.

Sometimes the driving factor for these "point clouds" is not known, but we still want to find groups in the data and obtain basic statistics about these potential groups.

Clustering methods help to segment data

We will present some of the methods throughout the next few slides.

A simple clustering algorithm: Agglomerative clustering

We start off with a simple but effective procedure. Let $X \in \mathbb{R}n \times p$ be a data matrix. Consider the data as n initial clusters. Then for $i = 1, \ldots, n$ do

- Find clusters which are closest by some closeness measure c,
- join the two closest clusters to a new cluster.

A simple clustering algorithm: Agglomerative clustering

We start off with a simple but effective procedure. Let $X \in \mathbb{R}n \times p$ be a data matrix. Consider the data as n initial clusters. Then for $i = 1, \ldots, n$ do

- Find clusters which are closest by some closeness measure c,
- join the two closest clusters to a new cluster.

Dendrograms: A method to visualize clusters

- The leafs are the individual data points.
- The root is the final cluster of all data points merged.
- Joining branches mean that clusters merged.

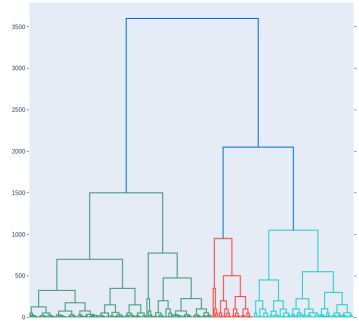


Figure: A dendrogram created from the distance matrix of the penguins dataset.

A note on cost functions to merge clusters

Suppose we are in step $i < n$ of the agglomerative clustering and currently have p clusters denoted by C_j, $j = 1, \ldots, p$. Then we merge two or possibly more clusters by

$$\underset{j,l=1,\ldots p, j \neq l}{\arg\min} \; c(C_j, C_l)$$

A note on cost functions to merge clusters

Suppose we are in step $i < n$ of the agglomerative clustering and currently have p clusters denoted by C_j, $j = 1, \dots, p$. Then we merge two or possibly more clusters by

$$\underset{j,l=1,\dots p, j \neq l}{\arg\min} \; c(C_j, C_l)$$

Where c is usually constructed using a p-norm, eventually to the p-th power and often the euclidean norm. A popular choice for c w.r.t. a p-norm is

$$c(C_j, C_l) := \min_{a \in C_j, b \in C_l} \|a - b\|_p$$

Many more choices are possible, for example merging by maximum distance, centroid distance or Haussdorff distance.

Finding k prototypes with k-means clustering

Sometimes we are interested to find a fixed number of clusters in our data x_i, $i = 1, \dots, n$ and want to extract extra information. For this, we solve

$$\min_{\substack{S_1,\dots,S_k \subseteq \{1,\dots,n\}, \\ \text{disjoint partition}}} \sum_{j=1}^{k} \sum_{i \in S_j} \|x_i - \mu_j\|_2^2, \quad \mu_j = \frac{1}{|S_j|} \sum_{i \in S_j} x_i.$$

From this we obtain group labels for our data x_i given by the set S_j, $j = 1, \dots, k$ the datapoint is assigned to and a group mean μ_j.

Mixture models for soft-margin clustering

A density of the form

$$f(x, \theta_1, \ldots, \theta_l, p_1, \ldots, p_l) := \sum_{j=1}^{l} p_j g_j(x, \theta_j), \qquad \sum_{j=1}^{l} p_j = 1,$$

is called a mixture model.

Mixture models for soft-margin clustering

A density of the form

$$f(x, \theta_1, \ldots, \theta_l, p_1, \ldots, p_l) := \sum_{j=1}^{l} p_j g_j(x, \theta_j), \quad \sum_{j=1}^{l} p_j = 1,$$

is called a mixture model.

For better understanding, if we want to sample a point from that model, we draw a point according to distribution g_j with probability p_j.
$\rightarrow$ l groups in this model, points in group follow same distribution.

Mixture models for soft-margin clustering

A density of the form

$$f(x, \theta_1, \ldots, \theta_l, p_1, \ldots, p_l) := \sum_{j=1}^{l} p_j g_j(x, \theta_j), \quad \sum_{j=1}^{l} p_j = 1,$$

is called a mixture model.

For better understanding, if we want to sample a point from that model, we draw a point according to distribution g_j with probability p_j.
$\rightarrow$ l groups in this model, points in group follow same distribution.

Gaussian assuptions make these models easy to compute

Again, if we add gaussian hot sauce, we get a good recipe to cook models with MLE!

Some notes on mixture modelling and k-means

Differences

- k-means is essentially model free, mixture models make strong parametric assumptions.
- (Gaussian) mixture models (GMM's) have "soft margins", since we also obtain the probability of a point belonging to each cluster, k-means assigns a fixed group label and might provide less information than GMM's.
- GMM's with full covariance models have fast growing number of parameters, hence, additional simplifications are needed in high dimensions.

Some notes on mixture modelling and k-means

Differences

- k-means is essentially model free, mixture models make strong parametric assumptions.
- (Gaussian) mixture models (GMM's) have "soft margins", since we also obtain the probability of a point belonging to each cluster, k-means assigns a fixed group label and might provide less information than GMM's.
- GMM's with full covariance models have fast growing number of parameters, hence, additional simplifications are needed in high dimensions.

Similiarities

- k-means appears as the limit of some GMM, $\rightsquigarrow$ no further details here...
- No guarantee for global optimal values in both procedures.