Lecture Notes
Institute of Bioinformatics, Johannes Kepler University Linz

Machine Learning: Unsupervised Techniques

Summer Semester 2014

by Sepp Hochreiter

Institute of Bioinformatics
Johannes Kepler University Linz
A-4040 Linz, Austria

Tel. +43 732 2468 8880
Fax +43 732 2468 9308
http://www.bioinf.jku.at

Literature

- Duda, Hart, Stork; Pattern Classification; Wiley & Sons, 2001
- C. M. Bishop; Neural Networks for Pattern Recognition, Oxford University Press, 1995
- C. M. Bishop; Pattern Recognition, Oxford University Press, 2008
- M. I. Jordan (ed.); Learning in Graphical Models, MIT Press, 1998 (Original by Kluwer Academic Pub.)
- T. M. Mitchell; Machine Learning, Mc Graw Hill, 1997
- R. Peck, C. Olsen and J. L. Devore; Introduction to Statistics and Data Analysis, 3rd edition, ISBN: 9780495118732, Brooks/Cole, Belmont, USA, 2009.
- B. Shahbaba; Biostatistics with R: An Introduction to Statistics Through Biological Data; Springer, series UseR!, ISBN 9781461413011, New York, 2012.
- C. T. Ekstrøm and H. Sørensen; Introduction to Statistical Data Analysis for the Life Sciences; CRC Press, Taylor & Francis Group, ISBN: 9781439825556, Boca Raton, USA, 2011.
- L. Kaufman and P. J. Rousseeuw; Finding Groups in Data. An Introduction to Cluster Analysis, Wiley, 1990.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Machine Learning Introduction

This course is part of the curriculum of the master in computer science (in particular the majors "Computational Engineering" and "Intelligent Information Systems") and part of the master in bioinformatics at the Johannes Kepler University Linz.

Machine learning is currently a major research topic at companies like Google, Microsoft, Amazon, Facebook, AltaVista, Zalando, and many more. Applications are found in computer vision (image recognition), speech recognition, recommender systems, analysis of Big Data, information retrieval. Companies that try to mine the world wide web are offering search engines, social networks, videos, music, information, or connecting people use machine learning techniques. Machine learning methods are used to classify and label web pages, images, videos, and sound recordings in web data. They can find specific objects in images and detect a particular music style if only given the raw data. Therefore Google, Microsoft, Facebook are highly interested in machine learning methods. Machine learning methods attracted the interest of companies offering products via the web. These methods are able to identify groups of similar users, to predict future behavior of customers, and can give recommendation of products in which customers will be interested based previous costumer data.

Machine learning has major applications in biology and medicine. Modern measurement techniques in both biology and medicine create a huge demand for new machine learning approaches. One such technique is the measurement of mRNA concentrations with microarrays and sequencing techniques. The measurement data are first preprocessed, then genes of interest are identified, and finally predictions made. Further machine learning methods are used to detect alternative splicing, nucleosome positions, gene regulation, etc. Alongside neural networks the most prominent machine learning techniques relate to support vector machines, kernel approaches, projection method and probabilistic models like latent variable models. These methods provide noise reduction, feature selection, structure extraction, classification / regression, and assist modeling. In the biomedical context, machine learning algorithms categorize the disease subtype or predict treatment outcomes based on DNA characteristics, gene expression profiles. Machine learning approaches classify novel protein sequences into structural or functional classes. For analyzing data of association studies, machine learning methods extract new dependencies between DNA markers (SNP - single nucleotide polymorphisms, SNV - single nucleotide variants, CNV - copy number variations) and diseases (Alzheimer, Parkinson, cancer, multiples sclerosis, schizophrenia or alcohol dependence).

The machine learning course series comprises:

- "Basic Methods of Data Analysis": this course gives a smooth introduction to machine learning with examples in R ; it covers summary statistics (mean, variance), data summary plots (boxplot, violin plot, scatter plot), principal component analysis, independent component analysis, multidimensional scaling (Kruskal's or Sammon's map), locally linear embedding, Isomap, hierarchical clustering, mixture models, k-means, similarity based clustering (affinity propagation), biclustering

- "Machine Learning: Supervised Methods": classification and regression techniques, time series prediction, kernel methods, support vector machines, neural networks, deep learning, deep neural and belief networks, ARMA and ARIMA models, recurrent neural networks, LSTM

- "Machine Learning: Unsupervised Methods": maximum likelihood estimation, maximum a posterior estimation, maximum entropy, expectation maximization, principal component analysis, statistical independence, independent component analysis, factor analysis, mixture models, sparse codes, population codes, kernel PCA, hidden Markov models (factorial HMMs and input-output HMMs), Markov networks and random fields, clustering, biclustering, restricted Boltzmann machines, auto-associators, unsupervised deep neural networks

- "Theoretical Concepts of Machine Learning": estimation theory (unbiased and efficient estimator, Cramer-Rao lower bound, Fisher information matrix), consistent estimator, complexity of model classes (VC-dimension, growth, annealed entropy), bounds on the generalization error, Vapnik and worst case bounds on the generalization error, optimization (gradient based methods and convex optimization), Bayes theory (posterior estimation, error bounds, hyperparameter optimization, evidence framework), theory on linear functions (statistical tests, intervals, ANOVA, generalized linear functions, mixed models)

In this course the most prominent machine learning techniques are introduced and their mathematical basis and derivatives are explained. If the student understands these techniques, then the student can select the methods which best fit to the problem at hand, the student is able to optimize the parameter settings for the methods, the student can adapt and improve the machine learning methods, and the student can develop new machine learning methods.

Most importantly, students should learn how to chose appropriate methods from a given pool of approaches for solving a specific problem. To this end, they must understand and evaluate the different approaches, know their advantages and disadvantages as well as where to obtain and how to use them. In a step further, the students should be able to adapt standard algorithms for their own purposes or to modify those algorithms for particular applications with certain prior knowledge or problem-specific constraints.

1.2 Course Specific Introduction

This course introduces unsupervised machine learning methods. If data has to be processed by machine learning methods, where the desired output is not given, then the learning task is called *unsupervised*. In contrast to supervised problems, the quality of models on unsupervised problems is mostly measured on the cumulative output on all objects. Typically measurements for

unsupervised methods include the information contents, the orthogonality of the constructed components, the statistical independence, the variation explained by the model, the probability that the observed data can be produced by the model (later introduced as *likelihood*), distances between and within clusters, etc. Supervised methods are used for performing prediction of future data while unsupervised methods allow to explore the data, find structure in the data, visualize the data, or compress the data. Unsupervised methods help to understand and explore the data and generate new knowledge but also compress the data for transmission or storage.

Objectives and theoretical concepts of unsupervised learning are maximum likelihood, maximum a posteriori, maximum entropy, expectation maximization, maximal variance, independence, non-Gaussianity, sub- and super-Gaussian distributions, sparse and population codes.

Methods of unsupervised machine learning are projection methods like "principal component analysis", "independent component analysis", "factor analysis", or "projection pursuit". Further clustering methods include "k-means", "hierarchical clustering", "mixture models", or "self-organizing maps". Generative unsupervised methods are density estimation ("kernel density estimation", "Gaussian mixtures"), "hidden Markov models" including factorial HMMs and input-output HMMs, or "belief networks". The latter are subsumed into "Markov networks" or "Markov random fields". Other and more advanced methods include "restricted Boltzmann machines", "neural network auto-associators", and "unsupervised deep neural networks". Unsupervised methods try to extract structure in the data, represent the data in a more compact or more useful way, or build a model of the data generating process or parts thereof.

Projection methods generate a new representation of objects given a representation of them as a feature vector. In most cases, they down-project feature vectors of objects into a lower-dimensional space in order to remove redundancies and components which are not relevant. "Principal Component Analysis" (PCA) represents the object through feature vectors which components give the extension of the data in certain orthogonal directions. The directions are ordered so that the first direction gives the direction of maximal data variance, the second the maximal data variance orthogonal to the first component, and so on. "Independent Component Analysis" (ICA) goes a step further than PCA and represents the objects through feature components which are statistically mutually independent. "Factor Analysis" extends PCA by introducing a Gaussian noise at each original component and assumes Gaussian distribution of the components. "Projection Pursuit" searches for components which are non-Gaussian, therefore, may contain interesting information. Clustering methods are looking for data clusters and, therefore, finding structure in the data. Clustering has been extended to "biclustering", where both the features and the objects are simultaneously clustered (objects are clustered only using features from a particular cluster). "Self-Organizing Maps" (SOMs) are a special kind of clustering methods which also perform a down-projection in order to visualize the data. The down-projection keeps the neighborhood of clusters. Density estimation methods attempt at producing the density from which the data was drawn. In contrast to density estimation methods generative models try to build a model which represents the density of the observed data. Goal is to obtain a world model for which the density of the data points produced by the model matches the observed data density.

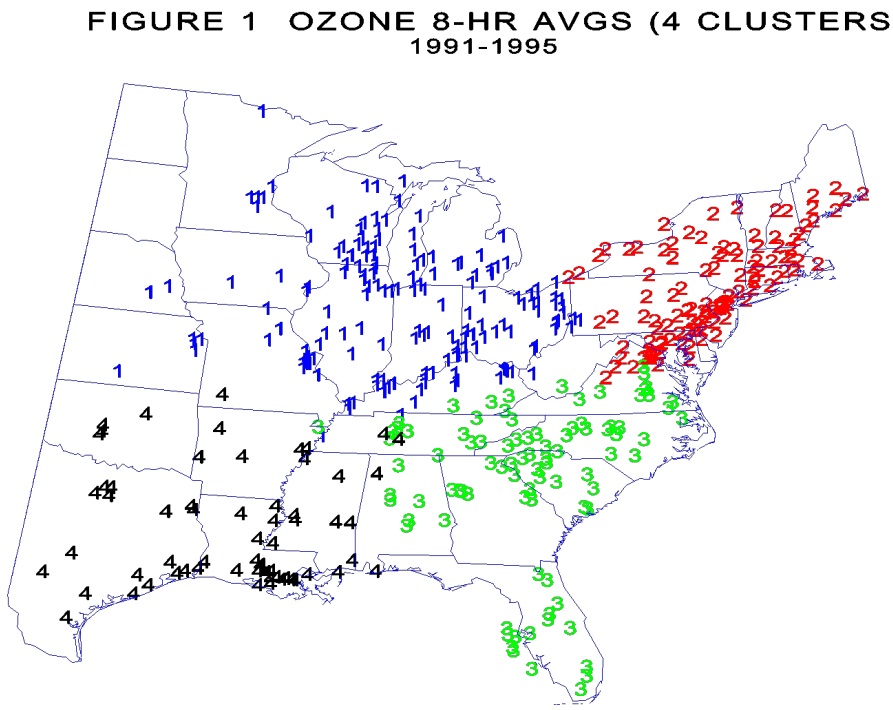


Figure 1.1: Example of a clustering algorithm. Ozone was measured and four clusters with similar ozone were found.

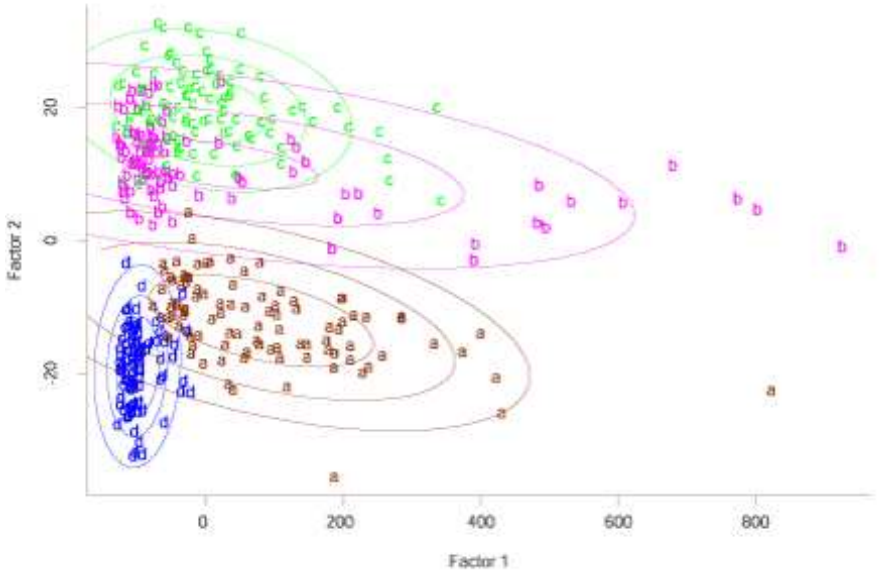


Figure 1.2: Example of a clustering algorithm where the clusters have different shape.

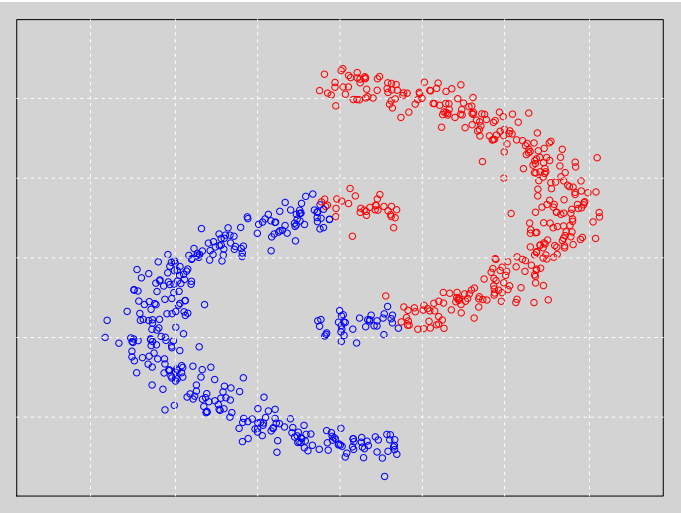


Figure 1.3: Example of a clustering where the clusters have a non-elliptical shape and clustering methods fail to extract the clusters.

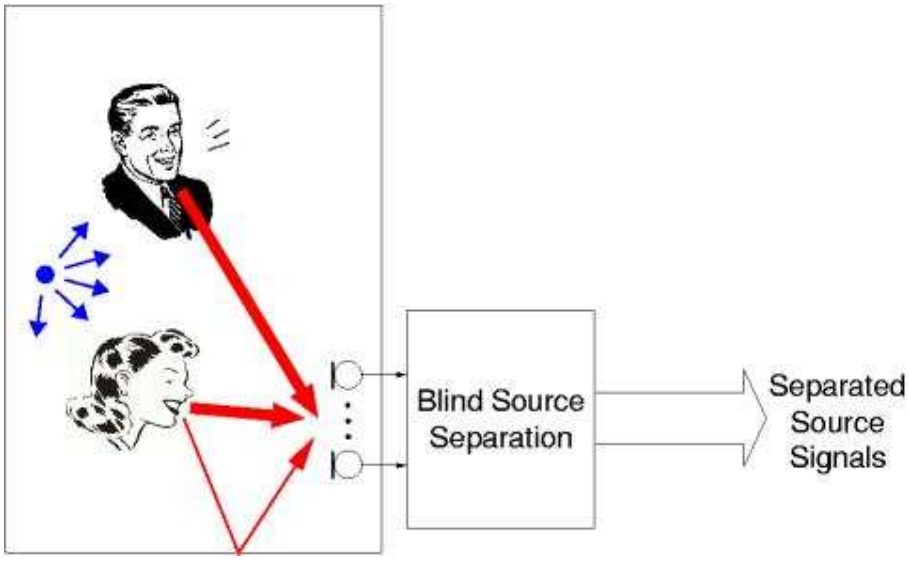


Figure 1.4: Two speakers recorded by two microphones. The speaker produce independent acoustic signals which can be separated by ICA (here called Blind Source Separation) algorithms.

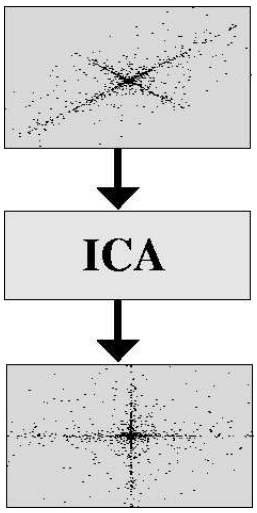


Figure 1.5: On top the data points where the components are correlated: knowing the x-coordinate helps to guess were the y-coordinate is located. The components are statistically dependent. After ICA the components are statistically independent.

1.3 Generative vs. Descriptive Models

In the previous section we mentioned the the kernel density estimator, where the model produces for a location the estimated density. And also for a training data point the density of its location is estimated, i.e. this data point has a new characteristic through the density at its location. We call this a *descriptive* model. Descriptive models supply an additional description of the data point or another representation. Therefore projection methods (PCA, ICA) are descriptive models as the data points are described by certain features (components).

Another machine learning approach to model selection is to model the data generating process. Such models are called *generative models*. Models are selected which produce the distribution observed for the real world data, therefore these models are describing or representing the data generation process. The data generation process may have also input components or random components which drive the process. Such input or random components may be included into the model. Important for the generative approach is to include as much prior knowledge about the world or desired model properties into the model as possible in order to restrict the number of models which can explain the observed data.

A generative model can be used to predict the data generation process for unobserved inputs, to predict the behavior of the data generation process if its parameters are externally changed, to generate artificial training data, or to predict unlikely events. Especially the modeling approaches can give new insights into the working of complex systems of the world like the brain or the cell.

Chapter 2

Basic Terms and Concepts

2.1 Unsupervised Learning in Bioinformatics

One important topic in bioinformatics is to analyze microarray measurements with unsupervised techniques. Goals are to figure out which genes are expressed simultaneously (are part of the same pathway), what metabolic stages are present, etc. Also time series of microarray data must be analyzed e.g. by cluster analysis (e.g. [Eisen et al., 1998]). Fig. 2.1 and Fig. 2.2 show microarray dendrograms obtained through hierarchical clustering.

Often unsupervised methods are used in bioinformatics to visualize dependencies and clusters like the use of principal component analysis for Spellman's cell-cycle data in Fig. 2.3.

For visualization the data must be in general down-projected to show the data in a 2-dimensional or 3-dimensional space.

Dimension reduction is an important step in preprocessing biological data. For example [Lilien et al., 2003] achieved best results on classification of healthy and cancerous persons of prostate cancer on the basis surface-enhanced laser desorption/ionization time-of-flight mass spectrometry (SELDI TOF MS) date if PCA was used.

2.2 Unsupervised Learning Categories

Many unsupervised learning procedures can be viewed as trying to bring two probability distributions into alignment. Two well known classes of unsupervised procedures that can be cast in this manner are *generative* and *recoding* models.

2.2.1 Generative Framework

In a generative unsupervised framework (see Fig. 2.4), the environment generates training examples – which we will refer to as *observations* or training data – by sampling from one distribution; the other distribution is embodied in the model. In the generative framework we want to model or to simulate the real world by generating samples with the same underlying distribution as the real world samples.

The hidden Markov models from previous Chapter 9 fall into the generative framework. Other examples of the generative framework are factor analysis [Jöreskog, 1967, Everitt, 1984, Neal

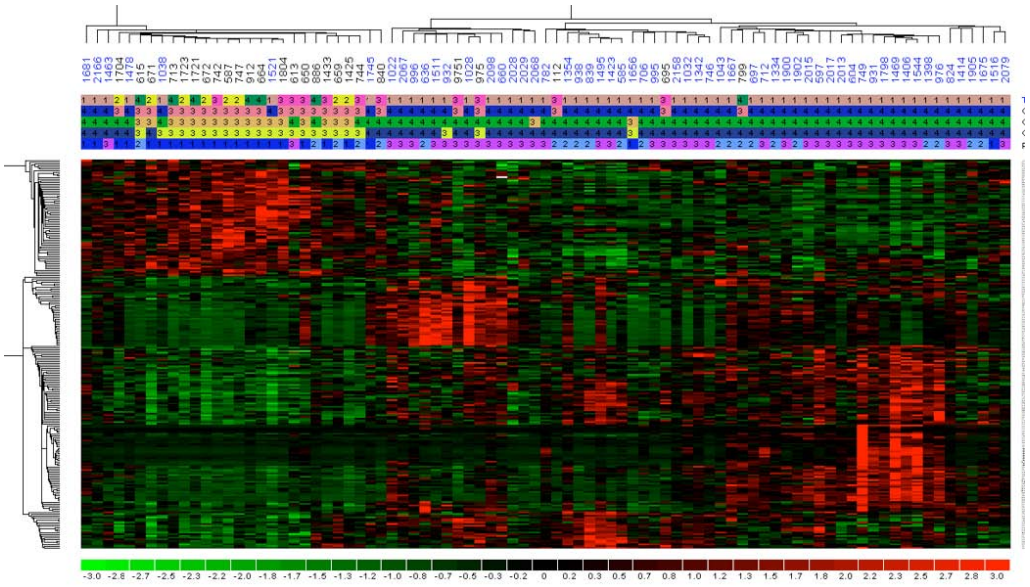


Figure 2.1: A microarray dendrogram obtained by hierarchical clustering.

and Dayan, 1997], Boltzmann machines [Hinton and Sejnowski, 1986], or mixture models. In the generative framework we have to bring two distributions to match: the fixed distribution of observations and the model output distribution.

2.2.2 Recoding Framework

In the recoding unsupervised framework (see Fig. 2.5), the model transforms observations to an output space. The distribution of the transformed observations (the outputs) is compared either to a reference (target) distribution or whether target distribution features are present. These features are measured by an objective function which must be optimized. Target features are used to represent a whole class of target distributions. In many cases the objective function can be replaced by the distribution from the target distribution set which is closest to the output distribution.

Goal of the recoding framework is to represent the observations in an appropriate way, e.g. desired data density, low dimensional representation, high variance (large information), non-Gaussianity, or independent components.

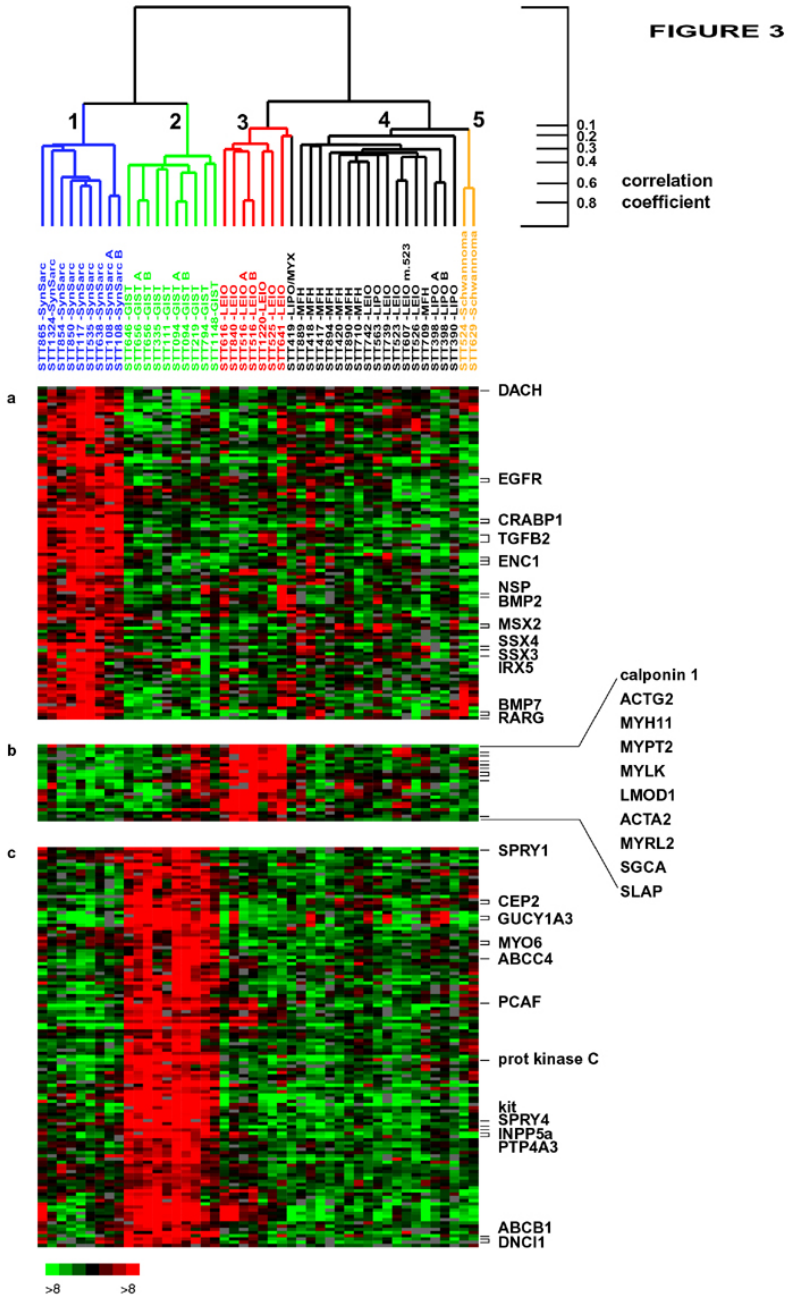


Figure 2.2: Another example of a microarray dendrogram obtained by hierarchical clustering. Representative portions of the tumor specific gene clusters. The spectrum of green to red spots represents the relative centered expression for each gene (sidebar shows fold difference from mean); selected gene names are shown on the right. Correlation coefficient bar shown to the right side of the dendrogram indicates the degree of relatedness between branches of the dendrogram.

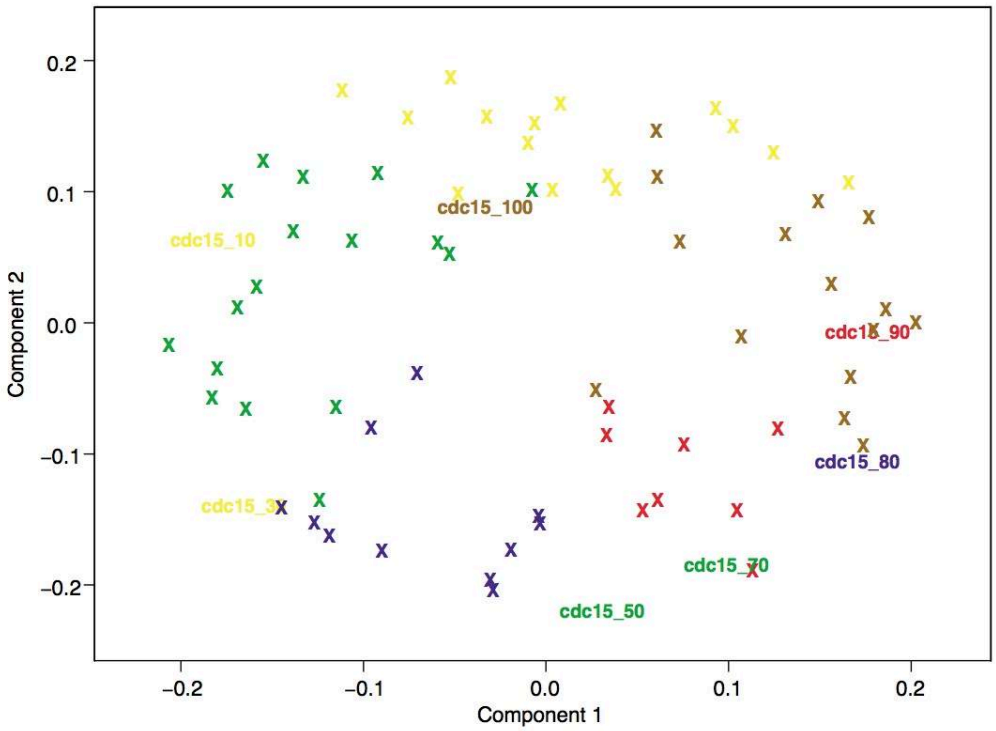


Figure 2.3: Spellman's cell-cycle data represented through the first principal components (Landgrebe et al., Genome Biology, 2002).

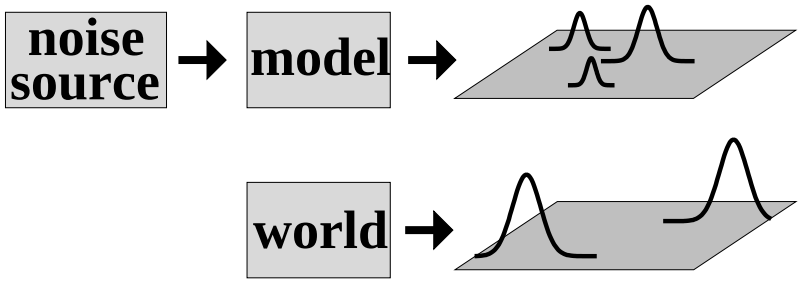


Figure 2.4: The generative framework is depicted. A noise source "drives" the model and produces an output distribution which should match the distribution observed in the world. Separation of model and noise source means that all adjustable parameters are contained in the model.

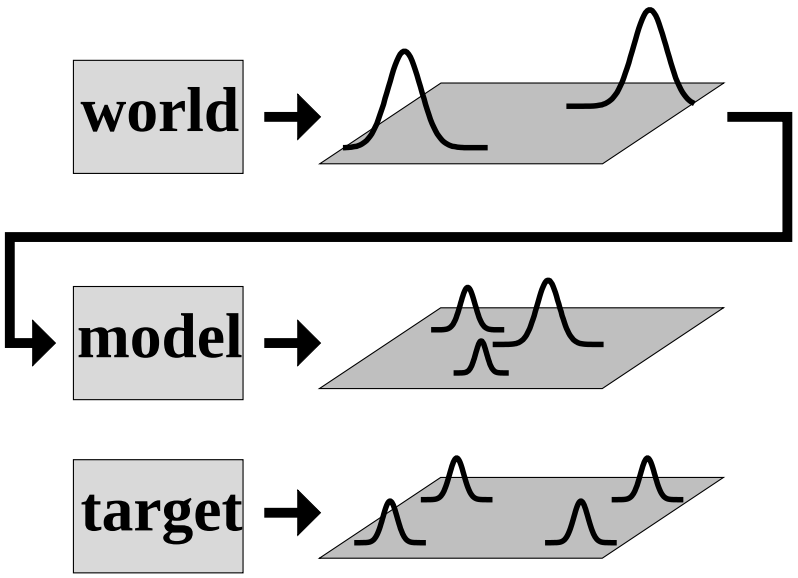


Figure 2.5: The recoding framework is depicted. The data from the world is mapped through a model to a model output distributions which should match a specified target distribution.

Recoding Example: Density Estimation.

Another special case of the recoding unsupervised framework is density estimation, where the reference (target) distribution is easy to calculate like a uniform distribution or a Gaussian distribution. A well known example is the mixtures of Gaussians (MoG) model [Pearson, 1894, Hasselblad, 1966, Duda and Hart, 1973]. The observation transformation must be locally invertible that is the Jacobian determinant must exist for the observation. That allows us to compute the density of the observation. Note that these constraints are weaker than assuming to know the inverse of the transformation. For example with neural networks the Jacobian can be computed [Flake and Pearlmutter, 2000] and, therefore, it is local invertible but the inverse model is unknown.

Other Recoding Approaches.

Projection Pursuit and PCA. Other examples within the recoding framework are projection methods such as *projection pursuit* (e.g., [Friedman and Tukey, 1974, Friedman and Stuetzle, 1981, Huber, 1985, Friedman, 1987, Jones, 1987, Jones and Sibson, 1987, Zhao and Atkeson, 1996, Intrator, 1993]), *principal component analysis (PCA)* (e.g. [Oja, 1982, 1989, Jolliffe, 1986, Jackson, 1991]. Projection pursuit aims at an output distribution which is as non-Gaussian as possible where the non-Gaussianity is measured by the entropy. Note, that for a given variance the Gaussian distribution maximizes the entropy. PCA's objective for a one-dimensional projection is maximal variance. The projection is constrained to a linear mapping, the coefficient vector has unit length, and is orthogonal to previous projections.

Clustering and Coincidence Detection. An objective function can be given for *Clustering* methods and *Coincidence detection* approaches. Clustering methods can be interpreted as mixture models and describe the data by a multimodal distribution. Coincidence detection is based on the fact that the multidimensional distribution of observations has high density regions indicating correlated components in the observations. Therefore coincidence detection is closely related to independent component analysis.

Self-organizing maps (SOMs). For *self-organizing maps (SOMs)* [Kohonen, 1982, 1988, 1990, 1995a, Ritter et al., 1992, 1991, Obermayer et al., 1992, Erwin et al., 1992] the objective function **cannot** always be expressed as a single scalar function (like an energy or error function). Scalar objectives are important to derive learning algorithms based on optimizing this function and to compare models. The objective of SOMs is a scalar function for discrete input spaces and for discrete neighborhood functions otherwise the objective function must be expressed as a vector valued potential function [Kohonen, 1995a, Cottrell et al., 1995, Ritter et al., 1992, 1991, Erwin et al., 1992]. The lack of a scalar objective function is one of the major drawbacks of SOMs because models cannot be compared, overfitting not detected, and stopping of training is difficult to determine.

2.2.3 Recoding and Generative Framework Unified

If the recoding model has a unique inverse then the generative framework can be applied in the recoding context. The inverse model can use the observations as training examples which must be produced from some target distribution. Then the model maps the training examples to target distributions.

If the inverse model is obtained from the generative approach, then the recoding model is also available.

The objective function of principal component analysis (PCA) and independent component analysis (ICA) [Hinton and Sejnowski, 1999, Attias, 1999] attempt at keeping maximal information about the observations in the code while fulfilling certain constraints. In the generative view they are treated as generative models, which try to produce the data, whereas the constraints are coded into the model e.g. by using specific target distributions.

Most recoding methods have higher input dimensionality than output dimensionality because the goal is to represent the input compactly and non-redundantly for visualization or for features extraction.

However, it is sometimes possible to formulate the recoding approach as a generative model even for a non-bijective model. The target distribution has to produce the observations by first generating a code which in turn generates the observations. Computing the likelihood requires the computation of the probabilities of the codes corresponding to the observations. Density estimation, projection pursuit and vector quantization can be treated is such a way [Hinton and Sejnowski, 1999, Attias, 1999]. The probabilities of the codes are the posterior of the code given the data, where we assume that an observation can be produced by different codes. For example, recoding methods which do not have bijective functions are *principal curves* [Mulier and Cherkassky, 1995, Ritter et al., 1992], which are a nonlinear generalization of principal components [Hastie and Stuetzle, 1989].

Example: Independent Component Analysis. An example for a recoding model treated as generative model and using objective functions is independent component analysis (ICA) [Schmidhuber, 1992, Bell and Sejnowski, 1995, Deco and Brauer, 1995, Pearlmutter and Parra, 1997, Cardoso and Laheld, 1996, Cichocki et al., 1994, Jutten and Herault, 1991, Comon, 1994, Yang and Amari, 1997, Yang et al., 1996, Cardoso and Souloumiac, 1993, Hyvärinen, 1999], a method that discovers a representation of vector-valued observations in which the statistical dependence

among the vector elements in the output space is minimized. With ICA, the model de-mixes observation vectors and the output should consist of statistically independent components. Towards this end the output distribution can be compared against a given factorial distribution. The generative approach assumes that the observations are produced by a model with independent hidden units [Cardoso, 1997, Moulines et al., 1997, Yang and Amari, 1997, Pearlmutter and Parra, 1997]. Alternatively, an objective or contrast function indicating statistically independent model output components can be used (see e.g. [Hyvärinen, 1999]).

Example: Density Estimation. The traditional density estimation framework supplies the inverse model and, therefore, can be viewed as a generative framework. For example in the mixture of Gaussians model each mixture component can be viewed as being generated by a Gaussian distribution with identity matrix as covariance matrix and thereafter transformed by a non-singular linear mapping. The posterior can be easily computed thus the model can serve as data generation model.

2.3 Quality of Parameter Estimation

Generative models estimate the optimal parameter given a parametrized model class. The given data is generated from a model of the model class. Goal is to find this model or approximate it closely in terms of the parameters (a model with similar parameters).

We consider the quality of parameter estimation in the generative framework. The true parameter is assumed to be known and an estimation method aims to estimate the true parameter from given training data. The difference between true parameter and estimated parameter determines the quality of the estimation technique.

The *training data* is $\{\boldsymbol{x}\} = \left\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^l\right\}$ for which we will often simply write $\boldsymbol{X}$ (the matrix of training data). The true parameter vector is denoted by $\boldsymbol{w}$ and its estimate by $\hat{\boldsymbol{w}}$.

An estimator is *unbiased* if

$$\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) = \boldsymbol{w}\,, \tag{2.1}$$

i.e. on the average the estimator will yield the true parameter.

The *bias* is

$$b(\hat{\boldsymbol{w}}) = \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) - \boldsymbol{w}\,. \tag{2.2}$$

The *variance* of the estimator is defined as

$$\mathrm{var}(\hat{\boldsymbol{w}}) = \mathrm{E}_{\boldsymbol{X}}\left((\hat{\boldsymbol{w}} - \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}))^T (\hat{\boldsymbol{w}} - \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}))\right)\,. \tag{2.3}$$

An evaluation criterion for supervised methods is the *mean squared error* (MSE) which is an expectation over future data points. Here we define the MSE as expectation over the training set, because we deal with unsupervised learning and evaluate the estimator. The MSE gives the expected error as squared distance between the estimated parameter and the true parameter.

$$\mathrm{mse}(\hat{\boldsymbol{w}}) = \mathrm{E}_{\boldsymbol{X}}\left((\hat{\boldsymbol{w}} - \boldsymbol{w})^T (\hat{\boldsymbol{w}} - \boldsymbol{w})\right)\,. \tag{2.4}$$

We can reformulate the MSE:

$$\begin{aligned}\mathrm{mse}(\hat{\boldsymbol{w}}) \;=\; \mathrm{E}_{\boldsymbol{X}}\left((\hat{\boldsymbol{w}} \;-\; \boldsymbol{w})^T\,(\hat{\boldsymbol{w}} \;-\; \boldsymbol{w})\right) \;= \\ \mathrm{E}_{\boldsymbol{X}}\Big(((\hat{\boldsymbol{w}} \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}})) \;+\; (\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w}))^T \\ ((\hat{\boldsymbol{w}} \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}})) \;+\; (\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w}))\Big) \;= \\ \mathrm{E}_{\boldsymbol{X}}\Big((\hat{\boldsymbol{w}} \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}))^T\,(\hat{\boldsymbol{w}} \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}})) \;- \\ 2\;(\hat{\boldsymbol{w}} \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}))^T\,(\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w}) \;+ \\ (\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w})^T\,(\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w})\Big) \;= \\ \mathrm{E}_{\boldsymbol{X}}\left((\hat{\boldsymbol{w}} \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}))^T\,(\hat{\boldsymbol{w}} \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}))\right) \;+ \\ (\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w})^T\,(\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w}) \;= \\ \mathrm{var}(\hat{\boldsymbol{w}}) \;+\; b^2(\hat{\boldsymbol{w}})\;.\end{aligned} \tag{2.5}$$

where the last but one equality comes from the fact that only $\hat{\boldsymbol{w}}$ depends on $\boldsymbol{X}$ and therefore

$$\begin{aligned}\mathrm{E}_{\boldsymbol{X}}\left((\hat{\boldsymbol{w}} \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}))^T\,(\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w})\right) \;= \\ (\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}))^T\,(\mathrm{E}_{\boldsymbol{X}}(\hat{\boldsymbol{w}}) \;-\; \boldsymbol{w}) \;=\; 0\,.\end{aligned} \tag{2.6}$$

The MSE is decomposed into a variance term $\mathrm{var}(\hat{\boldsymbol{w}})$ and a bias term $b^2(\hat{\boldsymbol{w}})$. The variance has high impact on the performance because large deviations from the true parameter have strong influence on the MSE through the quadratic error term.

Note that averaging linearly reduces the variance. The average is

$$\hat{\boldsymbol{w}}_{a_N} \;=\; \frac{1}{N}\sum_{i=1}^{N}\hat{\boldsymbol{w}}_i\;, \tag{2.7}$$

where

$$\begin{aligned}\hat{\boldsymbol{w}}_i \;&=\; \hat{\boldsymbol{w}}_i\,(\boldsymbol{X}_i) \\ \boldsymbol{X}_i \;&=\; \left\{\boldsymbol{x}^{(i-1)\;l/N\;+\;1},\ldots,\boldsymbol{x}^{i\;l/N}\right\}\;,\end{aligned} \tag{2.8}$$

i.e. $\boldsymbol{X}_i$ is the i-th subset of $\boldsymbol{X}$ and contains l/N elements. The size of the data is l and the examples of $\boldsymbol{X}_i$ range from $(i-1)\;l/N \;+\; 1$ to $i\;l/N$.

The average is unbiased:

$$\mathrm{E}_{\boldsymbol{X}}\left(\hat{\boldsymbol{w}}_{a_N}\right) \;=\; \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}_{\boldsymbol{X}_i}\hat{\boldsymbol{w}}_i \;=\; \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{w} \;=\; \boldsymbol{w}\;. \tag{2.9}$$

The variance is linearly reduced

$$\begin{aligned}\mathrm{covar}_{\boldsymbol{X}}\left(\hat{\boldsymbol{w}}_{a_N}\right) \;=\; \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{covar}_{\boldsymbol{X}_i}\left(\hat{\boldsymbol{w}}_i\right) \;= \\ \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{covar}_{\boldsymbol{X},l/N}\left(\hat{\boldsymbol{w}}\right) \;=\; \frac{1}{N}\mathrm{covar}_{\boldsymbol{X},l/N}\left(\hat{\boldsymbol{w}}\right)\;,\end{aligned} \tag{2.10}$$

where $\mathrm{covar}_{\boldsymbol{X},l/N}\left(\hat{\boldsymbol{w}}\right)$ is the estimator with l/N training points.

We used the facts:

$$\begin{aligned} &\mathrm{covar}_{\boldsymbol{X}}(\boldsymbol{a} + \boldsymbol{b}) = \\ &\mathrm{covar}_{\boldsymbol{X}}(\boldsymbol{a}) + \mathrm{covar}_{\boldsymbol{X}}(\boldsymbol{b}) + 2\,\mathrm{covar}_{\boldsymbol{X}}(\boldsymbol{a}, \boldsymbol{b}) \\ &\mathrm{covar}_{\boldsymbol{X}}(\lambda\,\boldsymbol{a}) = \lambda^2\,\mathrm{covar}_{\boldsymbol{X}}(\boldsymbol{a})\,. \end{aligned} \tag{2.11}$$

For averaging it is important that the training sets $\boldsymbol{X}_i$ are independent from each other and do not overlap. Otherwise the estimators are dependent and the covariance terms between the estimators do not vanish.

2.4 Maximum Likelihood Estimator

A very popular parameter estimator for generative models is the Maximum Likelihood Estimator (MLE). The popularity stems from the fact that it can be applied to a broad range of problems and that the MLE is asymptotically efficient and unbiased. That means the MLE does everything right and efficiently extracts information from the data.

The likelihood $\mathcal{L}$ of the data set $\{\boldsymbol{x}\} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^l\}$ is

$$\mathcal{L}(\{\boldsymbol{x}\}; \boldsymbol{w}) = p(\{\boldsymbol{x}\}; \boldsymbol{w})\,, \tag{2.12}$$

i.e. the probability of the model $p(\boldsymbol{x}; \boldsymbol{w})$ to produce the data set. However the set $\{\boldsymbol{x}\}$ has zero measure and therefore the density at the data set $\{\boldsymbol{x}\}$ must be used.

For identical and independently distributed (iid) data the likelihood is

$$\mathcal{L}(\{\boldsymbol{x}\}; \boldsymbol{w}) = p(\{\boldsymbol{x}\}; \boldsymbol{w}) = \prod_{i=1}^{l} p(\boldsymbol{x}^i; \boldsymbol{w})\,. \tag{2.13}$$

Instead of maximizing the likelihood $\mathcal{L}$ the log-likelihood $\ln \mathcal{L}$ is maximized or the negative log-likelihood $-\ln \mathcal{L}$ is minimized. The logarithm transforms the product of the iid data sampling into a sum:

$$-\ln \mathcal{L}(\{\boldsymbol{x}\}; \boldsymbol{w}) = -\sum_{i=1}^{l} \ln p(\boldsymbol{x}^i; \boldsymbol{w})\,. \tag{2.14}$$

Problem with the likelihood is that it is a product of densities at finite many points, therefore, it is a set with zero measure. In statistics, sets with zero measures are ignored as they vanish in all integrals, that is, in expectations. To motivate the use of the likelihood one can assume that if point $p(\boldsymbol{x}^i; \boldsymbol{w})$ is considered actually $p(\boldsymbol{x}^i; \boldsymbol{w})\,d\boldsymbol{x}$ is meant, which gives the probability mass and a probability of observing $\boldsymbol{x}$ in a region of volume $d\boldsymbol{x}$ around $\boldsymbol{x}^i$. In this case the likelihood gives the probability of the model to produce similar data points as $\{\boldsymbol{x}\}$, where similar means data points in a volume $d\boldsymbol{x}$ around the actual observed data points.

However the fact that the MLE is so popular is based on its simple use and its properties especially that it is optimal for the number of training points going to infinity.

An unbiased estimator is said to be *efficient* if it reaches the Cramer-Rao Lower Bound (CRLB):

$$\mathrm{covar}(\hat{\boldsymbol{w}}) \; - \; \boldsymbol{I}_F^{-1}(\boldsymbol{w}) \tag{2.15}$$

is positive definite:

$$\mathrm{covar}(\hat{\boldsymbol{w}}) \; - \; \boldsymbol{I}_F^{-1}(\boldsymbol{w}) \; \geq \; \mathbf{0} \, , \tag{2.16}$$

$\boldsymbol{I}_F$ is the *Fisher information matrix. An efficient estimator has the minimal variance of all unbiased estimators.* It is efficient in that it efficiently makes use of the data and extracts information to estimate the parameter.

A estimator is said to be *consistent* if

$$\hat{\boldsymbol{w}} \; \overset{l \to \infty}{\to} \; \boldsymbol{w} \, , \tag{2.17}$$

i.e. for large training sets the estimator approaches the true value. Consistency is important to ensure that more training examples lead to better models. How fast the estimator converges to the best model with more training examples may be different for each consistent estimator.

Properties of maximum likelihood estimator:

- the MLE is invariant under parameter change,
- the MLE is asymptotically unbiased,
- the MLE is asymptotically efficient, i.e. asymptotically optimal,
- the MLE is consistent for zero Cramer-Rao lower bound (CRLB).

2.5 Expectation Maximization

The likelihood can be maximized by gradient descent methods. However the likelihood must be expressed analytically to obtain the derivatives. For some models the likelihood cannot be computed analytically because of hidden states of the model, of a many-to-one output mapping of the model, or of non-linearities. Often the model has unobserved variables $\boldsymbol{u}$, i.e. *hidden variables* or *latent variables*. For models with hidden variables the likelihood is determined by all possible values of the hidden variables which can produce output $\boldsymbol{x}$.

For many models the joint probability $p(\{\boldsymbol{x}\}, \boldsymbol{u}; \boldsymbol{w})$ of the hidden variables $\boldsymbol{u}$ and observations $\{\boldsymbol{x}\}$ is easier to compute than the likelihood of the observations. If we can also estimate $p(\boldsymbol{u} \mid \{\boldsymbol{x}\}; \boldsymbol{w})$ of the hidden variables $\boldsymbol{u}$ using the parameters $\boldsymbol{w}$ and given the observations $\{\boldsymbol{x}\}$ then we can apply the Expectation Maximization (EM) algorithm.

Let us assume we have an estimation $Q(\boldsymbol{u} \mid \{\boldsymbol{x}\})$ for $p(\boldsymbol{u} \mid \{\boldsymbol{x}\}; \boldsymbol{w})$, which is some density

with respect to $\boldsymbol{u}$. The following inequality is the basis for the EM algorithm:

$$\begin{aligned}
\ln \mathcal{L}(\{\boldsymbol{x}\};\boldsymbol{w}) \;&=\; \ln p(\{\boldsymbol{x}\};\boldsymbol{w}) \;= \\
&\ln \int_U p(\{\boldsymbol{x}\},\boldsymbol{u};\boldsymbol{w})\, d\boldsymbol{u} \;= \\
&\ln \int_U \frac{Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})}{Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})} p(\{\boldsymbol{x}\},\boldsymbol{u};\boldsymbol{w})\, d\boldsymbol{u} \;\geq \\
&\int_U Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})\; \ln \frac{p(\{\boldsymbol{x}\},\boldsymbol{u};\boldsymbol{w})}{Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})}\, d\boldsymbol{u} \;= \\
&\int_U Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})\; \ln p(\{\boldsymbol{x}\},\boldsymbol{u};\boldsymbol{w})\, d\boldsymbol{u} - \\
&\int_U Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})\; \ln Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})\, d\boldsymbol{u} \;= \\
&\mathcal{F}(Q,\boldsymbol{w})\;.
\end{aligned} \tag{2.18}$$

where the "$\geq$" is the application of Jensen's inequality. Above inequality states that $\mathcal{F}(Q,\boldsymbol{w})$ is a lower bound to the log-likelihood $\ln \mathcal{L}(\{\boldsymbol{x}\};\boldsymbol{w})$.

The EM algorithm is an iteration between two steps, the "E"-step and the "M"-step:

$$\begin{aligned}
&\textbf{E-step:} \\
&Q_{k+1} \;=\; \arg\max_{Q}\; \mathcal{F}(Q,\boldsymbol{w}_k) \\
&\textbf{M-step:} \\
&\boldsymbol{w}_{k+1} \;=\; \arg\max_{\boldsymbol{w}}\; \mathcal{F}(Q_{k+1},\boldsymbol{w})\;.
\end{aligned} \tag{2.19}$$

It is important to note that in the E-step the maximal Q is

$$\begin{aligned}
Q_{k+1}(\boldsymbol{u}\mid\{\boldsymbol{x}\}) \;&=\; p(\boldsymbol{u}\mid\{\boldsymbol{x}\};\boldsymbol{w}_k) \\
\mathcal{F}(Q_{k+1},\boldsymbol{w}_k) \;&=\; \ln \mathcal{L}(\{\boldsymbol{x}\};\boldsymbol{w}_k)\;.
\end{aligned} \tag{2.20}$$

This means that the maximal Q is the posterior or the hidden variables $p(\boldsymbol{u}\mid\{\boldsymbol{x}\};\boldsymbol{w}_k)$ using the current parameters $\boldsymbol{w}_k$. Furthermore, the lower bound F is equal to the log-likelihood with the current parameters $\boldsymbol{w}_k$. **The bound is thight and reaches the log-likelihood.**

To see the last statement:

$$p(\boldsymbol{u},\{\boldsymbol{x}\};\boldsymbol{w}_k) \;=\; p(\boldsymbol{u}\mid\{\boldsymbol{x}\};\boldsymbol{w}_k)\, p(\{\boldsymbol{x}\};\boldsymbol{w}_k)\;, \tag{2.21}$$

therefore

$$\begin{aligned}
\mathcal{F}(Q,\boldsymbol{w}) \;&=\; \int_U Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})\; \ln \frac{p(\{\boldsymbol{x}\},\boldsymbol{u};\boldsymbol{w})}{Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})}\, d\boldsymbol{u} \;= \\
&\int_U Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})\; \ln \frac{p(\boldsymbol{u}\mid\{\boldsymbol{x}\};\boldsymbol{w})}{Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})}\, d\boldsymbol{u} \;+\; \ln p(\{\boldsymbol{x}\};\boldsymbol{w}) \;= \\
&-\int_U Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})\; \ln \frac{Q(\boldsymbol{u}\mid\{\boldsymbol{x}\})}{p(\boldsymbol{u}\mid\{\boldsymbol{x}\};\boldsymbol{w})}\, d\boldsymbol{u} \;+\; \ln \mathcal{L}(\{\boldsymbol{x}\};\boldsymbol{w})
\end{aligned} \tag{2.22}$$

The expression $\int_U Q(\boldsymbol{u} \mid \{\boldsymbol{x}\}) \ln \frac{Q(\boldsymbol{u}|\{\boldsymbol{x}\})}{p(\boldsymbol{u}|\{\boldsymbol{x}\};\boldsymbol{w})} \, d\boldsymbol{u}$ is the Kullback-Leibler divergence $D_{\mathrm{KL}}(Q \parallel p)$ between $Q(\boldsymbol{u} \mid \{\boldsymbol{x}\})$ and $p(\boldsymbol{u} \mid \{\boldsymbol{x}\}; \boldsymbol{w})$. The Kullback-Leibler divergence $\mathrm{KL}(p_1, p_2)$ is defined as

$$D_{\mathrm{KL}}(p_1 \parallel p_2) = \int_U p_1(\boldsymbol{u}) \ln \frac{p_1(\boldsymbol{u})}{p_2(\boldsymbol{u})} \, d\boldsymbol{u} \tag{2.23}$$

and the *cross entropy* as

$$- \int_U p_1(\boldsymbol{u}) \ln p_2(\boldsymbol{u}) \, d\boldsymbol{u} \, . \tag{2.24}$$

The *entropy* of a distribution p_1 is defined as

$$- \int_U p_1(\boldsymbol{u}) \ln p_1(\boldsymbol{u}) \, d\boldsymbol{u} \, . \tag{2.25}$$

The Kullback-Leibler divergence is always greater than or equal to zero:

$$D_{\mathrm{KL}}(p_1 \parallel p_2) \geq 0 \tag{2.26}$$

because

$$\begin{aligned} 0 = \ln 1 = \ln \int_U p_2(\boldsymbol{u}) \, d\boldsymbol{u} = \\ \ln \int_U p_1(\boldsymbol{u}) \frac{p_2(\boldsymbol{u})}{p_1(\boldsymbol{u})} \, d\boldsymbol{u} \geq \\ \int_U p_1(\boldsymbol{u}) \ln \frac{p_2(\boldsymbol{u})}{p_1(\boldsymbol{u})} \, d\boldsymbol{u} = - D_{\mathrm{KL}}(p_1 \parallel p_2) \, . \end{aligned} \tag{2.27}$$

Thus, if $D_{\mathrm{KL}}(Q \parallel p) = 0$ then $\mathcal{F}(Q, \boldsymbol{w}_k)$ is maximized because the Kullback-Leibler divergence, which enters the equation with a negative sign, is minimal. We have $Q(\boldsymbol{u} \mid \{\boldsymbol{x}\}) = p(\boldsymbol{u} \mid \{\boldsymbol{x}\}; \boldsymbol{w})$ and obtain

$$\mathcal{F}(Q, \boldsymbol{w}) = \ln \mathcal{L}(\{\boldsymbol{x}\}; \boldsymbol{w}) \, . \tag{2.28}$$

In the M-step only the cross-entropy $\int_U Q_{k+1}(\boldsymbol{u} \mid \{\boldsymbol{x}\}) \ln p(\{\boldsymbol{x}\}, \boldsymbol{u}; \boldsymbol{w}) \, d\boldsymbol{u}$ must be considered because the other term (the entropy of Q_{k+1}) is independent of the parameters $\boldsymbol{w}$. The EM algorithm can be interpreted as:

- E-step: Tighten the lower bound to equality: $\mathcal{F}(Q, \boldsymbol{w}) = \ln \mathcal{L}(\{\boldsymbol{x}\}; \boldsymbol{w})$.
- M-step: Maximize the lower bound which is at the equality and therefore increase the likelihood. This might lead to a lower bound which is no longer tight.

The EM algorithm increases the lower bound because in both steps the lower bound is maximized. Can it happen that maximizing the lower bound may decrease the likelihood? No! At the beginning of the M-step we have $\mathcal{F}(Q_{k+1}, \boldsymbol{w}_k) = \ln \mathcal{L}(\{\boldsymbol{x}\}; \boldsymbol{w}_k)$, and the E-step does not change the parameters $\boldsymbol{w}$:

$$\begin{aligned} \ln \mathcal{L}(\{\boldsymbol{x}\}; \boldsymbol{w}_k) = \mathcal{F}(Q_{k+1}, \boldsymbol{w}_k) \leq \\ \mathcal{F}(Q_{k+1}, \boldsymbol{w}_{k+1}) \leq \mathcal{F}(Q_{k+2}, \boldsymbol{w}_{k+1}) = \ln \mathcal{L}(\{\boldsymbol{x}\}; \boldsymbol{w}_{k+1}) \, , \end{aligned} \tag{2.29}$$

where the first "$\leq$" is from the M-step which gives w_{k+1} and the second "$\leq$" from the E-step which gives Q_{k+2}.

2.6 Maximum Entropy

A *maximum entropy probability distribution* is the distribution with maximal entropy given a class of distributions. If any prior knowledge is missing except that a distribution a certain class, then maximum entropy distribution should be chosen because

- it has minimal prior assumptions on the distribution and
- physical systems converge over time to maximal entropy configurations which makes it the most likely observed solution.

The *principle of maximum entropy* was first expounded by E.T. Jaynes in 1957, where he emphasized a natural correspondence between statistical mechanics and information theory.

For discrete random variables $p_k = p(x = x_k)$, the entropy of is defined as

$$H \;=\; -\sum_{k\geq 1} p_k \log p_k \; . \tag{2.30}$$

We assume $p_k log p_k = 0$ for $p_k = 0$. For continuous random variables x with probability density $p(x)$, the entropy is

$$H \;=\; -\int_{-\infty}^{\infty} p(x) \log p(x)\; dx \; , \tag{2.31}$$

where we set $p(x) log p(x) = 0$ for $p(x) = 0$.

The **normal distribution** $N(\mu, \sigma^2)$ has maximum entropy among all real-valued distributions with mean μ and standard deviation σ. Normality imposes the minimal prior assumptions given the first two moments. The **uniform distribution** on the interval $[a, b]$ is the maximum entropy distribution among all continuous distributions which are supported in the interval $[a, b]$. The **exponential distribution** with mean $1/\lambda$ is the maximum entropy distribution among all continuous distributions supported in $[0, \infty]$ that have a mean $1/\lambda$.

Discrete random variables X which satisfy the n conditions:

$$\mathrm{E}(f_j(X)) \;=\; a_j \quad \text{for } j = 1, \ldots, n \tag{2.32}$$

Maximum entropy distribution has the following shape:

$$\Pr(X = x_k) \;=\; c\, \exp\left(\sum_{j=1}^{n} \lambda_j\, f_j(x_k)\right) \quad \text{for } k = 1, 2, \ldots \; , \tag{2.33}$$

where the constants c and λ_j have to be determined so that the sum of the probabilities is 1 and the above conditions for the expected values are satisfied. Conversely, if constants c and λ_j as above can be found, then the above distribution is the maximum entropy distribution.

Not all classes of distributions contain a maximum entropy distribution. A class may contain distributions of arbitrarily large entropy (e.g. the class of all continuous distributions on $\mathbb{R}$ with mean 0 but arbitrary standard deviation). Or the entropies of distributions from a class are bounded from above but there is no distribution which attains the maximal entropy (e.g. the class of all

continuous distributions X on $\mathbb{R}$ with $\mathrm{E}(X) = 0$ and $\mathrm{E}(X^2) = \mathrm{E}(X^3) = 1$. The expected value restrictions for the class C may force the probability distribution to be zero in certain subsets of S. In that case Boltzmann's theorem does not apply, but S can be shrinked.

SOLUTION
We require our probability distribution p to satisfy

$$\sum_{i=1}^{n} p(x_i)\, f_k(x_i) \;=\; F_k \qquad k = 1, \ldots, m\,. \tag{2.34}$$

Furthermore, the probability must sum to one:

$$\sum_{i=1}^{n} p(x_i) \;=\; 1\,. \tag{2.35}$$

The probability distribution with maximum information entropy subject to these constraints is

$$p(x_i) \;=\; \frac{1}{Z(\lambda_1, \ldots, \lambda_m)}\, \exp\left(\lambda_1 f_1(x_i) \;+\; \cdots \;+\; \lambda_m f_m(x_i)\right)\,. \tag{2.36}$$

This distribution is called the *Gibbs distribution* in statistical mechanics. The normalization constant Z is determined by

$$Z(\lambda_1, \ldots, \lambda_m) \;=\; \sum_{i=1}^{n} \exp\left(\lambda_1 f_1(x_i) \;+\; \cdots \;+\; \lambda_m f_m(x_i)\right)\,, \tag{2.37}$$

and is called the *partition function*. The value of the Lagrange multipliers λ_k are determined by

$$F_k \;=\; \frac{\partial}{\partial \lambda_k} \log Z(\lambda_1, \ldots, \lambda_m)\,. \tag{2.38}$$

These m simultaneous equations may not have a closed form solution but can be solved numerically.

The derivatives are:

$$\begin{aligned} \frac{\partial}{\partial \lambda_k} \log Z(\lambda_1, \ldots, \lambda_m) \;&=\; \frac{1}{Z(\lambda_1, \ldots, \lambda_m)} \sum_{i=1}^{n} f_k(x_i) \exp\left(\lambda_1 f_1(x_i) \;+\; \cdots \;+\; \lambda_m f_m(x_i)\right) \\ &=\; \sum_{i=1}^{n} p(x_i)\, f_k(x_i)\,, \end{aligned} \tag{2.39}$$

which leads to the orgiginal contraints.

Chapter 3

Principal Component Analysis

Principal Component Analysis (PCA) Jöreskog [1967], Everitt [1984], Neal and Dayan [1997] also known as *Karhunen-Loéve transform* (KTL) or as *Hotelling transform* makes a transformation of the coordinate system so that the data has largest variance along the first coordinate, the second largest data variance is along the second coordinate, etc. The coordinates, that are vectors, are called *principal components*. Fig. 3.1 shows the principal components of a two-dimensional data set and Fig. 3.2 shows how the projection onto the first principal component is extracted from data points.

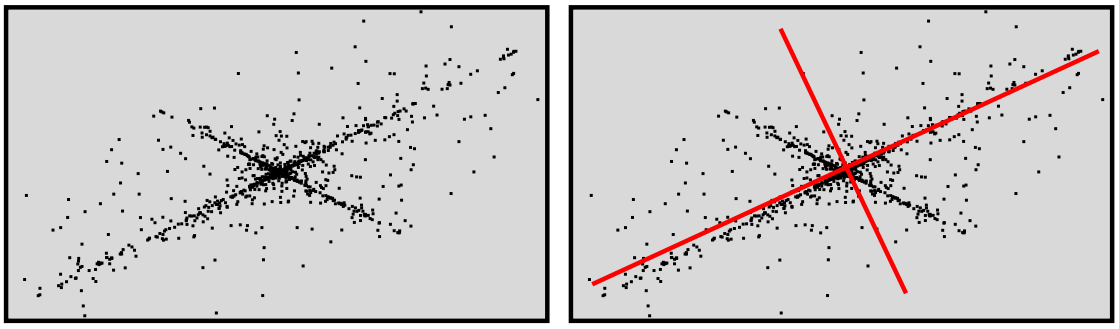


Figure 3.1: Principal component analysis for a two-dimensional data set. Left: the original data set. Right: The first principal component is shown as the line from lower left to upper right. The second principal component is orthogonal to the first component.

PCA is a very useful tool to summarize multivariate data because the first principal components capture most of the variation in the data. Therefore, they capture the most prominent structures in the data. Plotting observations by their projection onto the first two principal components often gives a first insight into the nature of the data.

Instead of a single scalar x, an observation is now represented as a vector $\boldsymbol{x}$ of m features: $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$. The data consisting of n observations $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ can be summarized in a data matrix $\boldsymbol{X}$ of size $n \times m$ which means n rows and m columns. The rows of $\boldsymbol{X}$ contain the n observations (each row contains one observation), while the columns contain the m features. We assume that the columns of $\boldsymbol{X}$, that are the features, have zero sample mean. Otherwise, the feature mean must be subtracted from each feature of each observation.

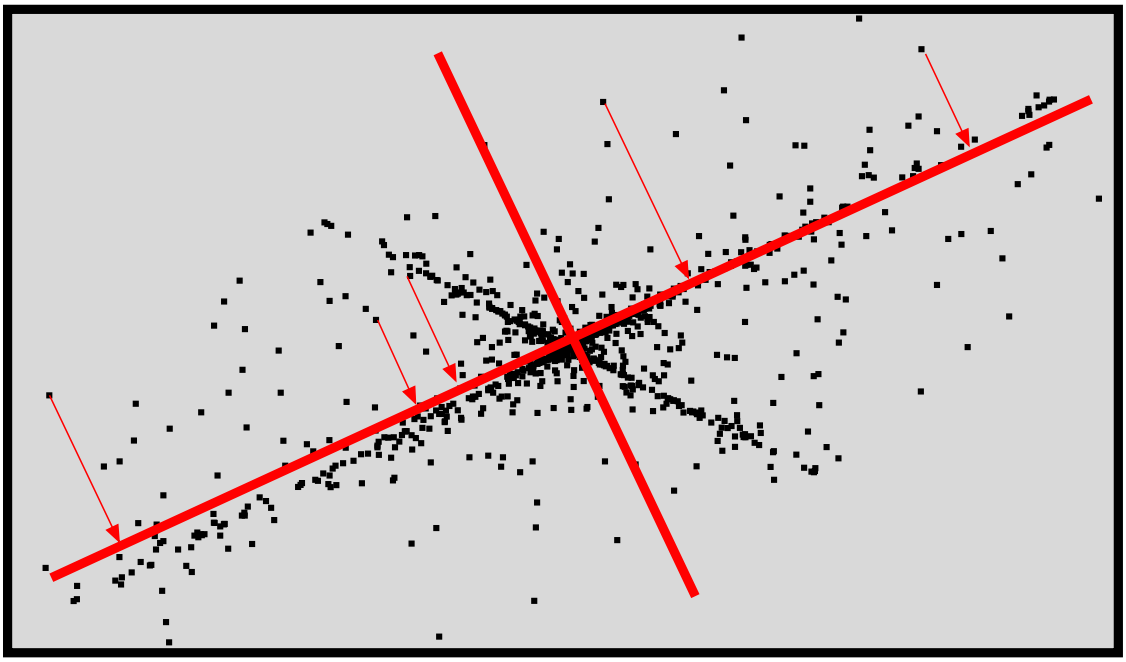


Figure 3.2: Principal component analysis for a two-dimensional data set. The projection onto the first principal component is extracted for data points. The points are projected onto the first component and then the distance to the origin is measured.

3.1 The Method

The $m \times m$ *sample covariance matrix* $\boldsymbol{C}$ of the features across the observations is defines as

$$C_{st} = \frac{1}{n} \sum_{i=1}^{n} x_{is}\, x_{it} \,, \tag{3.1}$$

where $x_{is} = (\boldsymbol{x}_i)_s$ and $x_{it} = (\boldsymbol{x}_i)_t$. For an unbiased estimate of the covariance matrix a factor $\frac{1}{n-1}$ instead of $\frac{1}{n}$ should be used. The covariance matrix $\boldsymbol{C}$ can be expressed as

$$\boldsymbol{C} = \frac{1}{n}\, \boldsymbol{X}^T \boldsymbol{X} = \frac{1}{n}\, \boldsymbol{U} \boldsymbol{D}_m\, \boldsymbol{U}^T \,, \tag{3.2}$$

where $\boldsymbol{U}$ is an orthogonal $m \times m$ matrix and $\boldsymbol{D}_m$ is an $m \times m$ diagonal matrix. This decomposition of $\boldsymbol{C}$ into $\boldsymbol{U}$ and $\boldsymbol{D}_m$ is the *eigendecomposition* or *spectral decomposition* of $\boldsymbol{C}$. This decomposition exists because $\boldsymbol{C}$ is a symmetric positive definite matrix. The diagonal entries of $\boldsymbol{D}_m$ are called *eigenvalues* and the column vectors $\boldsymbol{u}_i = [\boldsymbol{U}]_i$ are called *eigenvectors*. We assume that the eigenvalues of $\boldsymbol{D}_m$ are sorted decreasingly, so that the first value is the largest eigenvalue. $\boldsymbol{C}$ as a symmetric real matrix is always diagonalizable and, since it is positive definite, its eigenvalues are larger than or equal to zero. In the context of PCA, the eigenvectors $\boldsymbol{u}_i$ are called the *principal components*, where the first principal component corresponds to the largest eigenvalue.

We assume that $n \geq m$ and that at least m linear independent observations exist, in order to ensure that $\boldsymbol{C}$ has full rang. To ensure $n \geq m$, typically feature selection is performed prior

to a PCA analysis. Unsupervised feature selection may be based on variability, signal strength measured by the range, correlation between features (only one feature is kept if two features are highly correlated), non-Gaussianity, etc.

The *singular value decomposition* (SVD) of an $n \times m$ matrix $\boldsymbol{X}$ is

$$\boldsymbol{X} \;=\; \boldsymbol{V}\,\boldsymbol{D}\,\boldsymbol{U}^T\,, \tag{3.3}$$

where $\boldsymbol{U}$ is an orthogonal $m \times m$ matrix, $\boldsymbol{V}$ an orthogonal $n \times n$ matrix, and $\boldsymbol{D}$ is a diagonal (diagonal for the first m rows) $n \times m$ matrix with positive entries, the *singular values*. The diagonal values of $\boldsymbol{D}$ are sorted decreasingly, so that the first value is the largest value (the largest singular value), the second value is the second largest value, etc. Computing $\boldsymbol{X}^T\boldsymbol{X}$, we see that $\boldsymbol{D}_m \;=\; \boldsymbol{D}^T\;\boldsymbol{D}$ (the eigenvalues are the singular values squared) and $\boldsymbol{U}$ is the same orthogonal matrix as in the eigendecomposition. SVD is often used to perform PCA.

For performing PCA, it is sufficient to know $\boldsymbol{U}$, because the projection of feature vector $\boldsymbol{x}$ onto the principal directions is given by $\boldsymbol{U}^T\boldsymbol{x}$. Therefore, the data $\boldsymbol{X}$ is projected onto $\boldsymbol{U}$, which gives $\boldsymbol{Y}$:

$$\boldsymbol{Y} \;=\; \boldsymbol{X}\,\boldsymbol{U} \;=\; \boldsymbol{V}\,\boldsymbol{D}\,. \tag{3.4}$$

We see that the SVD automatically provides the PCA projections via $\boldsymbol{V}\;\boldsymbol{D}$. For single observations $\boldsymbol{x}$ that is

$$\boldsymbol{y} \;=\; \boldsymbol{U}^T\boldsymbol{x}\,. \tag{3.5}$$

In principle, PCA is a matrix decomposition problem:

$$\boldsymbol{X} \;=\; \boldsymbol{Y}\boldsymbol{U}^T\,, \tag{3.6}$$

where $\boldsymbol{U}$ is orthogonal, $\boldsymbol{Y}^T\;\boldsymbol{Y} \;=\; \boldsymbol{D}_m$ (the $\boldsymbol{y}$ are orthogonal, that is they are decorrelated), and the eigenvalues of $\boldsymbol{D}_m$ are sorted decreasingly. For single observations that is

$$\boldsymbol{x} \;=\; \boldsymbol{U}\;\boldsymbol{y}\,. \tag{3.7}$$

The SVD allows an outer product representation of the matrix $\boldsymbol{X}$:

$$\boldsymbol{X} \;=\; \sum_{i=1}^{m} D_{ii}\;\boldsymbol{v}_i\boldsymbol{u}_i^T \;= \tag{3.8}$$
$$\sum_{i=1}^{m} \boldsymbol{y}_i\boldsymbol{u}_i^T\,,$$

where u_i is the i-th orthogonal column vector of $\boldsymbol{U}$, $\boldsymbol{v}_i$ is the i-th orthogonal column vector of $\boldsymbol{V}$, and $\boldsymbol{y}_i \;=\; D_{ii}\;\boldsymbol{v}_i$.

Iterative methods for PCA are sometimes to prefer if the dimension m is large or if on-line methods should be implemented. Most famous is Oja's rule Oja [1982]. If the current projection is

$$t \;=\; \boldsymbol{u}^T\boldsymbol{x} \tag{3.9}$$

then Oja's rule is

$$\boldsymbol{u}^{\text{new}} \;=\; \boldsymbol{u} \;+\; \eta \,\left(t\,\boldsymbol{x} \;-\; t^2\,\boldsymbol{u}\right)\;, \tag{3.10}$$

where η is the learning rate.

The eigenvectors of $\boldsymbol{C}$ are the fixed points of Oja's rule and only the eigenvector with largest eigenvalue is a stable fixed point.

$$\begin{aligned} \mathrm{E}_{\boldsymbol{x}}(\boldsymbol{u}^{\text{new}}) \;&=\; \boldsymbol{u} \;+\; \eta\,\mathrm{E}_{\boldsymbol{x}}\left(\boldsymbol{x}(\boldsymbol{x}^T\boldsymbol{u}) \;-\; (\boldsymbol{u}^T\boldsymbol{x})(\boldsymbol{x}^T\boldsymbol{u})\,\boldsymbol{u}\right) \\ &=\; \boldsymbol{u} \;+\; \eta\,\left(\mathrm{E}_{\boldsymbol{x}}(\boldsymbol{x}\boldsymbol{x}^T)\boldsymbol{u} \;-\; \left(\boldsymbol{u}^T E_{\boldsymbol{x}}(\boldsymbol{x}\boldsymbol{x}^T)\boldsymbol{u}\right)\,\boldsymbol{u}\right) \\ &=\; \boldsymbol{u} \;+\; \eta\,\left(\boldsymbol{C}\boldsymbol{u} \;-\; (\boldsymbol{u}^T\boldsymbol{C}\boldsymbol{u})\,\boldsymbol{u}\right)\;. \end{aligned} \tag{3.11}$$

If $\boldsymbol{u}$ is and eigenvector of $\boldsymbol{C}$ with eigenvalue λ then

$$\mathrm{E}_{\boldsymbol{x}}(\boldsymbol{u}^{\text{new}}) \;=\;\; \boldsymbol{u} \;+\; \eta\,(\lambda\boldsymbol{u} \;-\; \lambda\,\boldsymbol{u}) \;=\; \boldsymbol{u}\,. \tag{3.12}$$

Therefore each eigenvector of $\boldsymbol{C}$ is a fixed point of Oja's rule.

3.2 Variance Maximization

The first principal component $\boldsymbol{u}_1$ is the direction of the maximum possible data variance:

$$\boldsymbol{u}_1 \;=\; \underset{\|\boldsymbol{u}\|=1}{\arg\max} \sum_{i=1}^{n} \left(\boldsymbol{u}^T\,\boldsymbol{x}_i\right)^2\;. \tag{3.13}$$

This can easily be seen because

$$\begin{aligned} \sum_{i=1}^{n} \left(\boldsymbol{u}^T\boldsymbol{x}_i\right)^2 \;&=\; \sum_{i=1}^{n} \left(\boldsymbol{u}^T\boldsymbol{x}_i\right)\left(\boldsymbol{x}_i^T\boldsymbol{u}\right) \;= \\ &\boldsymbol{u}^T \sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{u} \;=\; n\,\boldsymbol{u}^T\boldsymbol{C}\boldsymbol{u}\,. \end{aligned} \tag{3.14}$$

With $\boldsymbol{C} = \sum_{i=1}^{m} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T$, $\boldsymbol{u} = \sum_{i=1}^{m} a_i \boldsymbol{u}_i$, and $\sum_{i=1}^{m} a_i^2 = 1$ we have

$$\boldsymbol{u}^T\boldsymbol{C}\boldsymbol{u} \;=\; \sum_{i=1}^{m} \lambda_i a_i^2 \tag{3.15}$$

and $\sum_{i=1}^{m} a_i^2 = 1$. The value $\sum_{i=1}^{m} \lambda_i a_i^2$ is maximal for $a_1 = 1$ and all other $a_i = 0$, because all $\lambda_i > 0$ and λ_1 is the largest eigenvalue.

Furthermore, principal components correspond to the direction of the maximum possible variance orthogonal to all previous components. If we remove the subspace of all previous components $1, \ldots, k$:

$$\boldsymbol{x}_i^k \;=\; \boldsymbol{x}_i \;-\; \sum_{t=1}^{k-1} \left(\boldsymbol{u}_t^T\,\boldsymbol{x}_i\right)\,\boldsymbol{u}_t \tag{3.16}$$

then the k-th principal component is the direction of the maximum data variance:

$$\boldsymbol{u}_k \;=\; \underset{\|\boldsymbol{u}\|=1}{\arg\max} \sum_{i=1}^{n} \left(\boldsymbol{u}^T\, \boldsymbol{x}_i^k\right)^2 \;. \tag{3.17}$$

This can inductively been proved analog to the first principal component. Since the components are sorted, the first l components span the l-dimensional subspace with maximal data variance.

3.3 Uniqueness

Is PCA unique or not, that is, is there only one PCA solution. Multiple solutions may fulfill the PCA criteria. We consider the decomposition

$$\boldsymbol{X} \;=\; \boldsymbol{Y}\boldsymbol{U}^T \,, \tag{3.18}$$

where $\boldsymbol{U}$ is orthogonal, $\boldsymbol{Y}^T\,\boldsymbol{Y} \;=\; \boldsymbol{D}_m$ with $\boldsymbol{D}_m$ as m-dimensional diagonal matrix, and the eigenvalues of $\boldsymbol{D}_m$ are sorted decreasingly.

PCA is unique up to signs, if the eigenvalues of the covariance matrix are different from each other.

Begin proof
To prove this statement, assume another representation

$$\boldsymbol{X} \;=\; \boldsymbol{Y}'\boldsymbol{U}'^T \,, \tag{3.19}$$

where $\boldsymbol{U}'$ is orthogonal, $(\boldsymbol{Y}')^T\boldsymbol{Y}' \;=\; \boldsymbol{D}'_m$ with $\boldsymbol{D}'_m$ as m-dimensional diagonal matrix, and the eigenvalues of $\boldsymbol{D}'_m$ are sorted decreasingly.

If eigenvalues of $\boldsymbol{D}_m$ are different from each other, then at most one eigenvalue can be zero. If one eigenvalue of $\boldsymbol{D}_m$ is zero, the observations do not have any variance in the direction of the according eigenvector. This direction is unique, becomes principal component $\boldsymbol{u}_m$, and can be removed from the data. Subsequent, we can perform PCA on the remaining $(m-1)$-dimensional space, where all eigenvalues are larger than zero.

We assume that all eigenvalues of $\boldsymbol{D}_m$ (therefore also of $\boldsymbol{Y}$) are larger than zero. Therefore a matrix $\boldsymbol{A} = \boldsymbol{Y}^{-1}\boldsymbol{Y}'$ exists with

$$\boldsymbol{Y}' \;=\; \boldsymbol{Y}\,\boldsymbol{A}\,. \tag{3.20}$$

We obtain

$$\boldsymbol{X} \;=\; \boldsymbol{Y}\,\boldsymbol{U}^T \;=\; \boldsymbol{Y}'\,\boldsymbol{U}'^T \;=\; \boldsymbol{Y}\,\boldsymbol{A}\,\boldsymbol{U}'^T \tag{3.21}$$

after multiplying with $\boldsymbol{Y}^{-1}$ from the left we get

$$\boldsymbol{U}^T \;=\; \boldsymbol{A}\,\boldsymbol{U}'^T\,. \tag{3.22}$$

Since $\boldsymbol{U}$ and $\boldsymbol{U}'$ are orthogonal, we obtain

$$\boldsymbol{I} \;=\; \boldsymbol{U}^T\boldsymbol{U} \;=\; \boldsymbol{A}\boldsymbol{U}'\boldsymbol{U}'^T\boldsymbol{A}^T \;=\; \boldsymbol{A}\boldsymbol{A}^T\,. \tag{3.23}$$

Therefore A is an orthogonal matrix and

$$\boldsymbol{U}' \;=\; \boldsymbol{U}\,\boldsymbol{A}\,. \tag{3.24}$$

We obtain

$$\boldsymbol{D}'_m \;=\; (\boldsymbol{Y}')^T\,\boldsymbol{Y}' \;=\; \boldsymbol{A}^T\,\boldsymbol{Y}^T\,\boldsymbol{Y}\,\boldsymbol{A} \;=\; \boldsymbol{A}^T\,\boldsymbol{D}_m\boldsymbol{A}\,. \tag{3.25}$$

Thus, the eigenvalues of $\boldsymbol{D}'_m$ match the diagonal elements of $\boldsymbol{D}_m$. Further the i-th eigenvalue λ_i of the covariance matrix $\boldsymbol{C}$ is $\lambda_i = D_{ii}$. According to our assumption, the eigenvalues are sorted in both $\boldsymbol{D}'_m$ and $\boldsymbol{D}_m$. The sorting is unique, because we assumed mutually different eigenvalues. Therefore we have

$$\boldsymbol{D}'_m \;=\; \boldsymbol{D}_m\,. \tag{3.26}$$

It follows that

$$\boldsymbol{A}\,\boldsymbol{D}_m \;=\; \boldsymbol{D}_m\,\boldsymbol{A} \tag{3.27}$$

and

$$[\boldsymbol{A}\,\boldsymbol{D}_m]_{ij} \;=\; a_{ij}\,D_{jj} \;=\; a_{ij}\,d_{ii} \;=\; [\boldsymbol{D}_m\,\boldsymbol{A}]_{ij} \tag{3.28}$$

which gives

$$a_{ij}\,(D_{jj} \;-\; D_{ii}) \;=\; 0\,. \tag{3.29}$$

For $i \neq j$ our assumption is that $D_{jj} \;\neq\; D_{ii}$ ($\lambda_j = D_{jj}$), therefore we deduce $a_{ij} \;=\; 0$. Hence, $\boldsymbol{A}$ is diagonal and orthogonal. Consequently, $\boldsymbol{A}$ is diagonal and contains only ones and minus ones on its diagonal. Thus, PCA is unique up to signs, if the eigenvalues are mutually different.
End proof

3.4 Properties of PCA

- The projection of the data on the first principal component (PC) has maximal variance of all possible one-dimensional projections. That means $\boldsymbol{u}_1$ maximizes

$$\boldsymbol{u}^T\,\boldsymbol{C}\,\boldsymbol{u} \;\text{ s.t. }\; \|\boldsymbol{u}\| = 1\,. \tag{3.30}$$

  The first l PCs maximize

$$\sum_{i=1}^{l} \boldsymbol{u}_i^T\,\boldsymbol{C}\,\boldsymbol{u}_i \;\text{ s.t. }\; \boldsymbol{u}_i^T\,\boldsymbol{u}_j = \delta_{ij}\,. \tag{3.31}$$

- The projections onto PCs have zero means:

$$\frac{1}{n}\,\sum_{i=1}^{n} \boldsymbol{u}_k^T\,\boldsymbol{x}_i \;=\; \boldsymbol{u}_k^T\,\left(\frac{1}{n}\,\sum_{i=1}^{n} \boldsymbol{x}_i\right) \;=\; \boldsymbol{u}_k^T\boldsymbol{0} \;=\; 0\,. \tag{3.32}$$

- The projections onto PCs are mutually uncorrelated (second moment), that is, they are orthogonal to each other. We already expressed this by $\boldsymbol{Y}^T\boldsymbol{Y} = \boldsymbol{I}$ but it can also be seen at

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{u}_t^T\,\boldsymbol{x}_i)\,(\boldsymbol{u}_s^T\,\boldsymbol{x}_i) &= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{u}_t^T\,\boldsymbol{x}_i)\,(\boldsymbol{x}_i^T\,\boldsymbol{u}_s) \\ &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{u}_t^T\,(\boldsymbol{x}_i\,\boldsymbol{x}_i^T)\,\boldsymbol{u}_s \\ &= \boldsymbol{u}_t^T\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\,\boldsymbol{x}_i^T\right)\boldsymbol{u}_s \\ &= \boldsymbol{u}_t^T\,\boldsymbol{C}\,\boldsymbol{u}_s = \lambda_s\,\boldsymbol{u}_t^T\,\boldsymbol{u}_s = 0\,.\end{aligned} \tag{3.33}$$

  For the last equation we used $\boldsymbol{C} = \sum_{j=1}^{m}\lambda_j\boldsymbol{u}_j\boldsymbol{u}_j^T$. Therefore correlation coefficients between projections of the observations onto PCs are zero.

- The sample variance of the k-th projection is equal to the k-th eigenvalue of the sample covariance matrix $\boldsymbol{C}$

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{u}_k^T\boldsymbol{x}_i\right)^2 &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{u}_k^T\left(\boldsymbol{x}_i\boldsymbol{x}_i^T\right)\boldsymbol{u}_k \\ &= \boldsymbol{u}_k^T\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\,\boldsymbol{x}_i^T\right)\boldsymbol{u}_k \\ &= \boldsymbol{u}_k^T\,\boldsymbol{C}\,\boldsymbol{u}_k = \lambda_k\,\boldsymbol{u}_k^T\,\boldsymbol{u}_k = \lambda_k\,.\end{aligned} \tag{3.34}$$

  where λ_k is the k-th eigenvalue of the covariance matrix $\boldsymbol{C}$.

- PCs are ranked decreasingly according to their eigenvalues which are the variances in the PC directions.

- The first l PCs minimize the mean-squared error.

  The representation of $\boldsymbol{x}$ by $\hat{\boldsymbol{x}}$ with the first l PCs is

$$\hat{\boldsymbol{x}} = \sum_{k=1}^{l}\boldsymbol{u}_k\,\boldsymbol{u}_k^T\boldsymbol{x}\,, \tag{3.35}$$

  where

$$\boldsymbol{C} = \sum_{k=1}^{m}\lambda_k\,\boldsymbol{u}_k\,\boldsymbol{u}_k^T\,. \tag{3.36}$$

For the approximation of $\boldsymbol{x}$ by $\hat{\boldsymbol{x}}$, the mean-squared error is

$$
\begin{aligned}
\mathrm{E}\left(\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2\right) &= \mathrm{E}\left(\boldsymbol{x}^T\boldsymbol{x} - 2\,\boldsymbol{x}^T\,\hat{\boldsymbol{x}} + \hat{\boldsymbol{x}}^T\hat{\boldsymbol{x}}\right) \qquad (3.37)\\
&= \mathrm{E}\left(\mathrm{Tr}\left(\boldsymbol{x}\boldsymbol{x}^T\right) - 2\,\mathrm{Tr}\left(\sum_{k=1}^{l}\boldsymbol{u}_k\,\boldsymbol{u}_k^T\boldsymbol{x}\boldsymbol{x}^T\right) + \mathrm{Tr}\left(\sum_{k=1}^{l}\boldsymbol{u}_k\,\boldsymbol{u}_k^T\boldsymbol{x}\boldsymbol{x}^T\right)\right)\\
&= \mathrm{Tr}\left(\mathrm{E}\left(\boldsymbol{x}\boldsymbol{x}^T\right) - 2\sum_{k=1}^{l}\boldsymbol{u}_k\,\boldsymbol{u}_k^T\mathrm{E}\left(\boldsymbol{x}\boldsymbol{x}^T\right) + \sum_{k=1}^{l}\boldsymbol{u}_k\,\boldsymbol{u}_k^T\mathrm{E}\left(\boldsymbol{x}\boldsymbol{x}^T\right)\right)\\
&= \mathrm{Tr}\left(\boldsymbol{C} - \sum_{k=1}^{l}\boldsymbol{u}_k\,\boldsymbol{u}_k^T\boldsymbol{C}\right)\\
&= \mathrm{Tr}\left(\boldsymbol{C} - \sum_{k=1}^{l}\boldsymbol{u}_k\,\boldsymbol{u}_k^T\sum_{k=1}^{m}\lambda_k\boldsymbol{u}_k\,\boldsymbol{u}_k^T\right)\\
&= \mathrm{Tr}\left(\sum_{k=1}^{m}\lambda_k\,\boldsymbol{u}_k\,\boldsymbol{u}_k^T - \sum_{k=1}^{l}\lambda_k\boldsymbol{u}_k\,\boldsymbol{u}_k^T\right)\\
&= \mathrm{Tr}\left(\sum_{k=l+1}^{m}\lambda_k\,\boldsymbol{u}_k\,\boldsymbol{u}_k^T\right)\\
&= \sum_{k=l+1}^{m}\lambda_k\,\mathrm{Tr}\left(\boldsymbol{u}_k\,\boldsymbol{u}_k^T\right)\\
&= \sum_{k=l+1}^{m}\lambda_k\,\mathrm{Tr}\left(\boldsymbol{u}_k^T\,\boldsymbol{u}_k\right)\\
&= \sum_{k=l+1}^{m}\lambda_k\,.
\end{aligned}
$$

where λ_k is the square root of the k-th eigenvalue of $\boldsymbol{C}$ or the k-th singular value of $\boldsymbol{X}$.

Each representation of the data by projections to other l vectors $(\boldsymbol{u}')_k$ will have a larger mean squared error. Using the transformations of the last equation, we obtain for the mean squared error

$$
\mathrm{Tr}\left(\boldsymbol{C} - \sum_{k=1}^{l}(\boldsymbol{u}')_k(\boldsymbol{u}')_k^T\boldsymbol{C}\right)\,. \qquad (3.38)
$$

If $(\boldsymbol{u}')_k = \sum_{i=1}^{m} b_{ki}\boldsymbol{u}_i$ with $\sum_{i=1}^{m} b_{ki}^2 = 1$.

The mean squared error for projection onto the l vectors $(\boldsymbol{u}')_k$ is

$$\begin{aligned} \mathrm{Tr}\left(\boldsymbol{C} - \sum_{k=1}^{l}(\boldsymbol{u}')_k(\boldsymbol{u}')_k^T\boldsymbol{C}\right) &= \sum_{i=1}^{m}\lambda_i - \sum_{k=1}^{l}\sum_{i=1}^{m} b_{ki}^2\lambda_i \\ &= \sum_{i=1}^{m}\lambda_i\left(1 - \sum_{k=1}^{l} b_{ki}^2\right). \end{aligned} \tag{3.39}$$

The Hessian matrix of this objective with respect to the parameters b_{ki} has negative eigenvalues, therefore this is a strict concave function. The maximum principle states that the minimum of this objective is found on the boundary. That means $b_{ki} = 0$ or $b_{ki} = 1$. Therefore the $(\boldsymbol{u}')_k$ are a permutation of $\boldsymbol{u}_k$. $\sum_{k=l+1}^{m}\lambda'_k \geq \sum_{k=l+1}^{m}\lambda_k$ where equality is only achieved if the λ'_k is a permutation of λ_k for $l+1 \leq k \leq m$. Therefore the first l vectors $(\boldsymbol{u}')_k$ are a permutation of the first l $\boldsymbol{u}_k$. If we assume that the eigenvectors are sorted according to the eigenvalues, then $(\boldsymbol{u}')_k = Bu_k$ for $1 \leq k \leq l$. Thus, a projection onto other l vectors than the principal components leads to a larger mean squared error than those of PCA.

3.5 Examples

3.5.1 Iris Data Set

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Fisher (1936) as an example of discriminant analysis Fisher [1936]. Iris is a genus of 260–300 species of flowering plants with showy flowers (see Fig. 3.3). The name stems from the Greek word for a rainbow, as the flower colors have a broad variety. The three species of the data set are Iris setosa (Beachhead Iris), Iris versicolor (Larger Blue Flag, Harlequin Blueflag), and Iris virginica (Virginia Iris). This data set is sometimes called Anderson's Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species Anderson [1935]. Two of the three species were collected in the Gaspe Peninsula "all from the same pasture, and picked on the same day and measured at the same time by the same person with the same apparatus" Anderson [1935].

Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters(see Fig. 3.4). For each of the three species 50 flowers were measured (see part of the data in Tab. 3.1). Based on these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

We perform PCA on this iris data set:

```
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
```

Figure 3.3: Examples of Iris flowers.

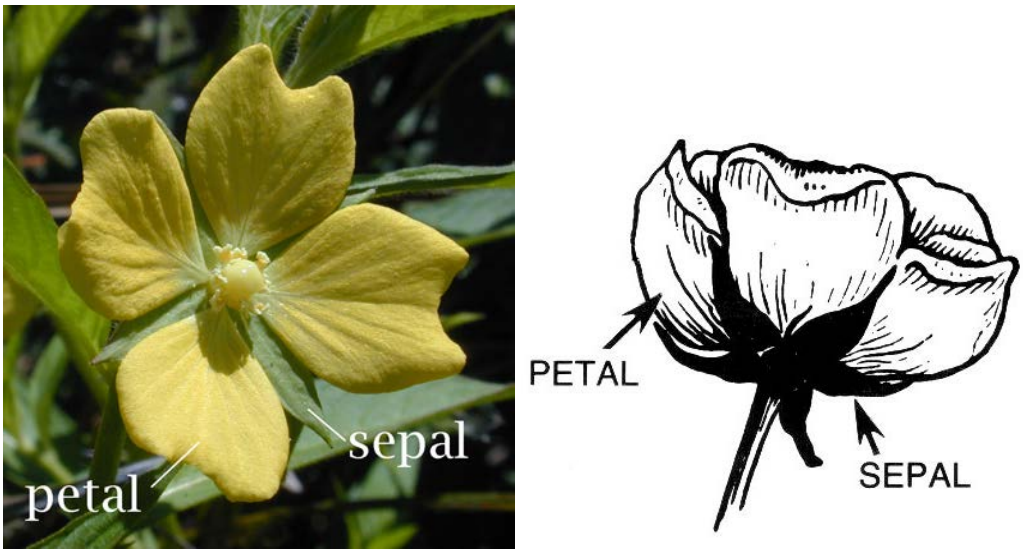


Figure 3.4: Flowerparts petal and septal are depicted and marked.

Table 3.1: Part of the iris data set with features sepal length, sepal width, petal length, and petal width.

No.	Sepal		Petal		Species
	Length	Width	Length	Width	
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
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.5	1.5	versicolor
53	6.9	3.1	4.9	1.5	versicolor
54	5.5	2.3	4.0	1.3	versicolor
55	6.5	2.8	4.6	1.5	versicolor
101	6.3	3.3	6.0	2.5	virginica
102	5.8	2.7	5.1	1.9	virginica
103	7.1	3.0	5.9	2.1	virginica
104	6.3	2.9	5.6	1.8	virginica
105	6.5	3.0	5.8	2.2	virginica

We see that the first principal component explains 92% of the variance in the data. This means that the features are correlated and the variance driving this correlation is captured by principal component 1. Probably PC1 expresses the size of the blossom which is reflected in all four features.

Fig. 3.5 shows scatter plots for pairs of principal components, more precisely, scatter plots of the projection of the observations to pairs of PCs. Only PC1 helps to separate the species.

3.5.2 Multiple Tissue Data Set

This data set consists of microarray data from the Broad Institute "Cancer Program Data Sets" which was produced by Su et al. [2002]. The data contains gene expression profiles from human and mouse samples across a diverse set of tissues, organs, and cell lines. The goal was to have a reference for the normal mammalian transcriptome. The microarray platforms were Affymetrix human (U95A) or mouse (U74A) high-density oligonucleotide arrays. The authors profiled the gene expression level from 102 human and mouse samples and selected 5,565 genes. Gene selection is an important first step when analyzing microarray data Hochreiter and Obermayer [2004, 2005], Talloen et al. [2007], Kasim et al. [2010], Talloen et al. [2010].

The samples predominantly come from a normal physiological state in the human and the mouse. The data set represents a preliminary, but substantial, description of the normal mammalian transcriptome. Mining these data may reveal insights into molecular and physiological gene function, mechanisms of transcriptional regulation, disease etiology, and comparative genomics. Hoshida et al. [2007] used this data set to identify subgroups in the samples by using

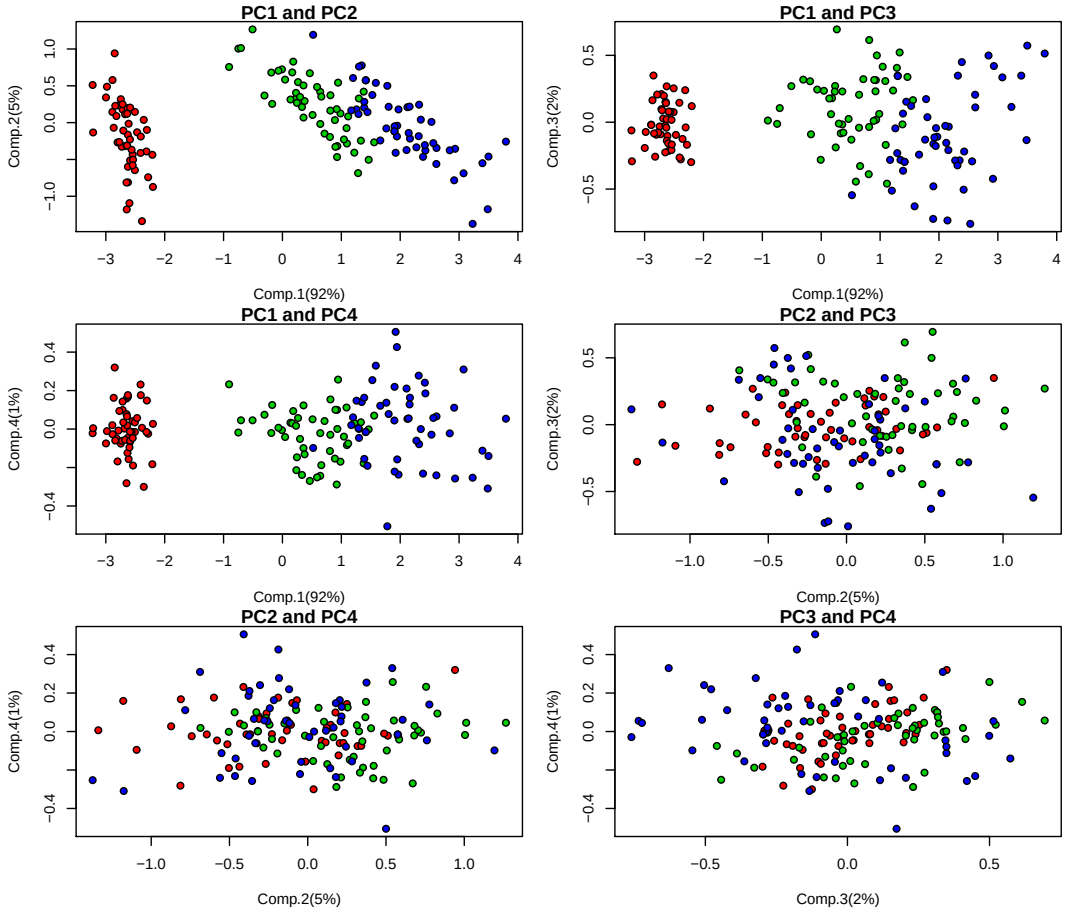


Figure 3.5: PCA applied to Anderson's iris data. The matrix shows scatter plots for pairs of principal components.

additional data of the same kind. The four distinct tissue types are:

- breast (Br),
- prostate (Pr),
- lung (Lu),
- and colon (Co).

These tissue types are indicated in the data set.

We apply PCA to the multiple tissue microarray data set. Gene expression values for different tissue types for human and mouse are measured. The data set contains 102 samples for each of which expression values of 5,565 genes are available. Four distinct tissue types are indicated in the data: breast (Br), prostate (Pr), lung (Lu), and colon (Co). We want to see if PCA allows to identify these tissue types.

Fig. 3.6 shows scatter plots for pairs of principal components, i.e. the projections of the observations to pairs of PCs. PC1 separates the prostate samples (green) from the rest. PC2 separates the colon samples (orange) but also breast samples (red). PC3 separates some lung samples (blue).

Next we perform variance filtering before PCA. For microarray data, variance filtering is justified because genes that are differentially expressed across the samples have higher variance. For such genes the noise variance and the variance due to the signal add up. Therefore, genes with largest variance are assumed to contain a signal and to have higher signal-to-noise ratio. The following filtered data sets are considered:

```
XMultiF1: 101 genes with the highest variance
XMultiF2:  13 genes with the highest variance
XMultiF3:   5 genes with the highest variance
```

For the 101 genes with the highest variance, Fig. 3.7 shows scatter plots for pairs of principal components, i.e. the projections of the observations to pairs of PCs. Principal component 1 separates the prostate samples (green) from the rest. PC2 separates the colon samples (orange) from the rest. PC3 separates the breast samples (red) from the rest and at the same time lung samples (blue) from the rest. PCA on the filtered data set separates the tissues better than PCA on the whole data set.

PCA on the multiple tissue data with 13 genes that have largest variance is shown in Fig. 3.8. PC1 separates the prostate samples (green) from the rest. PC2 separates the colon samples (orange) from the rest. PC3 separates the breast samples (red) from the rest at one side but at the other side time lung samples (blue).

PCA on the multiple tissue data with 5 genes that have largest variance is shown in Fig. 3.9. Still PC1 separates the prostate samples (green) from the rest. However other tissues are difficult to separate. Four out of the 5 genes are highly correlated and give the same signal. Probably this signal is indicative for the prostate tissue.

```
              ACPP        KLK2         KRT5        MSMB       TRGC2
ACPP   1.000000000  0.97567890 -0.004106762  0.90707887 0.947433227
```

```
KLK2   0.975678903  1.00000000 -0.029900946  0.89265825 0.951841913
KRT5  -0.004106762 -0.02990095  1.000000000 -0.05565599 0.008877815
MSMB   0.907078869  0.89265825 -0.055655985  1.00000000 0.870922667
TRGC2  0.947433227  0.95184191  0.008877815  0.87092267 1.000000000
```

In the GeneCards database `http://www.genecards.org` we find:

> ACPP "is synthesized under androgen regulation and is secreted by the epithelial cells of the prostate gland."

further we find

> KLK2 "is primarily expressed in prostatic tissue and is responsible for cleaving pro-prostate-specific antigen into its enzymatically active form."

and

> MSMB "is synthesized by the epithelial cells of the prostate gland and secreted into the seminal plasma."

We now select genes which are not so closely correlated to each other. Toward this end we first cluster (see later in the course) the genes and then select one prototype from each cluster. These 10 genes are not as closely related as the genes which are selected based on variance alone:

```
               ABP1       ACPP      AKR1C1     ALDH1A3      ANXA8          APOD
ABP1     1.00000000 -0.1947766 -0.04224634 -0.21577195 -0.2618053 -0.3791812658
ACPP    -0.19477662  1.0000000 -0.22929893  0.88190657 -0.2978638  0.4964638048
AKR1C1  -0.04224634 -0.2292989  1.00000000 -0.07536066  0.4697886 -0.1793466620
ALDH1A3 -0.21577195  0.8819066 -0.07536066  1.00000000 -0.1727669  0.4113925823
ANXA8   -0.26180526 -0.2978638  0.46978864 -0.17276688  1.0000000 -0.1863923785
APOD    -0.37918127  0.4964638 -0.17934666  0.41139258 -0.1863924  1.0000000000
BST2    -0.02752210 -0.1858633  0.03341592 -0.18706898  0.1672327  0.0001475666
CA12    -0.03390577 -0.5266892  0.20825388 -0.55430511  0.1535930 -0.0861446268
CLDN3    0.33206818  0.3547601 -0.52997065  0.24516720 -0.6819272  0.2272871855
IGHA1   -0.14341643 -0.2835074  0.45479347 -0.08918854  0.2726503 -0.1157383141
                BST2        CA12       CLDN3       IGHA1
ABP1    -0.0275221025 -0.03390577  0.3320682 -0.14341643
ACPP    -0.1858633000 -0.52668918  0.3547601 -0.28350737
AKR1C1   0.0334159199  0.20825388 -0.5299707  0.45479347
ALDH1A3 -0.1870689799 -0.55430511  0.2451672 -0.08918854
ANXA8    0.1672327418  0.15359297 -0.6819272  0.27265032
APOD     0.0001475666 -0.08614463  0.2272872 -0.11573831
BST2     1.0000000000  0.08971880 -0.1918497  0.16460367
CA12     0.0897187966  1.00000000 -0.3170681  0.17639489
CLDN3   -0.1918497331 -0.31706813  1.0000000 -0.39690211
IGHA1    0.1646036701  0.17639489 -0.3969021  1.00000000
```

Fig. 3.10 shows the PCA result. The tissues are not as well separated as with maximizing the variance of the genes because some highly variable genes are missed. Tissues can be separated but not with the same quality as with more genes.

Next we did feature selection based on hierarchical clustering and variance maximization within one cluster. For each cluster the gene with maximal variance is selected. Fig. 3.11 shows the PCA result for feature selection based on hierarchical clustering, which gave 92 genes. Results are very similar to variance based feature selection. However one improvement is visible. PC3 separates breast samples (red) from lung samples (blue) which was not achieved by the other projections.

Next we do feature selection based on hierarchical clustering but now the distance between genes is based on their correlation. Fig. 3.12 shows the PCA result for feature selection based on hierarchical clustering based on the correlation coefficient matrix. For each of the 95 clusters the gene with maximal variance was selected. Again, the results are very similar to variance based feature selection. PC3 separates breast samples (red) from lung samples (blue) almost as good as in previous example.

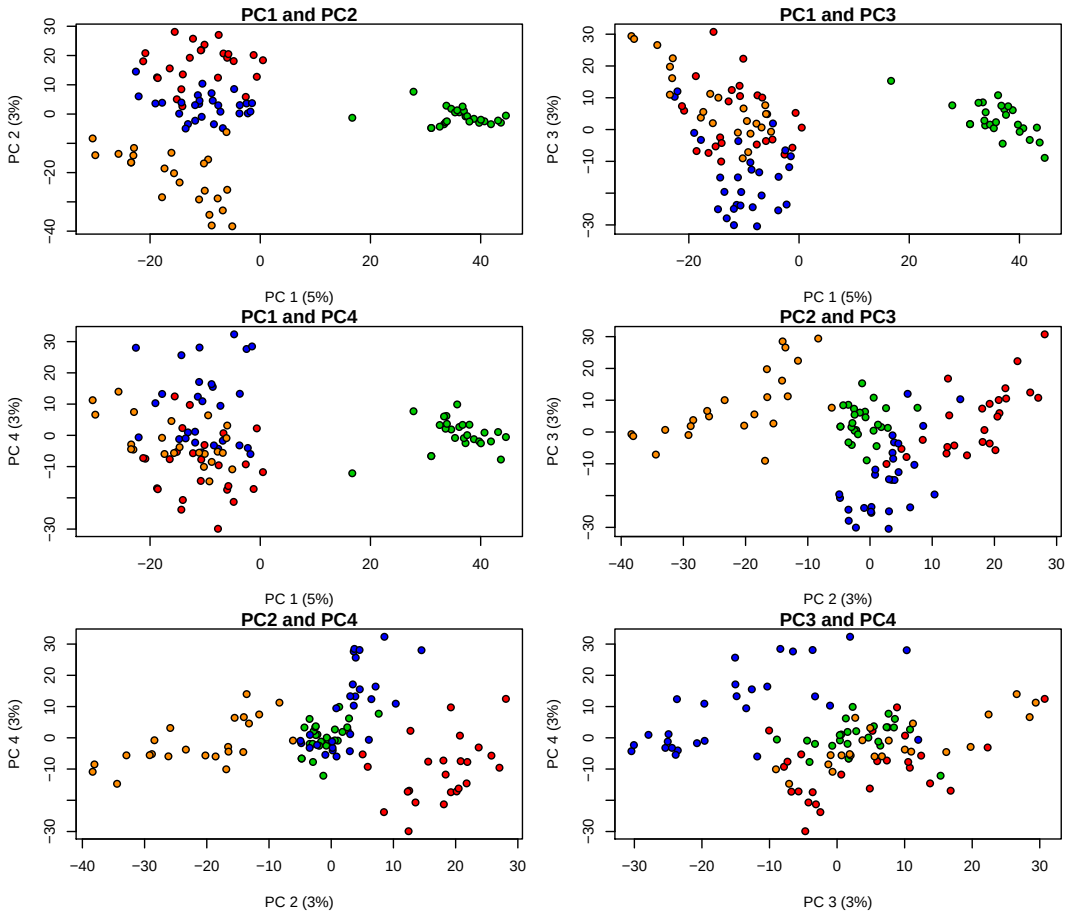


Figure 3.6: PCA applied to multiple tissue data. PC1 separates the prostate samples (green) from the rest. PC2 separates the colon samples (orange) from the rest. PC3 separates some lung samples (blue).

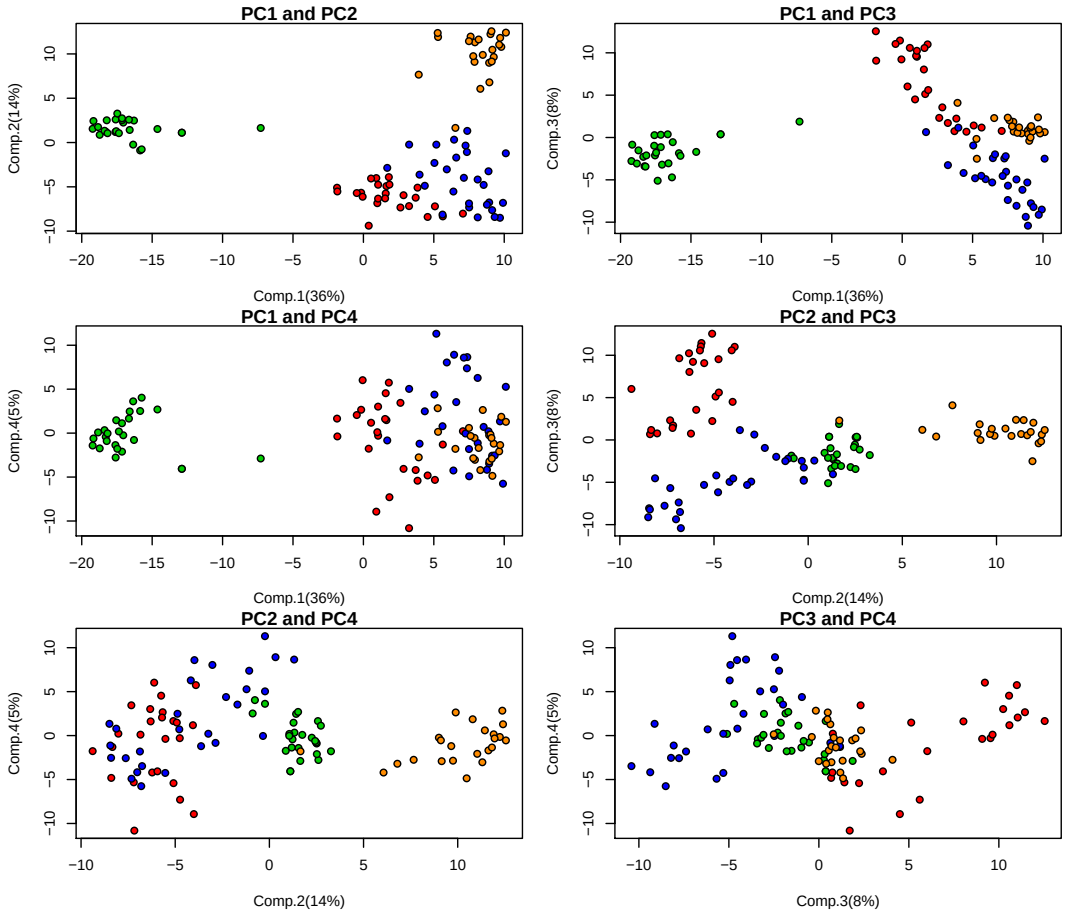


Figure 3.7: PCA applied to multiple tissue data with 101 most variable genes. PC1 separates the prostate samples (green) from the rest. PC2 separates the colon samples (orange) from the rest. To the left, PC3 separates the breast samples (red) from the rest but, to the right, it also separates lung samples (blue).

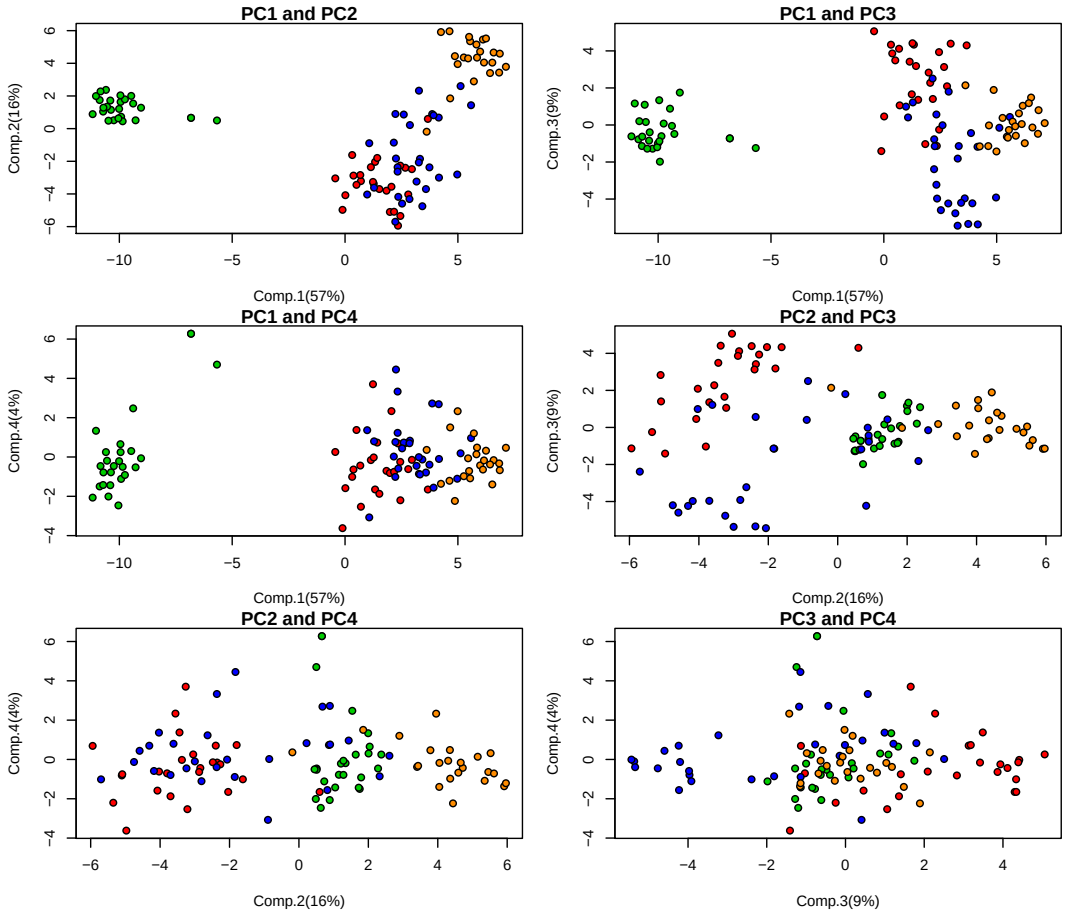


Figure 3.8: PCA applied to multiple tissue data with 13 most variable genes. Again PC1 separates the prostate samples (green) from the rest. However, the separation of colon samples (orange) by PC2 is worse than with 101 genes. Also the separation of the breast samples (red) and lung samples (blue) by PC3 is worse that with 101 genes.

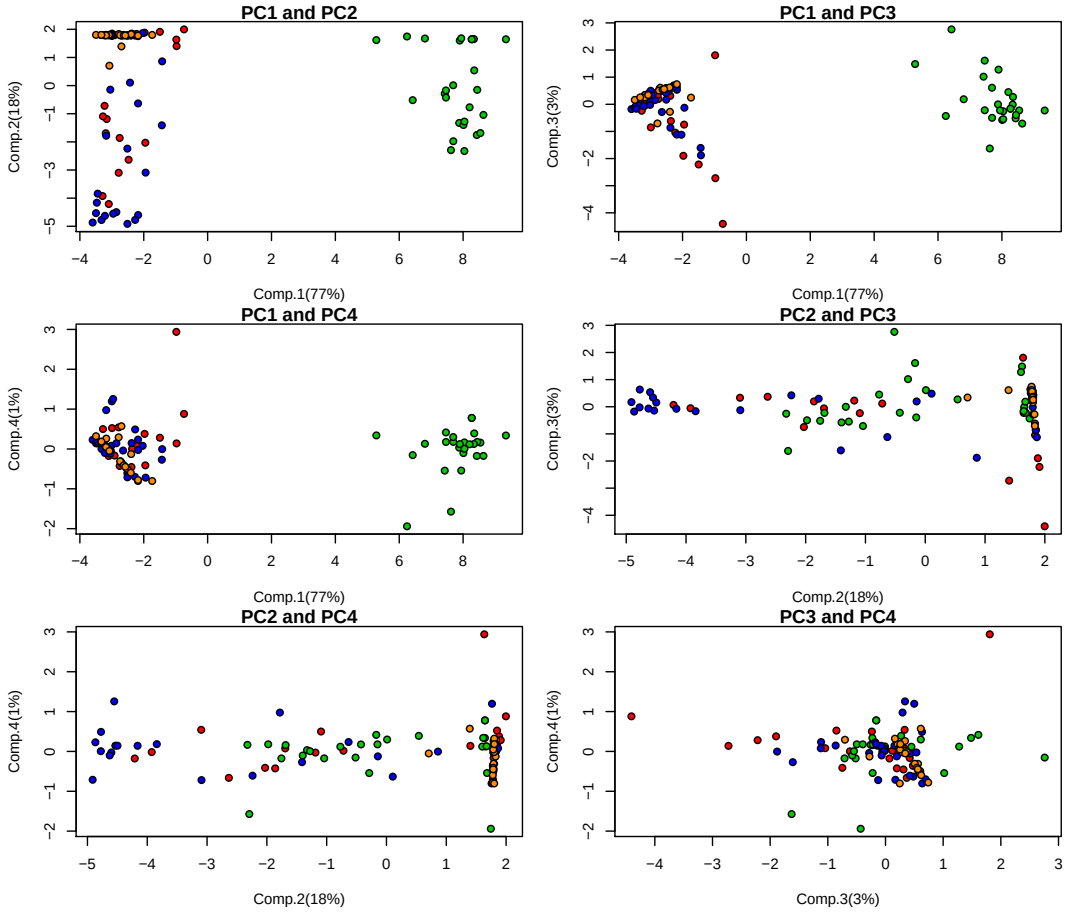


Figure 3.9: PCA applied to multiple tissue data with 5 most variable genes. Still PC1 separates the prostate samples (green) from the rest. However other tissues were not separated.

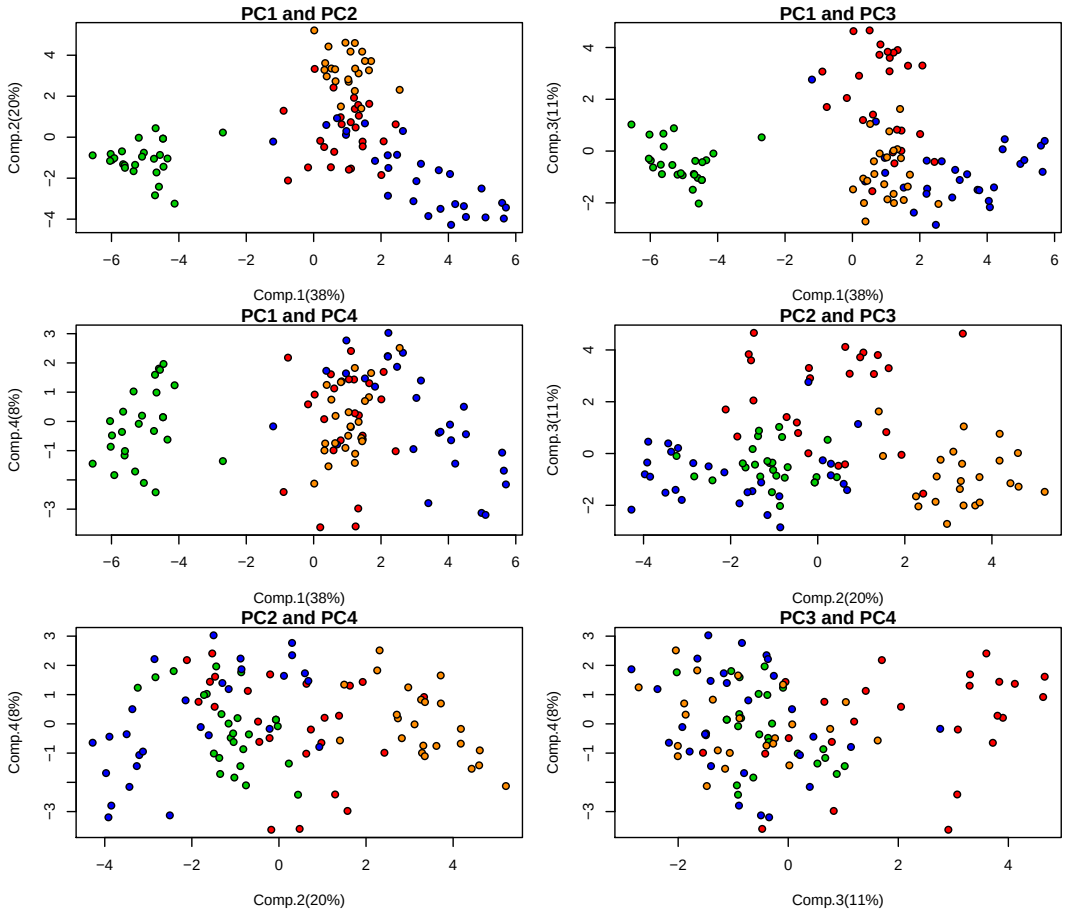


Figure 3.10: PCA applied to multiple tissue data with 10 genes which are not too closely correlated.

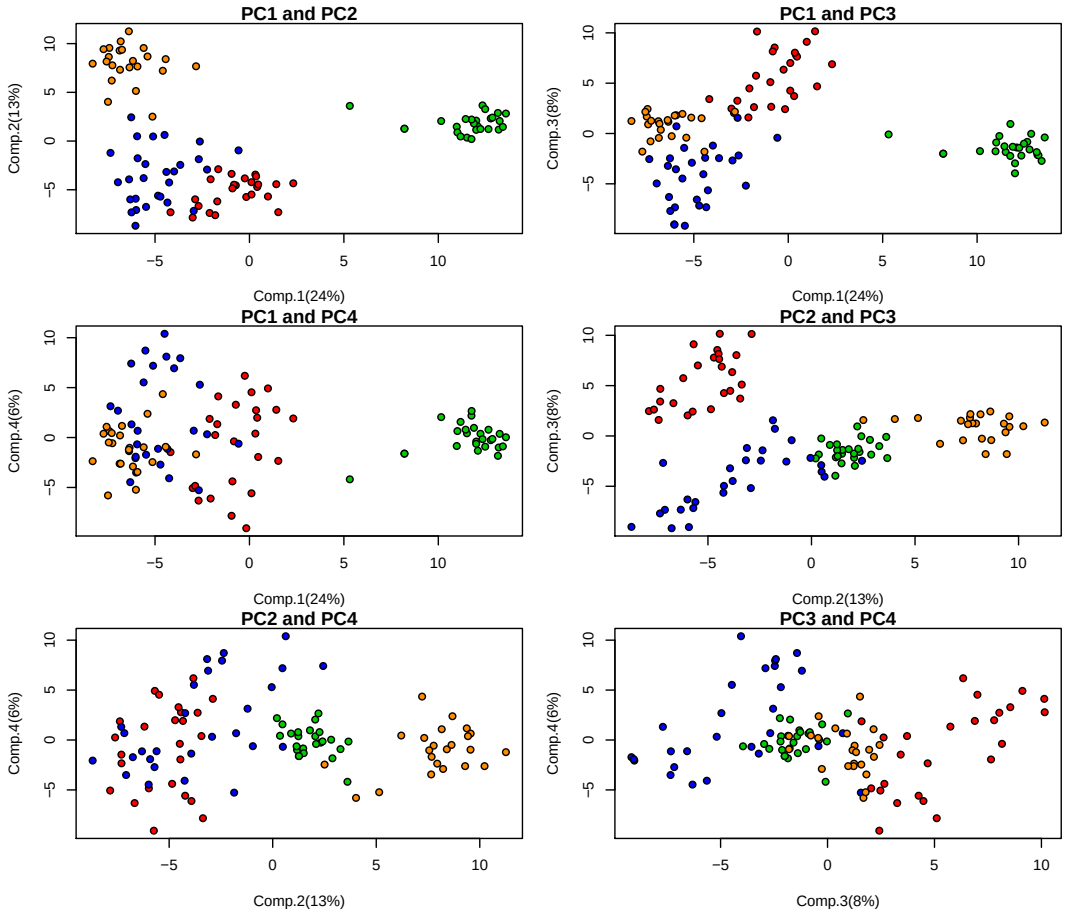


Figure 3.11: PCA applied to multiple tissue data with 92 genes selected by hierarchical clustering. PC3 separates breast samples (red) from lung samples (blue) which was not achieved by the other projections.

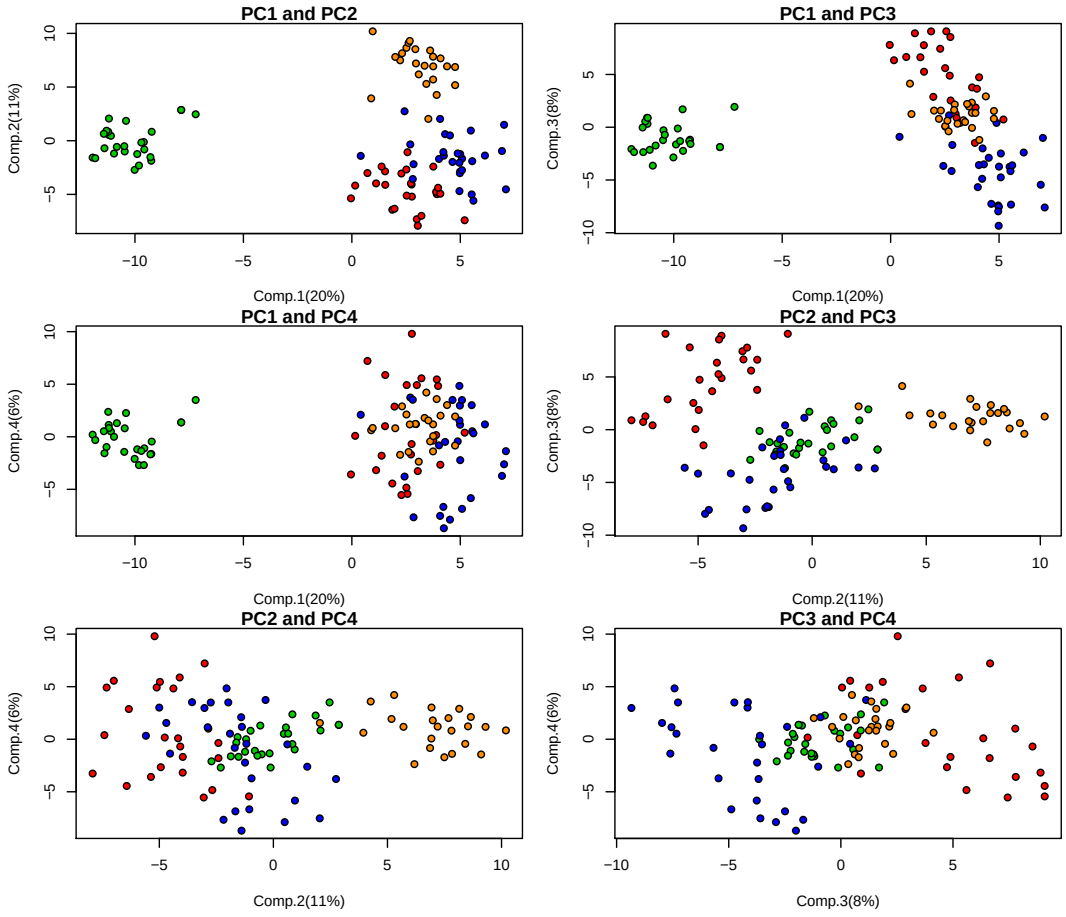


Figure 3.12: PCA applied to multiple tissue data with 95 genes selected by hierarchical clustering on the correlation coefficient matrix. PC3 separates breast samples (red) from lung samples (blue) almost as good as in previous example.

3.6 Kernel Principal Component Analysis

3.6.1 The Method

Kernel Principal Component Analysis or *kernel PCA* (KPCA) extends PCA to nonlinear projections using kernel techniques. Using a kernel, the originally linear operations of PCA are performed in a reproducing kernel Hilbert space to which the original vectors are non-linearly mapped.

We assume that the data is projected into the feature space by

$$\boldsymbol{x} \;\mapsto\; \boldsymbol{\Phi}(\boldsymbol{x}) \;. \tag{3.40}$$

Let us for the moment assume that the data is centered in the feature space, that is

$$\sum_{i=1}^{n} \boldsymbol{\Phi}(\boldsymbol{x}_i) \;=\; \mathbf{0} \;. \tag{3.41}$$

The covariance matrix in feature space is given by

$$\boldsymbol{C} \;=\; \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\Phi}(\boldsymbol{x}_i)\, \boldsymbol{\Phi}^T(\boldsymbol{x}_i) \;. \tag{3.42}$$

Note, that the Gram matrix is $\boldsymbol{K} \;=\; \sum_{i=1}^{n} \boldsymbol{\Phi}^T(\boldsymbol{x}_i)\boldsymbol{\Phi}(\boldsymbol{x}_i)$.

For PCA the eigenvectors of $\boldsymbol{C}$ should be identified, i.e. the vectors fulfilling

$$\boldsymbol{C}\,\boldsymbol{w} \;=\; \lambda\,\boldsymbol{w} \;. \tag{3.43}$$

As PCA maximizes the variance, only solutions in directions, where the data has variance, are of interest. Consequently, we restrict the solutions $\boldsymbol{w}$ to the span of $\{\boldsymbol{\Phi}(\boldsymbol{x}_1), \ldots, \boldsymbol{\Phi}(\boldsymbol{x}_n)\}$. Therefore we are searching for vectors $\boldsymbol{w}$ which fulfill

$$\begin{aligned} &\forall_{1 \,\leq\, s \,\leq\, n}: \;\; (\lambda\,\boldsymbol{w})^T \boldsymbol{\Phi}(\boldsymbol{x}_s) \;=\; \lambda\,\boldsymbol{w}^T \boldsymbol{\Phi}(\boldsymbol{x}_s) \;= \\ &(\boldsymbol{C}\,\boldsymbol{w})^T \boldsymbol{\Phi}(\boldsymbol{x}_s) \;=\; \boldsymbol{w}^T \boldsymbol{C}\,\boldsymbol{\Phi}(\boldsymbol{x}_s) \;. \end{aligned} \tag{3.44}$$

The solutions of these equations are unique in the span of the mapped data vectors and correspond to eigenvectors of $\boldsymbol{C}$ in the span.

As $\boldsymbol{w}$ is in the span of the mapped data vectors, $\boldsymbol{w}$ can be represented by

$$\boldsymbol{w} \;=\; \sum_{i=1}^{n} \alpha_i\, \boldsymbol{\Phi}(\boldsymbol{x}_i) \;. \tag{3.45}$$

Inserting this equation together with the definition of $\boldsymbol{C}$ in Eq. (3.42) into Eq. (3.44) gives

$$\begin{aligned} &\lambda \sum_{i=1}^{n} \alpha_i\, \boldsymbol{\Phi}^T(\boldsymbol{x}_i) \boldsymbol{\Phi}(\boldsymbol{x}_s) \;= \\ &\frac{1}{n} \left(\sum_{i=1}^{n} \alpha_i \sum_{j=1}^{n} \boldsymbol{\Phi}^T(\boldsymbol{x}_i) \left(\boldsymbol{\Phi}(\boldsymbol{x}_j)\, \boldsymbol{\Phi}^T(\boldsymbol{x}_j) \right) \right) \boldsymbol{\Phi}(\boldsymbol{x}_s) \;. \end{aligned} \tag{3.46}$$

We use the Gram matrix $\boldsymbol{K}$ with $K_{ij} = \boldsymbol{\Phi}^T(\boldsymbol{x}_j)\boldsymbol{\Phi}(\boldsymbol{x}_i)$. This equation holds for $1 \leq i \leq n$ and the vectors $\boldsymbol{\Phi}(\boldsymbol{x}_i)$ span the whole space, therefore we reformulate Eq. (3.46) as follows:

$$n\,\lambda\,\boldsymbol{K}\,\boldsymbol{\alpha} \;=\; \boldsymbol{K}^2\,\boldsymbol{\alpha}\,. \tag{3.47}$$

To solve this equation we solve the eigenvalue problem

$$n\,\lambda\,\boldsymbol{\alpha} \;=\; \boldsymbol{K}\,\boldsymbol{\alpha}\,. \tag{3.48}$$

The $\boldsymbol{\alpha}$ describes the eigenvector $\boldsymbol{w}$ which must have the length 1, resulting in

$$1 \;=\; \boldsymbol{w}^T\boldsymbol{w} \;=\; \sum_{ij=(1,1)}^{(n,n)} \alpha_i\,\alpha_j\,\boldsymbol{\Phi}^T(\boldsymbol{x}_j)\boldsymbol{\Phi}(\boldsymbol{x}_i) \;= \tag{3.49}$$

$$\sum_{ij=(1,1)}^{(n,n)} \alpha_i\,\alpha_j\,K_{ij} \;=\; \boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} \;=\; n\,\lambda\,\boldsymbol{\alpha}^T\boldsymbol{\alpha}\,.$$

The vector $\boldsymbol{\alpha}$ has to be normalized to fulfill

$$n\,\lambda\,\|\boldsymbol{\alpha}\|^2 \;=\; 1 \tag{3.50}$$

$$\|\boldsymbol{\alpha}\| \;=\; \frac{1}{\sqrt{n\,\lambda}} \tag{3.51}$$

by

$$\alpha_i^{\text{new}} \;=\; \frac{\alpha_i}{\|\boldsymbol{\alpha}\|\,\sqrt{n\,\lambda}}\,. \tag{3.52}$$

The projection onto $\boldsymbol{w}$ can be computed as

$$\boldsymbol{w}^T\boldsymbol{\Phi}(\boldsymbol{x}) \;=\; \sum_{i=1}^{n} \alpha_i\,\boldsymbol{\Phi}^T(\boldsymbol{x}_i)\boldsymbol{\Phi}(\boldsymbol{x}) \;=\; \sum_{i=1}^{n} \alpha_i\,k(\boldsymbol{x}_i,\boldsymbol{x})\,. \tag{3.53}$$

It can be seen that the explicit representation $\boldsymbol{\Phi}(\boldsymbol{x})$ is not necessary.

We made the assumption that the data are centered in feature space. Now we consider how this requirement can be fulfilled. We see that

$$\left(\boldsymbol{\Phi}(\boldsymbol{x}_i) \;-\; \frac{1}{n}\sum_{t=1}^{n}\boldsymbol{\Phi}(\boldsymbol{x}_t)\right)^T \left(\boldsymbol{\Phi}(\boldsymbol{x}_j) \;-\; \frac{1}{n}\sum_{t=1}^{n}\boldsymbol{\Phi}(\boldsymbol{x}_t)\right) \;= \tag{3.54}$$

$$\boldsymbol{\Phi}^T(\boldsymbol{x}_i)\boldsymbol{\Phi}(\boldsymbol{x}_j) \;-\; \frac{1}{n}\sum_{t=1}^{n}\boldsymbol{\Phi}^T(\boldsymbol{x}_t)\boldsymbol{\Phi}(\boldsymbol{x}_j) \;-\; \frac{1}{n}\sum_{t=1}^{n}\boldsymbol{\Phi}^T(\boldsymbol{x}_i)\boldsymbol{\Phi}(\boldsymbol{x}_t) \;+$$

$$\frac{1}{n^2}\sum_{(s,t)=(1,1)}^{(n,n)}\boldsymbol{\Phi}^T(\boldsymbol{x}_s)\boldsymbol{\Phi}(\boldsymbol{x}_t)$$

and that

$$\begin{aligned}
\frac{1}{n}\sum_{t=1}^{n}\boldsymbol{\Phi}^T(\boldsymbol{x}_t)\boldsymbol{\Phi}(\boldsymbol{x}_i) &= \left[\frac{1}{n}\boldsymbol{K}\,\mathbf{1}\right]_i \\
\frac{1}{n}\sum_{t=1}^{n}\boldsymbol{\Phi}^T(\boldsymbol{x}_i)\boldsymbol{\Phi}(\boldsymbol{x}_t) &= \left[\frac{1}{n}\mathbf{1}^T\boldsymbol{K}\right]_i \\
\frac{1}{n^2}\sum_{(s,t)=(1,1)}^{(n,n)}\boldsymbol{\Phi}^T(\boldsymbol{x}_s)\boldsymbol{\Phi}(\boldsymbol{x}_t) &= \frac{1}{n^2}\,\mathbf{1}^T\boldsymbol{K}\,\mathbf{1}\,.
\end{aligned} \tag{3.55}$$

Therefore the following equation produces a centered kernel matrix:

$$\boldsymbol{K} - \frac{1}{n}\,\boldsymbol{K}\,\mathbf{1}\,\mathbf{1}^T - \frac{1}{n}\,\mathbf{1}\,\mathbf{1}^T\boldsymbol{K} + \frac{1}{n^2}\left(\mathbf{1}^T\boldsymbol{K}\,\mathbf{1}\right)\mathbf{1}\,\mathbf{1}^T\,. \tag{3.56}$$

A new data point $\boldsymbol{x}$ can be centered by first computing

$$\boldsymbol{k}(\boldsymbol{x},.) = (k(\boldsymbol{x},\boldsymbol{x}_1),\ldots,k(\boldsymbol{x},\boldsymbol{x}_l))^T \tag{3.57}$$

and then

$$\boldsymbol{k}(\boldsymbol{x},.) - \frac{1}{n}\,\boldsymbol{K}\,\mathbf{1} - \frac{1}{n}\,\mathbf{1}^T\boldsymbol{k}(\boldsymbol{x},.)\,\mathbf{1} + \frac{1}{n^2}\left(\mathbf{1}^T\boldsymbol{K}\,\mathbf{1}\right)\mathbf{1}\,. \tag{3.58}$$

A pseudo code for kernel PCA is given in Alg. 3.1.

Algorithm 3.1 Kernel PCA

Given: gram matrix $\boldsymbol{K}$ with $K_{ij} = k(\boldsymbol{x}_i,\boldsymbol{x}_j)$

Centering
center the Gram matrix $\boldsymbol{K}$ according to Eq. (3.56)

Eigenvalues
compute eigenvectors $\boldsymbol{\alpha}$ and eigenvalues λ of the Gram matrix $\boldsymbol{K}$

Normalization
normalize eigenvectors $\boldsymbol{\alpha}$ according to Eq. (3.52)

Projection of a new vector
project a new vector $\boldsymbol{x}$ onto eigenvectors by first centering it using Eq. (3.58) and then project it according to Eq. (3.53)

3.6.2 Examples

Kernel PCA toy examples are shown in Figures 3.13 and 3.16.

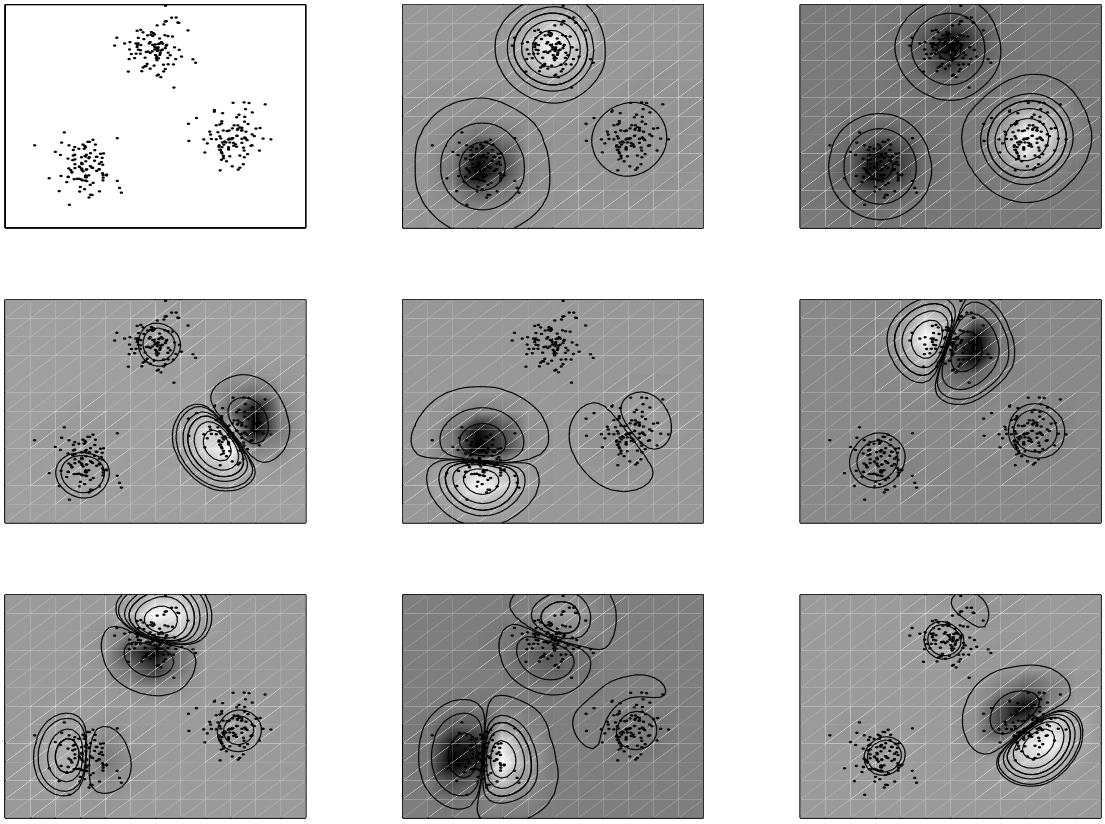


Figure 3.13: Kernel PCA example. Top left: original data. From top middle to upper right the 8 first principal components are depicted. The first two separate the clusters, the next 3 (middle panel) split the clusters, and the next 3 (bottom panel) split them orthogonal to the previous 3 components. An RBF-kernel with $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-10 \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2\right)$ was used. Figure is from Schölkopf et al. [1998].

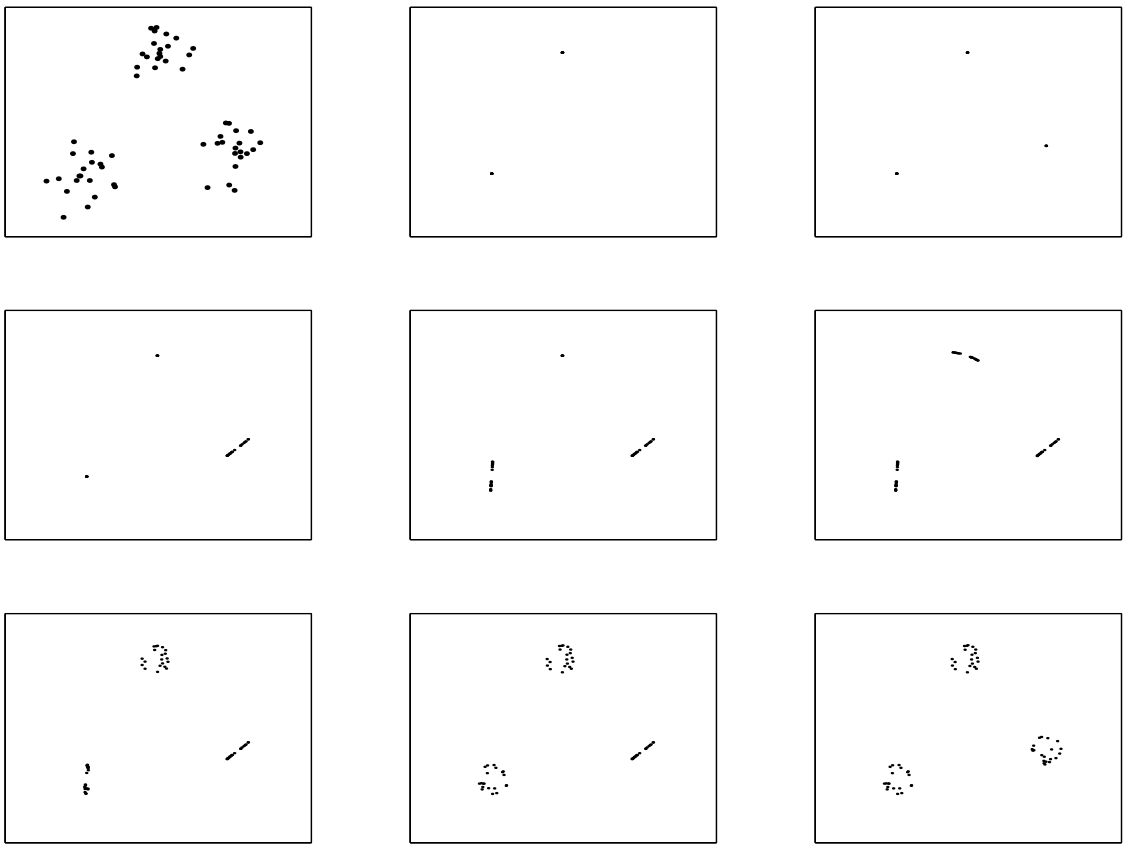


Figure 3.14: Kernel PCA example. Reconstruction by projection onto the eigenvectors. Figure is from Schölkopf et al. [1998].

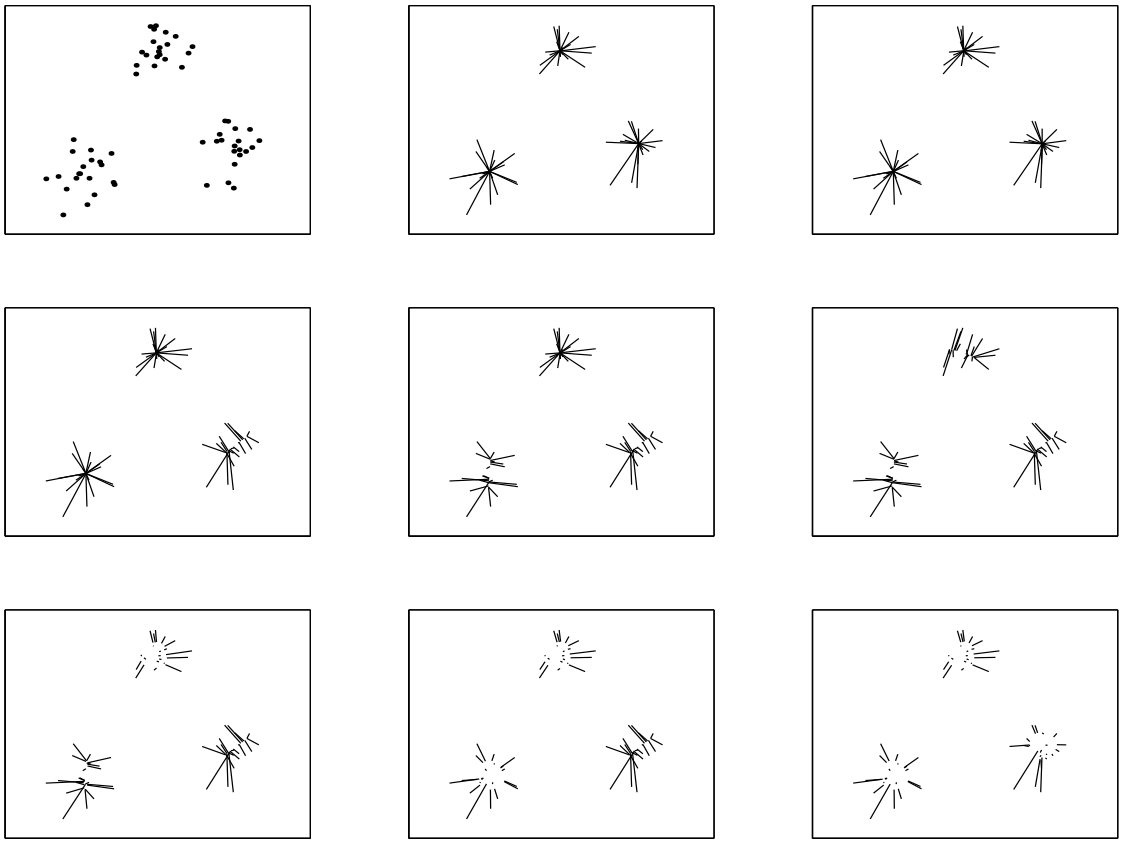


Figure 3.15: Kernel PCA example. Error or "move" avert reconstruction by projection onto eigenvectors. Figure is from Schölkopf et al. [1998].

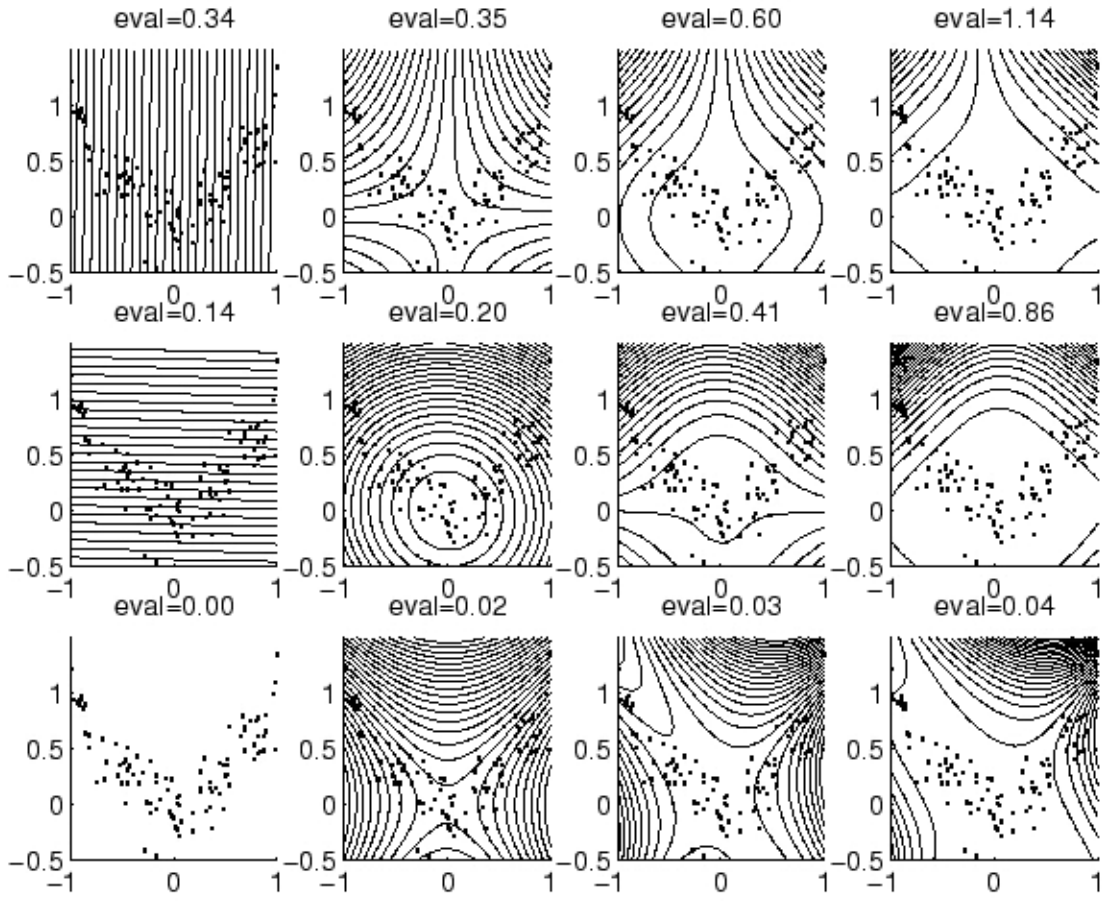


Figure 3.16: Another kernel PCA example. Each column corresponds to one method based on PCA. Each column gives the three largest eigenvalues with the according eigenvectors depicted as contour lines. In the first column linear PCA, and in the second, third and last column polynomial kernel with degree 2,3, and 4, respectively, is shown. Figure is from Schölkopf and Smola [2002].

Chapter 4

Independent Component Analysis

Independent component analysis (ICA) goes beyond PCA, which decorrelated the components, by requiring statistically independent components Schmidhuber [1992], Bell and Sejnowski [1995], Deco and Brauer [1995], Pearlmutter and Parra [1997], Cardoso and Laheld [1996], Cichocki et al. [1994], Jutten and Herault [1991], Comon [1994], Yang and Amari [1997], Yang et al. [1996], Cardoso and Souloumiac [1993], Hyvärinen et al. [2001], Hyvärinen [1999], Hochreiter and Schmidhuber [1997a,b, 1999f,b,d], Hochreiter and Mozer [2000, 2001a,c]. ICA decomposes a multivariate observation into additive components, where the components are non-Gaussian and statistically independent from each other. ICA differs from PCA in four major issues:

1. ICA does not maximize the variance,
2. ICA does not enforce orthogonal projection or demixing matrices,
3. ICA aims at statistically independent components,
4. ICA components are not ranked.

For a comparison of ICA and PCA see Hochreiter and Schmidhuber [1998, 1999g,c,e,a] and for an overview of ICA see Hyvärinen et al. [2001, 2009], Hyvärinen [1999].

ICA assumes that the observations $\boldsymbol{x}$ are generated by mixing the sources $\boldsymbol{y}$, where both $\boldsymbol{x}$ and $\boldsymbol{y}$ are m-dimensional vectors:

$$\boldsymbol{x} \;=\; \boldsymbol{U}\, \boldsymbol{y}\;. \tag{4.1}$$

The independence assumption states that the sources $\boldsymbol{y}$ are statistically independent:

$$p(\boldsymbol{y}) \;=\; \prod_{j=1}^{l} p(y_j)\;. \tag{4.2}$$

Goal is to find a matrix $\boldsymbol{W}$ with

$$\boldsymbol{y} \;=\; \boldsymbol{W}\, \boldsymbol{x}\;, \tag{4.3}$$

where for full rank matrices $\boldsymbol{W} = \boldsymbol{U}^{-1}$ holds.

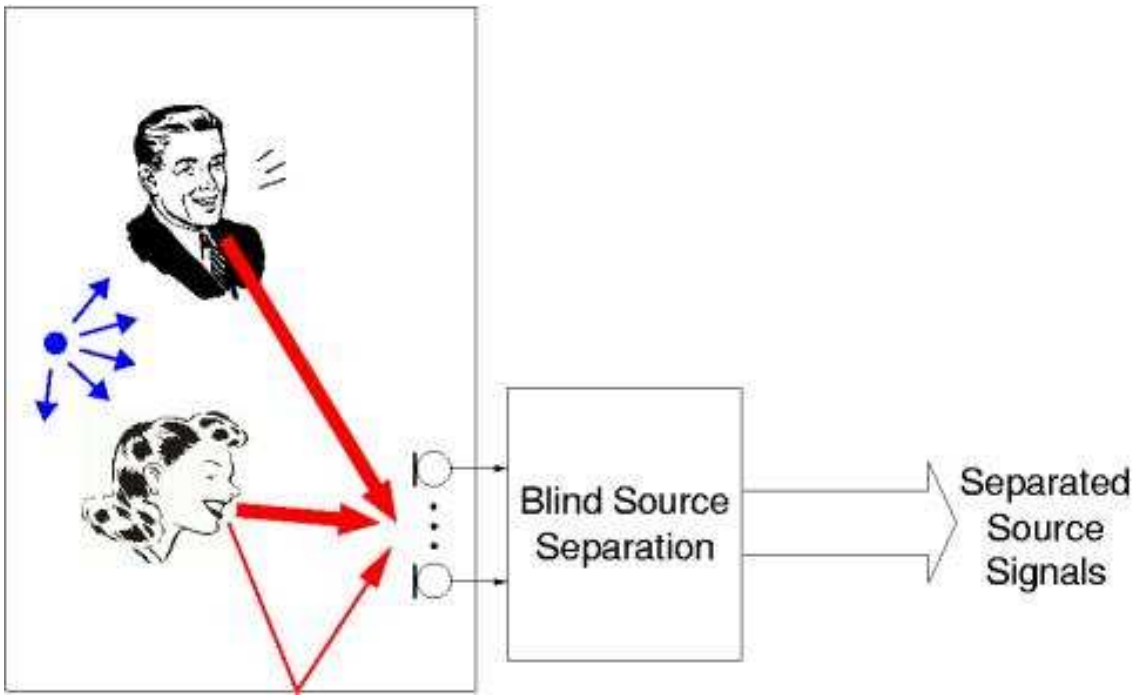


Figure 4.1: Two speakers recorded by two microphones. The speakers produce independent acoustic signals which can be separated by ICA.

Independent component analysis can be considered as a matrix decomposition. The matrix decomposition problem for ICA is:

$$\boldsymbol{X} \;=\; \boldsymbol{Y}\,\boldsymbol{U}^T\,, \tag{4.4}$$

where $\boldsymbol{Y}^T\,\boldsymbol{Y} \;=\; \boldsymbol{D}_m$, that is, independent components are decorrelated. In contrast to PCA, the components y_j must be statistically independent from each other, while $\boldsymbol{U}$ is not required to be orthogonal. ICA also works if the number of sources is l with $l \leq m$, that means the number of sources is smaller or equal the number of observations.

The outer product representation is

$$\boldsymbol{X} \;=\; \sum_{j=1}^{l} \boldsymbol{y}_j\,\boldsymbol{u}_j^T\,, \tag{4.5}$$

where $\boldsymbol{u}_j$ and $\boldsymbol{y}_j$ are the j-th column vector of $\boldsymbol{U}$ and $\boldsymbol{Y}$, respectively.

Fig. 4.1 shows how two speakers (the sources $\boldsymbol{y}$) speak independently from each other. The microphones record the acoustic signals that are the observations $\boldsymbol{x}$.

Fig. 4.2 shows the ICA solution of the data set of Fig. 3.1 and Fig. 4.3 compares the PCA and the ICA solution.

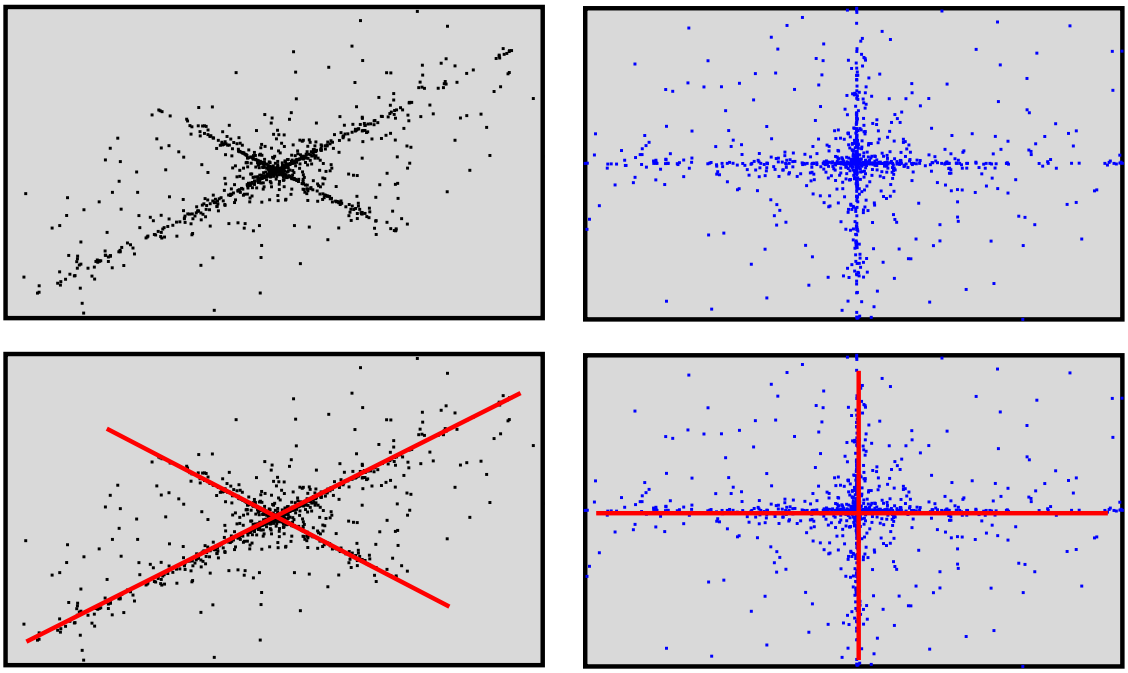


Figure 4.2: Independent component analysis on the data set of Fig. 3.1. Left sub-figures show the original data and the right sub-figures the transformed data by ICA. The figures in the lower panels show the same figures as in the upper panels but only the ICA components are added (red lines).

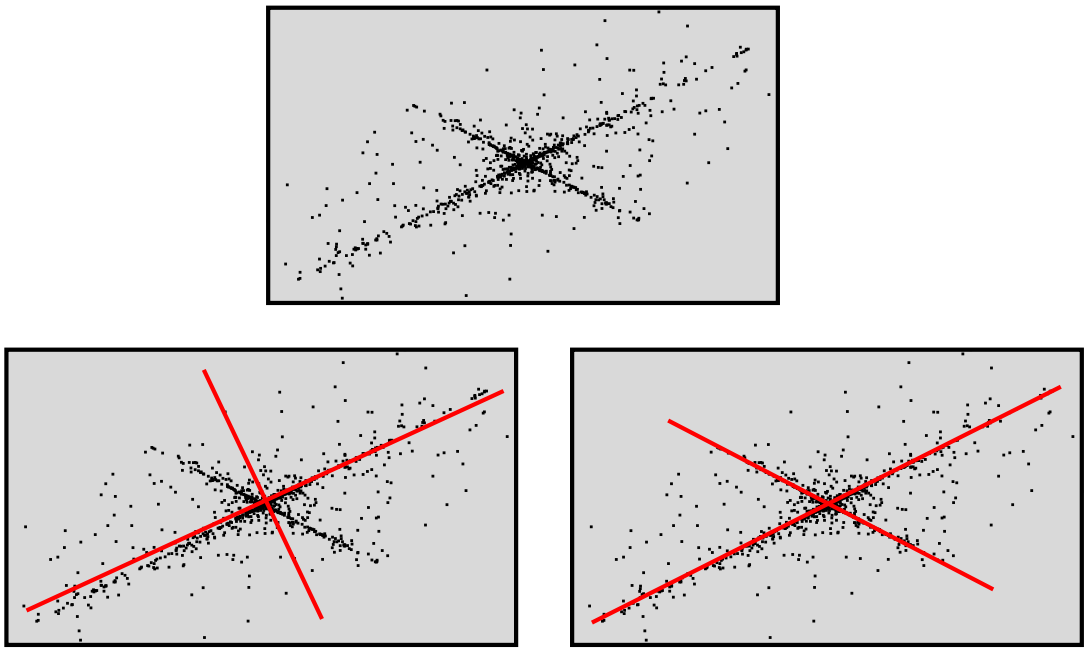


Figure 4.3: Comparison of PCA and ICA on the data set of Fig. 3.1. Top panel: original data set. Bottom left panel: PCA solution. Bottom right panel: ICA solution.

4.1 Identifiability and Uniqueness

In an optimal situation, we obtain $\boldsymbol{W} = \boldsymbol{U}^{-1}$. However the ICA solution is not unique because

$$\boldsymbol{x} = \boldsymbol{U}\,\boldsymbol{P}^{-1}\,\boldsymbol{P}\,\boldsymbol{y} \tag{4.6}$$

holds true for all full rank matrices $\boldsymbol{P}$. For another solution $\boldsymbol{Y}'$ we have

$$\boldsymbol{Y}' = \boldsymbol{P}\,\boldsymbol{Y} \tag{4.7}$$

with

$$\boldsymbol{Y}'^{T}\boldsymbol{Y}' = \boldsymbol{D}'_m\,. \tag{4.8}$$

Next we report a central theorem for identifiability and uniqueness of ICA.

Theorem 4.1 (Darmois' theorem (1953))
Define the two random variables x_1 and x_2 as

$$x_1 = \sum_{j=1}^{m} a_j y_j \quad \textit{and} \quad x_2 = \sum_{j=1}^{m} b_j y_j\,, \tag{4.9}$$

where y_i are independent random variables. Then if x_1 and x_2 are independent, all variables y_j for which $a_j b_j \neq 0$ are Gaussian.

This theorem can be found in Darmois [195] but is also reported in [Comon, 1994, THEOREM 19], [Kagan et al., 1973, p. 89], and [Rao, 1973, p. 218].

This theorem states that if two variables are independent from each other and they are a weighted sum of independent variables, then they are constructed by mutually different variables. If we have m variables x_i and m independent variables y_i from which the x_i are constructed as a weighted sum, then mutual independence of the x_i implies $x_i = a_{ij}y_j$ and $x_k \neq by_j$ for $k \neq i$ and some b.

The exception in the theorem are Gaussian distributions. This can also be seen by the fact that Gaussian mixtures may still factorize, that is, their components are independent. If the Gaussian variable $\boldsymbol{y}$ factorizes, then the covariance matrix is diagonal. If the covariance matrix is even σ^2 times the identity, then an orthogonal transformation $\boldsymbol{x} = \boldsymbol{U}\boldsymbol{y}$ of the variables gives again σ^2 times the identity as covariance matrix, therefore, the distribution of the Gaussian $\boldsymbol{x}$ factorizes.

Since we exclude Gaussian sources, $\boldsymbol{P}$ cannot mix the statistically independent components of $\boldsymbol{y}$ when calculating $\boldsymbol{Y}'$. Otherwise the components of $\boldsymbol{Y}'$ would be statistically dependent. Therefore $\boldsymbol{P}$ is a permutation matrix multiplied with a diagonal scaling matrix. **The ICA solution is for non-Gaussian sources unique up to permutation and scaling** Hyvärinen and Pajunen [1999].

Following assumptions ensure the ICA solution to be unique up to permutations and scalings:

- the source components y_i are non-Gaussian (except for at most one component),
- observation dimension m must be at least as large as the number of sources l: $m \geq l$,

- the mixing matrix $\boldsymbol{U}$ must have full rank l.

We will combine the last two assumptions by assuming that the observations $\boldsymbol{x}$ and the sources $\boldsymbol{y}$ live in an m-dimensional space and $\boldsymbol{U}^{-1} \in \mathbb{R}^{m \times m}$ exists. Since $\boldsymbol{W} = \boldsymbol{U}^{-1}$ states that the model in invertible, a generative framework can be used to find a descriptive model (see beginning of this chapter). However for complete the generative framework we have to make assumptions on the densities $p(y_i)$. These densities are often approximated, e.g. by super-Gaussians, where it turned out that different unimodal super-Gaussian distributions work well (approximate the true distribution well enough). Thus, a generative framework based on maximum likelihood can be used to infer both the mixing and demixing matrix.

4.2 Measuring Independence

Similar to a goodness of fit for linear regression we require an objective for measuring independence. We have to measure independence of the components y_i of $\boldsymbol{y}$ in order to assess the quality of an ICA solution. The y_i should not only be pairwise independent but for each i the following should hold:

$$p(y_i \mid y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_l) \;=\; p(y_i)\;. \tag{4.10}$$

Independence of the components y_i are in most ICA methods measured by one of following two criteria:

- mutual information between the y_i, or
- non-Gaussianity of the y_i.

4.2.1 Mutual Information

The entropy of a factorial code is larger than the entropy of the joint distribution. The difference of the two expressions is the mutual information I between the variables y_j:

$$I(y_1, \ldots, y_l) \;=\; \sum_{j=1}^{l} H(y_j) \;-\; H(\boldsymbol{y})\;, \tag{4.11}$$

where H denotes the entropy

$$H(\boldsymbol{a}) \;=\; -\int p(\boldsymbol{a})\;\ln p(\boldsymbol{a})\;d\boldsymbol{a}\;. \tag{4.12}$$

If we set

$$\boldsymbol{y} \;=\; \boldsymbol{W}\,\boldsymbol{x} \tag{4.13}$$

then

$$I(y_1, \ldots, y_m) \;=\; \sum_{j=1}^{m} H(y_j) \;-\; H(\boldsymbol{x}) \;-\; \ln|\boldsymbol{W}|\;, \tag{4.14}$$

where $|\boldsymbol{W}|$ is the absolute value of the determinant of the matrix $\boldsymbol{W}$. This equation stems from the fact that

$$p(\boldsymbol{y}) \;=\; \frac{p(\boldsymbol{x})}{|\boldsymbol{W}|}\,. \tag{4.15}$$

4.2.2 Non-Gaussianity

The negentropy is defined as

$$J(\boldsymbol{y}) \;=\; H(\boldsymbol{y}_{\mathrm{gauss}}) \;-\; H(\boldsymbol{y})\,, \tag{4.16}$$

where $\boldsymbol{y}_{\mathrm{gauss}}$ is a Gaussian random vector with the same covariance matrix as $\boldsymbol{y}$. The negentropy is an affine invariant version of the entropy.

The maximal negentropy is equivalent to representations where the mutual information between the components is minimized. Here the connection between ICA and projection pursuit is clear because both can be expressed through maximizing the distance to Gaussian distributions. The Gaussian is the distribution with the largest entropy given the mean and the variance. Therefore negentropy maximization is closely related to entropy maximization.

Unfortunately, the negentropy cannot be used easily because its estimation is difficult. The non-Gaussianity can be measured through other parameters for example by cummulants.

For zero mean variables the cummulants are defined as

$$\kappa_1 \;=\; \mathrm{E}(x) \;=\; 0 \tag{4.17}$$

$$\kappa_2 \;=\; \mathrm{E}(x^2) \tag{4.18}$$

$$\kappa_3 \;=\; \mathrm{E}(x^3) \tag{4.19}$$

$$\kappa_4 \;=\; \mathrm{E}(x^4) \;-\; 3\left(\mathrm{E}(x^2)\right)^2\,. \tag{4.20}$$

$\frac{\kappa_4}{\kappa_2^2}$ is called *kurtosis* or the *excess kurtosis*. For normal distributions we have $\kappa_3 = \kappa_4 = 0$. Therefore, the fourth cummulant and the kurtosis are common measures for non-Gaussianity. Positive kurtosis indicates super-Gaussians, the tails are smaller than for Gaussians, and negative kurtosis indicates sub-Gaussians the tails are larger than for Gaussians.

For x_1 and x_2 independent, the 4th cummulant is a linear function:

$$\kappa_4(x_1 \;+\; x_2) \;=\; \kappa_4(x_1) \;+\; \kappa_4(x_2) \tag{4.21}$$

$$\kappa_4(\alpha\, x) \;=\; \alpha^4\, \kappa_4(x)\,. \tag{4.22}$$

For super-Gaussians y_i the kurtosis should be maximized because mixtures have a smaller kurtosis than the original sources. In Section 4.11, we show that maximizing the kurtosis of a mixture of super-Gaussian signals recovers one of the signals up to scaling and permutation.

Instead of maximizing the kurtosis, the sparseness (see Földiák [1990], Young and Yamane [1992], Rolls and Tovee [1995], Olshausen and Field [1996], Hinton and Ghahramani [1997], Girolami [2001]) of y_i can be maximized to make the y_i independent from each other. Sparseness means that a variable is in most cases zero and rarely deviates from zero (see Fig. 4.4). If it deviates from zero the values may be relatively large (compared to Gaussian with the same variance).

Therefore, sparseness does not mean small variance. If the variance is fixed: the more values are zero, the larger are the non-zero values as the variance must be kept. Maximal kurtosis and maximal sparseness are equivalent, because kurtosis is a measure of sparseness [Hyvärinen et al., 2009, p. 133, Sec. 6.2.2]. Maximizing the kurtosis of the $\boldsymbol{y}$ components finds independent, zero centered, symmetric super-Gaussian signals. On the other hand, super-Gaussians at zero produce many zero values, thus they are sparse. However kurtosis is not a robust measure as it is based on the fourth moment which is easily affected by outliers.

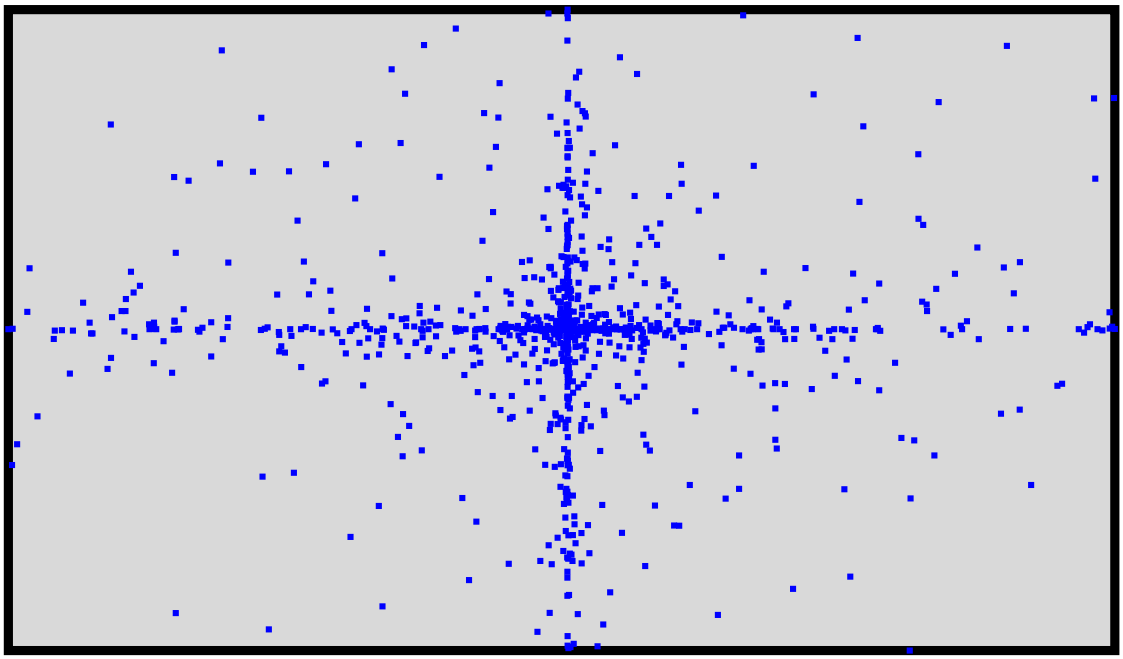


Figure 4.4: Sparse data. Most points are on the axis which means that the according component is zero. At the same time the components have a large kurtosis.

Many ICA algorithms use *contrast functions* which measure the independence of the variables and are used as objective functions. Common contrast functions are

- $\kappa_4(y)$ or the kurtosis $\frac{\kappa_4(y)}{\kappa_2^2(y)}$ (see above),
- $\frac{1}{12}\,\kappa_3^2(y) + \frac{1}{48}\,\kappa_4^2(y)$, where the variable y is normalized to zero mean and unit variance,
- $|\mathrm{E}_y(G(y)) - \mathrm{E}_\nu(G(\nu))|^p$, where ν is a standardized Gaussian, $p = 1, 2$, and y is normalized to zero mean and unit variance. Here G can be the kurtosis for which $G(\nu) = 0$ would hold. Other choices for G are $G(x) = \log\cosh(ax)$ and $G(x) = \exp(-ax^2/2)$ with $a \geq 1$.

4.3 Whitening and Rotation Algorithms

Whitening and rotation algorithms measure independence by non-Gaussianity, e.g. via the kurtosis. A well known algorithm of this kind is FastICA Hyvärinen and Oja [1999], Hyvärinen et al. [2001] which we present later.

ICA requires $\boldsymbol{Y}^T\,\boldsymbol{Y} \;=\; \boldsymbol{D}_l$, but as the sources cannot be determined up to scaling, we can require $\boldsymbol{Y}^T\,\boldsymbol{Y} \;=\; \boldsymbol{I}$, that is *whitened data* or *sphered data* $\boldsymbol{Y}$. Consequently, for both ICA and PCA, we require $\boldsymbol{Y}^T\,\boldsymbol{Y} \;=\; \boldsymbol{I}$. Therefore, as a first step in ICA, PCA is performed for whitening (sphering) the data. The matrix decompositions for ICA is

$$\boldsymbol{X} \;=\; \boldsymbol{Y}\,\boldsymbol{U}^T \tag{4.23}$$

which gives

$$\boldsymbol{Y} \;=\; \boldsymbol{X}\,\boldsymbol{U}^{-T}\,. \tag{4.24}$$

Using the co-variance matrix $\boldsymbol{C}$ we have

$$\begin{aligned}\boldsymbol{I} \;&=\; \boldsymbol{C}^{-1/2}\,\underbrace{\frac{1}{n}\,\boldsymbol{X}^T\boldsymbol{X}}_{\boldsymbol{C}}\,\boldsymbol{C}^{-1/2} \\ &=\; \frac{1}{n}\,\boldsymbol{C}^{-1/2}\boldsymbol{U}\,\boldsymbol{Y}^T\,\boldsymbol{Y}\,\boldsymbol{U}^T\,\boldsymbol{C}^{-1/2} \;=\; \frac{1}{n}\,\boldsymbol{C}^{-1/2}\,\boldsymbol{U}\,\boldsymbol{U}^T\,\boldsymbol{C}^{-1/2}\,.\end{aligned} \tag{4.25}$$

Thus, the matrix $\hat{\boldsymbol{U}} \;=\; \frac{1}{\sqrt{n}}\,\boldsymbol{C}^{-1/2}\,\boldsymbol{U}$ is orthogonal ($\boldsymbol{C}^{-1/2}$ is symmetric and $\hat{\boldsymbol{U}}^T$ is orthogonal). We have

$$\boldsymbol{Y} \;=\; \frac{\sqrt{n}}{\sqrt{n}}\,\boldsymbol{X}\;\boldsymbol{C}^{-1/2}\,\boldsymbol{C}^{1/2}\boldsymbol{U}^{-T} \;=\; \frac{1}{\sqrt{n}}\,\boldsymbol{X}\,\boldsymbol{C}^{-1/2}\,\hat{\boldsymbol{U}}^{-T} \;=\; \frac{1}{\sqrt{n}}\,\boldsymbol{X}\,\boldsymbol{C}^{-1/2}\,\hat{\boldsymbol{U}}\,, \tag{4.26}$$

because $\hat{\boldsymbol{U}}$ is orthogonal.

Subsequent to whitening $\boldsymbol{X}\,\boldsymbol{C}^{-1/2}$, we have to determine the orthogonal matrix $\hat{\boldsymbol{U}}$ in order to obtain $\boldsymbol{Y}$. This is just a rotation of the whitened data, because ICA is not unique up to permutations and changing signs. To find a proper rotation, we assume that

- the components of $\boldsymbol{y}$ are super-Gaussian OR
- the components of $\boldsymbol{y}$ are sparse that is most components are zero.

For both assumptions we have a heavy-tailed distribution with a sharp peak at zero. The kurtosis is larger than zero (the kurtosis of a Gaussian). ICA is looking for a rotation matrix which either maximizes the kurtosis or supplies sparse projections. Instead of the kurtosis other contrast functions can be used to identify super-Gaussians (see above).

In Subsection 4.9.1 whitening and rotation is demonstrated on a toy data set.

4.4 INFOMAX Algorithm

The INFOMAX algorithm Bell and Sejnowski [1995] is based on achieving independent components by minimizing the mutual information. The mutual information is minimized by maximizing the entropy $H(\boldsymbol{g}(\boldsymbol{y}))$, where $\boldsymbol{g}(\boldsymbol{y})$ is the vector $\boldsymbol{g}(\boldsymbol{y}) = (g(y_1), g(y_2), \ldots, g(y_l))$ with some function g. The vector $\boldsymbol{y}$ is computed by

$$\boldsymbol{y} \;=\; \boldsymbol{W}\,\boldsymbol{x}\,. \tag{4.27}$$

If the entropy is maximized then

$$I(g(y_1), \ldots, g(y_l)) \;=\; \sum_{j=1}^{l} H(g(y_j)) \;-\; H(\boldsymbol{g}(\boldsymbol{y})) \;=\; 0 \tag{4.28}$$

and the components $(g(y_1), \ldots, g(y_l))$ are statistically independent. A very common choice for g is

$$g(y_i) \;=\; \tanh(y_i)\,. \tag{4.29}$$

We have

$$p(\boldsymbol{g}(\boldsymbol{y})) \;=\; p(\boldsymbol{x}) \left| \frac{\partial \boldsymbol{g}(\boldsymbol{y})}{\partial \boldsymbol{y}} \, \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \right|^{-1} \;=\; p(\boldsymbol{x}) \left| \frac{\partial \boldsymbol{g}(\boldsymbol{y})}{\partial \boldsymbol{y}} \boldsymbol{W} \right|^{-1} \tag{4.30}$$

where

$$\left| \frac{\partial \boldsymbol{g}(\boldsymbol{y})}{\partial \boldsymbol{y}} \boldsymbol{W} \right| \;=\; \left| \prod_{j=1}^{l} g'(y_j) \right| \, |\boldsymbol{W}|\;. \tag{4.31}$$

Here we see the connection of g' and the density p of the sources from Subsection 4.2.1: $g'(y_i) = p(y_i)$. Thus, the function g can be considered as an affine transformed probability function.

The entropy is

$$\begin{aligned} H(\boldsymbol{g}(\boldsymbol{y})) \;&=\; \mathrm{E}\left(-\,\ln p(\boldsymbol{g}(\boldsymbol{y}))\right) \\ &=\; H(\boldsymbol{x}) \;+\; \mathrm{E}\left(\sum_{j=1}^{l} \left| \ln g'(y_j) \right| \right) \;+\; \ln |\boldsymbol{W}| \\ &\approx\; H(\boldsymbol{x}) \;+\; \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{l} \left| \ln g'(y_{ij}) \right| \;+\; \ln |\boldsymbol{W}|\;, \end{aligned} \tag{4.32}$$

where

$$\boldsymbol{y}_i \;=\; \boldsymbol{W}\, \boldsymbol{x}_i\,. \tag{4.33}$$

Now we can maximize the entropy using

$$g(y_j) \;=\; \tanh(y_j) \tag{4.34}$$

which gives

$$\frac{\partial}{\partial \boldsymbol{w}_j} \ln g'(y_j) \;=\; \frac{g''(y_j)}{g'(y_j)}\, \boldsymbol{x}^T \;=\; -\,2\, g(y_j)\, \boldsymbol{x}^T\,. \tag{4.35}$$

If we use instead of $\tanh$ a sigmoid activation function

$$g(y_j) \;=\; \frac{1}{1 \;+\; e^{-y_j}}\,, \tag{4.36}$$

then this formula becomes

$$\frac{\partial}{\partial \boldsymbol{w}_j} \ln g'(y_j) \;=\; (1 \;-\; 2\, g(y_j))\; \boldsymbol{x}^T \;. \tag{4.37}$$

Further we have

$$\frac{\partial}{\partial \boldsymbol{W}} \ln |\boldsymbol{W}| \;=\; \left(\boldsymbol{W}^T\right)^{-1} \;. \tag{4.38}$$

Since $H(\boldsymbol{x})$ does not depend on $\boldsymbol{W}$, we obtain

$$\frac{\partial}{\partial \boldsymbol{W}} H(\boldsymbol{g}(\boldsymbol{y})) \;=\; \left(\boldsymbol{W}^T\right)^{-1} \;-\; 2\, \boldsymbol{g}(\boldsymbol{y})\; \boldsymbol{x}^T \tag{4.39}$$

for $\tanh$ and for the sigmoid activation function

$$\frac{\partial}{\partial \boldsymbol{W}} H(\boldsymbol{g}(\boldsymbol{y})) \;=\; \left(\boldsymbol{W}^T\right)^{-1} \;+\; (\mathbf{1} \;-\; 2\, \boldsymbol{g}(\boldsymbol{y}))\; \boldsymbol{x}^T \tag{4.40}$$

for the derivatives.

The update rules are for the $\tanh$ activation function

$$\Delta \boldsymbol{W} \;\propto\; \left(\boldsymbol{W}^T\right)^{-1} \;-\; 2\, \boldsymbol{g}(\boldsymbol{y})\; \boldsymbol{x}^T \tag{4.41}$$

and for the sigmoid activation function

$$\Delta \boldsymbol{W} \;\propto\; \left(\boldsymbol{W}^T\right)^{-1} \;+\; (\mathbf{1} \;-\; 2\, \boldsymbol{g}(\boldsymbol{y}))\; \boldsymbol{x}^T \;. \tag{4.42}$$

For this update rule, the natural gradient can be applied which takes the geometrical structure of the parameter space into account. In this case the update rule is multiplied with $\boldsymbol{W}^T\boldsymbol{W}$. The update rule is now for the $\tanh$ activation function

$$\Delta \boldsymbol{W} \;\propto\; \left(\boldsymbol{I} \;-\; 2\, \boldsymbol{g}(\boldsymbol{y})\, \boldsymbol{y}^T\right)\; \boldsymbol{W} \tag{4.43}$$

and for the sigmoid activation function

$$\Delta \boldsymbol{W} \;\propto\; \left(\boldsymbol{I} \;+\; (\mathbf{1} \;-\; 2\, \boldsymbol{g}(\boldsymbol{y}))\; \boldsymbol{y}^T\right)\; \boldsymbol{W} \;. \tag{4.44}$$

In the last equations we used $\boldsymbol{y}^T = \boldsymbol{x}^T \boldsymbol{W}^T$.

INFOMAX is equivalent to a generative approach using maximum likelihood, when $g_i'(y_i) = p(y_i)$. In this case ICA is viewed as a generative approach. The generative approach assumes that the observations are produced by a model with independent factors or latent variables Cardoso [1997], Moulines et al. [1997], Yang and Amari [1997], Pearlmutter and Parra [1997]. Alternatively, an objective or contrast function indicating statistically independent components can be used (see e.g. Hyvärinen [1999]).

4.5 EASI Algorithm

The Equivariant Adaptive Separation via Independence (EASI) algorithm Cardoso and Laheld [1996] uses the following update rule to adjust $\boldsymbol{W}$:

$$\Delta \boldsymbol{W} \;\propto\; \left(\boldsymbol{I} \;-\; \boldsymbol{y}\, \boldsymbol{y}^T \;-\; \boldsymbol{g}(\boldsymbol{y})\boldsymbol{y}^T \;+\; \boldsymbol{y}\, \boldsymbol{g}^T(\boldsymbol{y})\right)\; \boldsymbol{W} \;. \tag{4.45}$$

The nonlinear functions g are the same contrast functions as used by the INFOMAX algorithm.

4.6 FastICA Algorithm

The FastICA algorithm Hyvärinen and Oja [1999], Hyvärinen et al. [2001] is probably the most popular ICA algorithm and an example for a whitening and rotation algorithm as previously mentioned. FastICA is a fixed point algorithm (like Oja's rule for PCA) which extracts one weight vector $\boldsymbol{w}$. Originally FastICA was based on the kurtosis maximization but it was extended to other contrast functions.

The iteration step of the FastICA algorithm is

$$\boldsymbol{w}^{\text{new}} \;=\; \mathrm{E}\left(\boldsymbol{x}\; g(\boldsymbol{w}^T\,\boldsymbol{x})\right) \;-\; \mathrm{E}\left(g'(\boldsymbol{w}^T\,\boldsymbol{x})\right)\;\boldsymbol{w}\;, \tag{4.46}$$

where g' is the derivative of the contrast function g, which has been discussed previously. Instead of the expectation E the empirical mean over the training examples is used. The FastICA algorithm has been extended to extract multiple components simultaneously.

4.7 ICA Extensions

Following extensions to the ICA algorithm have been proposed:

- generative approach assuming specific sparse distributions like see $g_i'(y_i) \;=\; p(y_i)$ to derive the INFOMAX algorithm,
- sub-Gaussian distributions with specific assumptions,
- non-linear extensions which are often not unique Hochreiter and Schmidhuber [1997a,b, 1998, 1999a,f,b,d], Hochreiter and Mozer [2000],
- overcomplete basis: more observations $\boldsymbol{x} \in \mathbb{R}^m$ than sources $\boldsymbol{y} \in \mathbb{R}^l$, $l > m$; most approaches use sparseness of $\boldsymbol{y}$ as the criterion Lewicki and Sejnowski [1998], Lewicki and Olshausen [1998], Zibulevsky and Pearlmutter [2001],
- fewer sources $\boldsymbol{y} \in \mathbb{R}^l$ than observations $\boldsymbol{x} \in \mathbb{R}^m$ with $l < m$, fewer components than dimension of the observations; standard ICA can be used if only l components of the observations are selected from the m available components.

4.8 ICA vs. PCA

Major differences between ICA and PCA are:

independent component analysis	principal component analysis
causes of the data	geometrical abstractions
statistical independent	decorrelated (orthogonal)
explain super-Gaussians	explain all variance
scale invariant	not scale invariant
unique up to scale and permutation	unique
assume super-Gauss	no assumptions
no ranking	ranked by eigenvalues

4.9 Artificial ICA Examples

We present some ICA examples including toy and real world data sets. First we apply ICA to artificial data sets and then move on to real world data sets.

4.9.1 Whitening and Rotation

We demonstrate the principle of the whitening and rotation algorithm on artificial data. 1,000 data points drawn from uniform distributions. Fig. 4.5 shows the scatter plot of the two random variables which serve as sources.

The two independently drawn sets of data points (the sources) are now mixed. The resulting mixed components are dependent because they are constructed from the same sources. Fig. 4.6 shows the scatter plot of these linearly mixed components, that is, a linear transformation of the sources from Fig. 4.5.

We apply independent component analysis (ICA) to the mixed data. ICA decorrelates the data like PCA does, that is the covariance is the identity. Exactly that is the first step of ICA: to whiten the data, that is to transform the data to obtain data with the identity as covariance matrix. Before whitening the data has to be centered. Fig. 4.7 shows a scatter plot of the data after whitening.

Orthogonal transformations do not affect the covariance structure of the data. Therefore we rotate the data until the components are maximally independent. Fig. 4.8 shows the whitened data rotated.

We do the same steps again but now for for super-Gaussians. We first generate super-Gaussian data. Fig. 4.9 shows the scatter plot for super-Gaussian components.

Fig. 4.10 shows the mixture of the super-Gaussians.

We whiten these data. Fig. 4.11 shows the super-Gaussians mixture after whitening.

Again the final step is to rotate the data. Here the data is rotated until the kurtosis of the components is maximized. Fig. 4.12 shows the result of ICA on super-Gaussian mixtures.

We determined the rotation directly from the mixing matrix which is cheating. ICA has to determine the rotation based on maximizing the kurtosis or other contrast functions. This is an iterative update algorithm. Fig. 4.13 shows the result as a scatter plot.

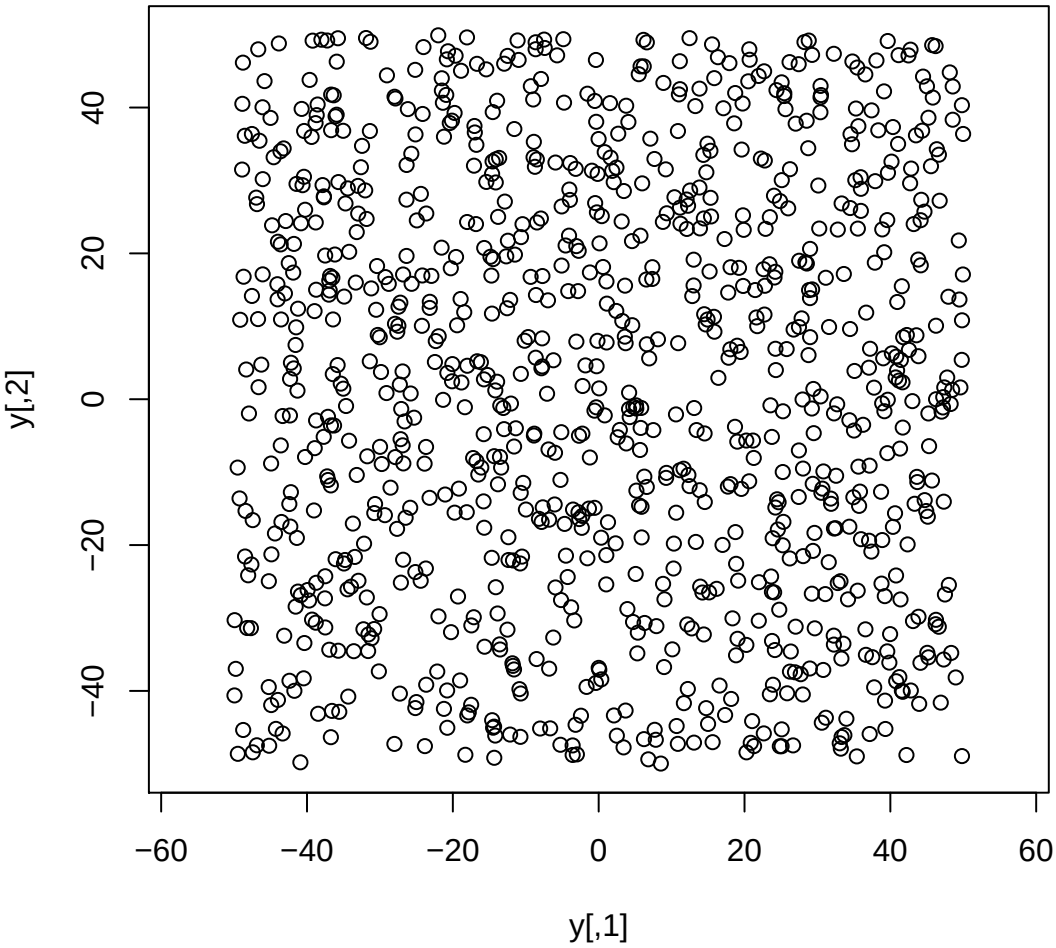


Figure 4.5: The original data. 1,000 two-dimensional data points where each component is independently drawn from an uniform distribution.

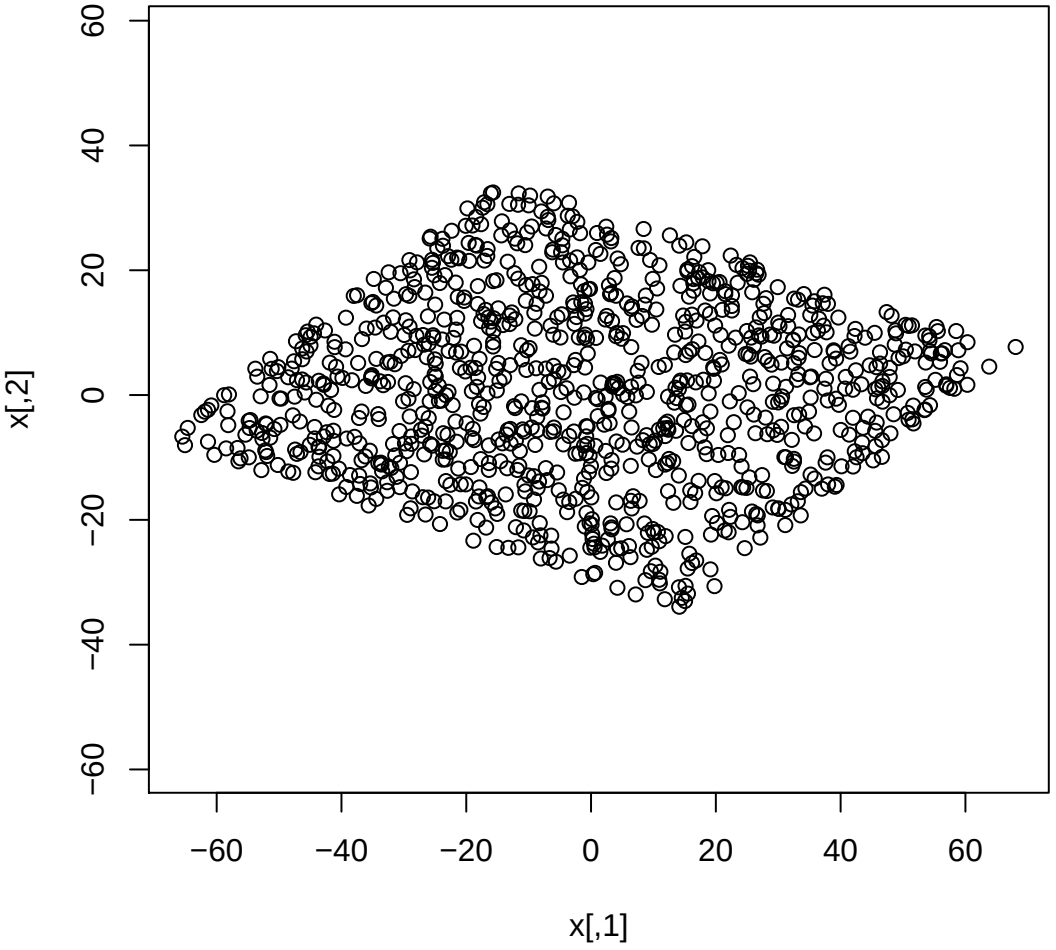


Figure 4.6: The mixed data. The original data are linearly mixed, that is, linearly transformed.

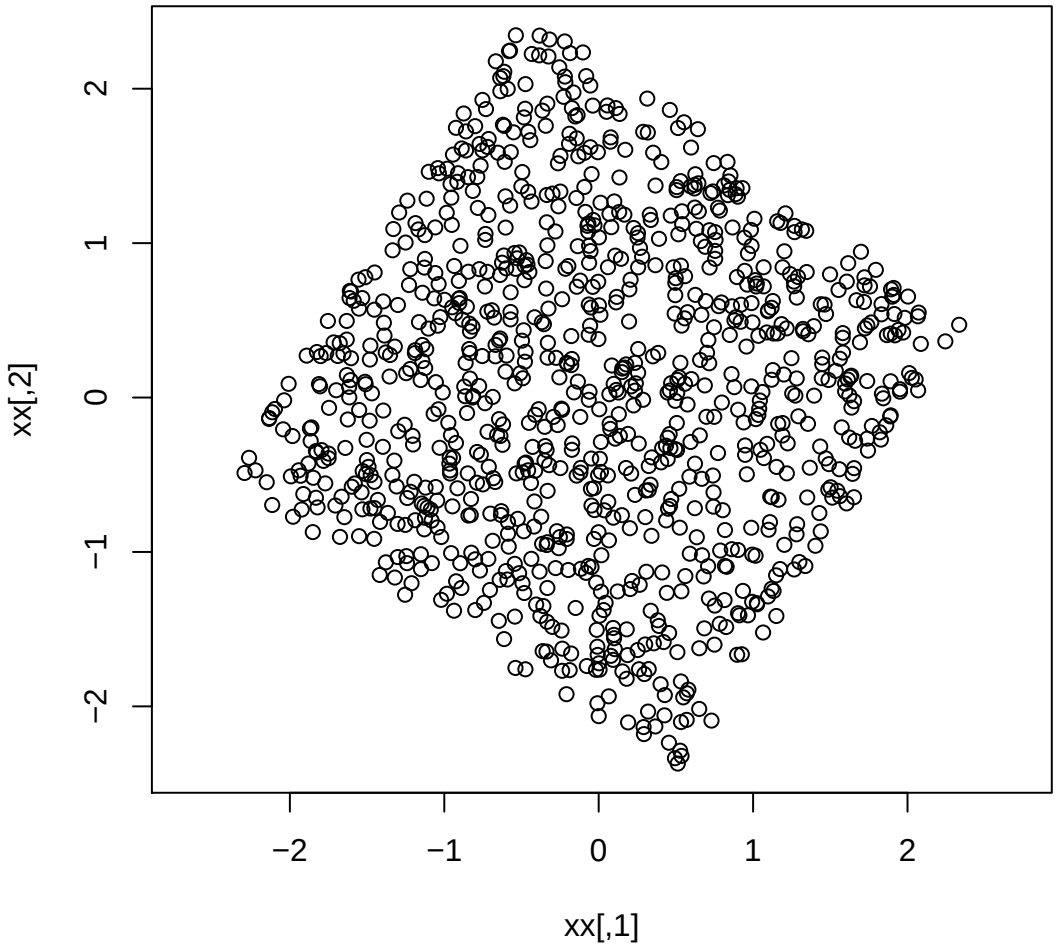


Figure 4.7: The data after whitening. The covariance matrix is now the identity matrix.

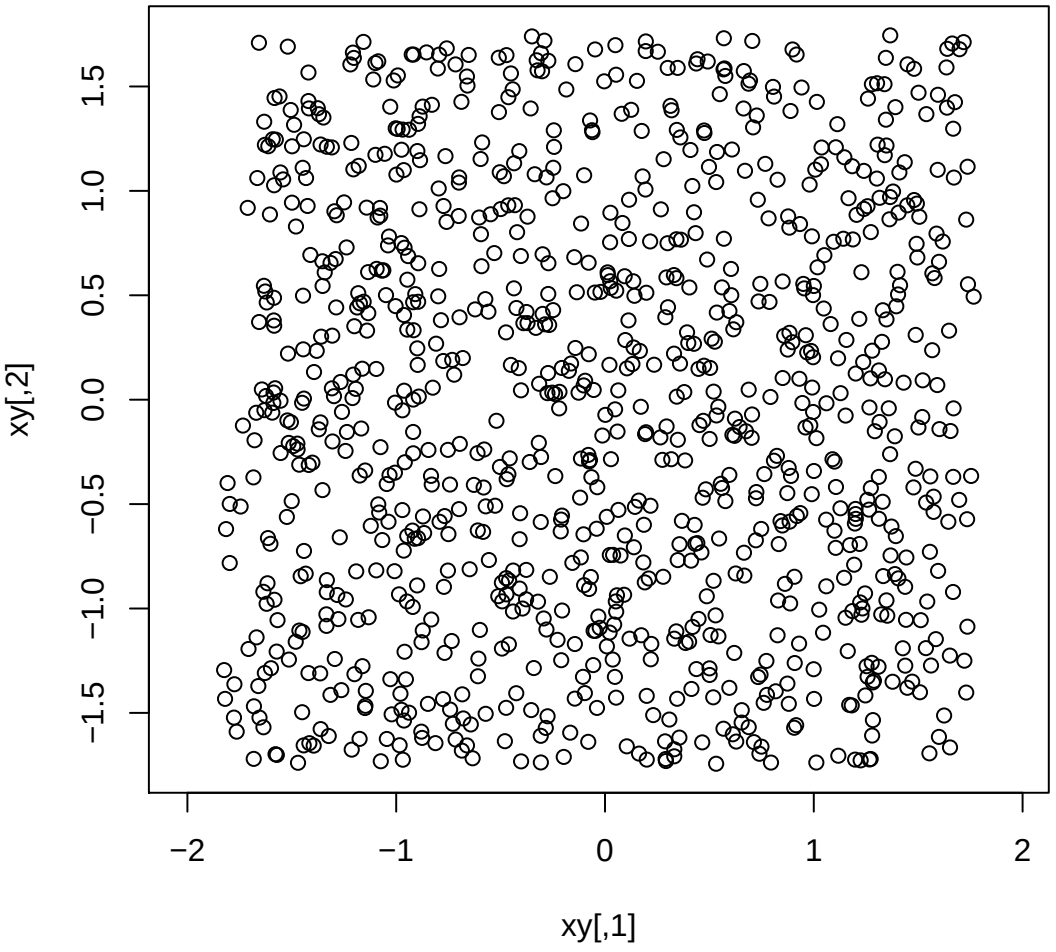


Figure 4.8: The whitened data rotated: this is the ICA solution.

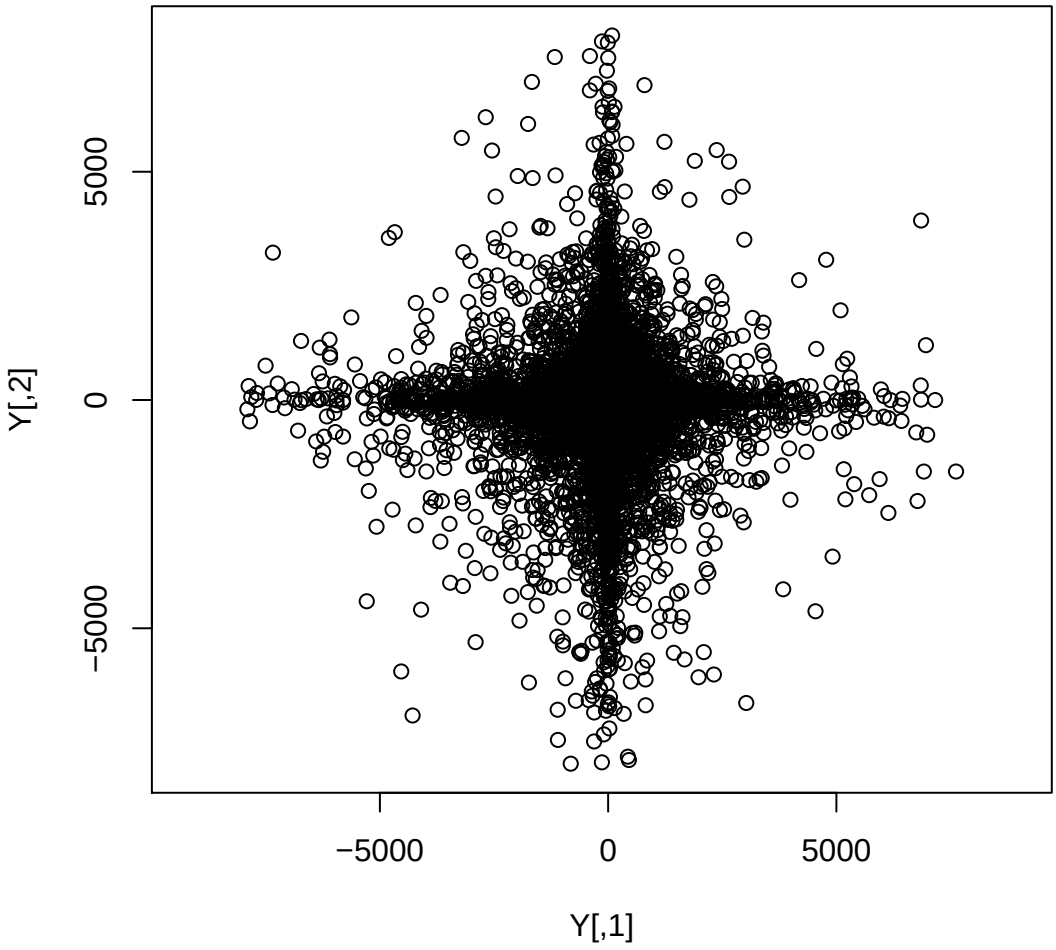


Figure 4.9: Original data with super-Gaussian components.

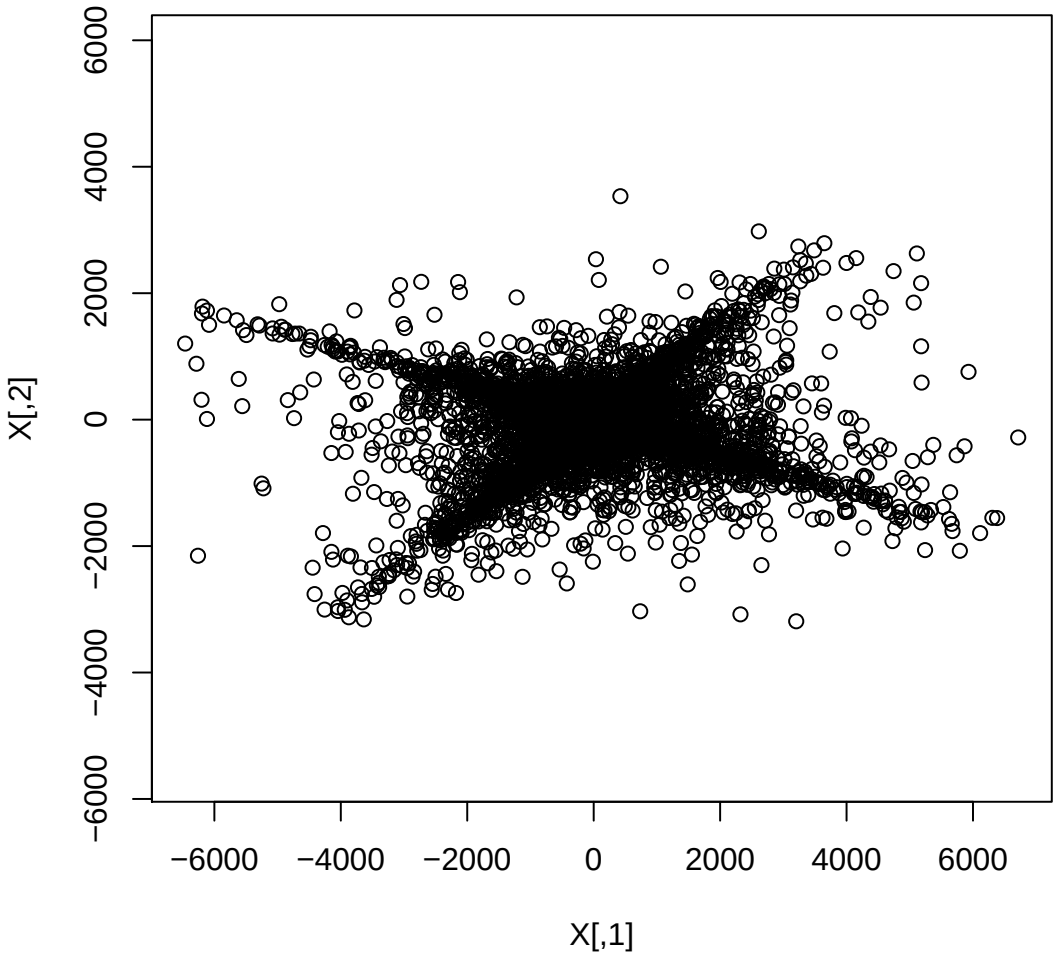


Figure 4.10: Mixture of the super-Gaussians.

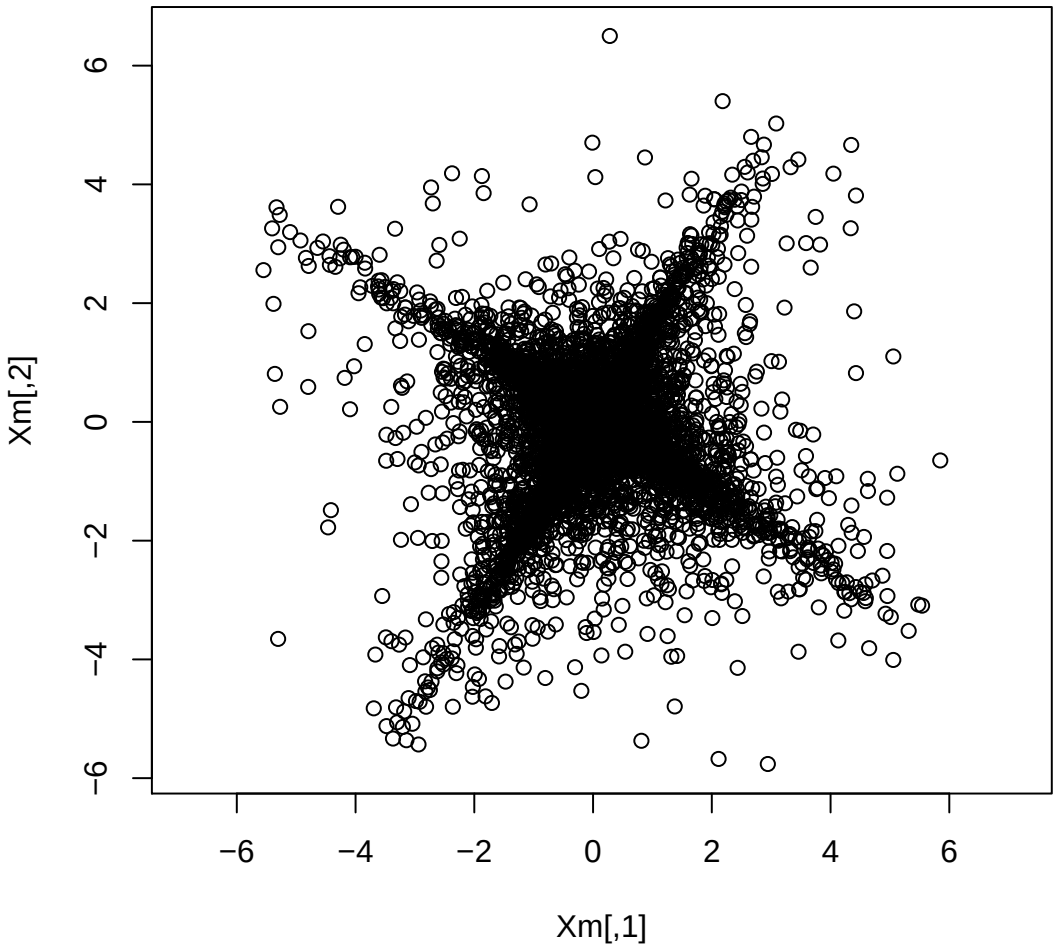


Figure 4.11: The super-Gaussians mixture after whitening.

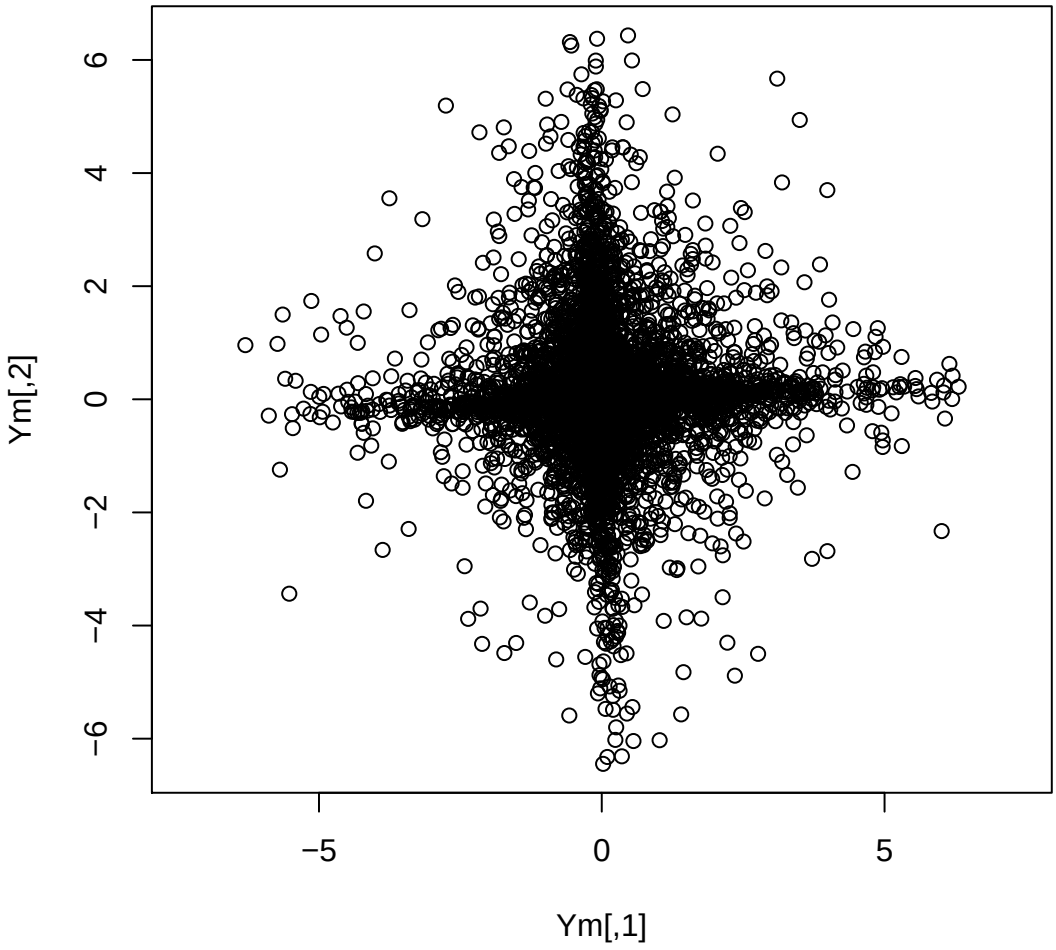


Figure 4.12: Result of ICA on super-Gaussian mixtures.

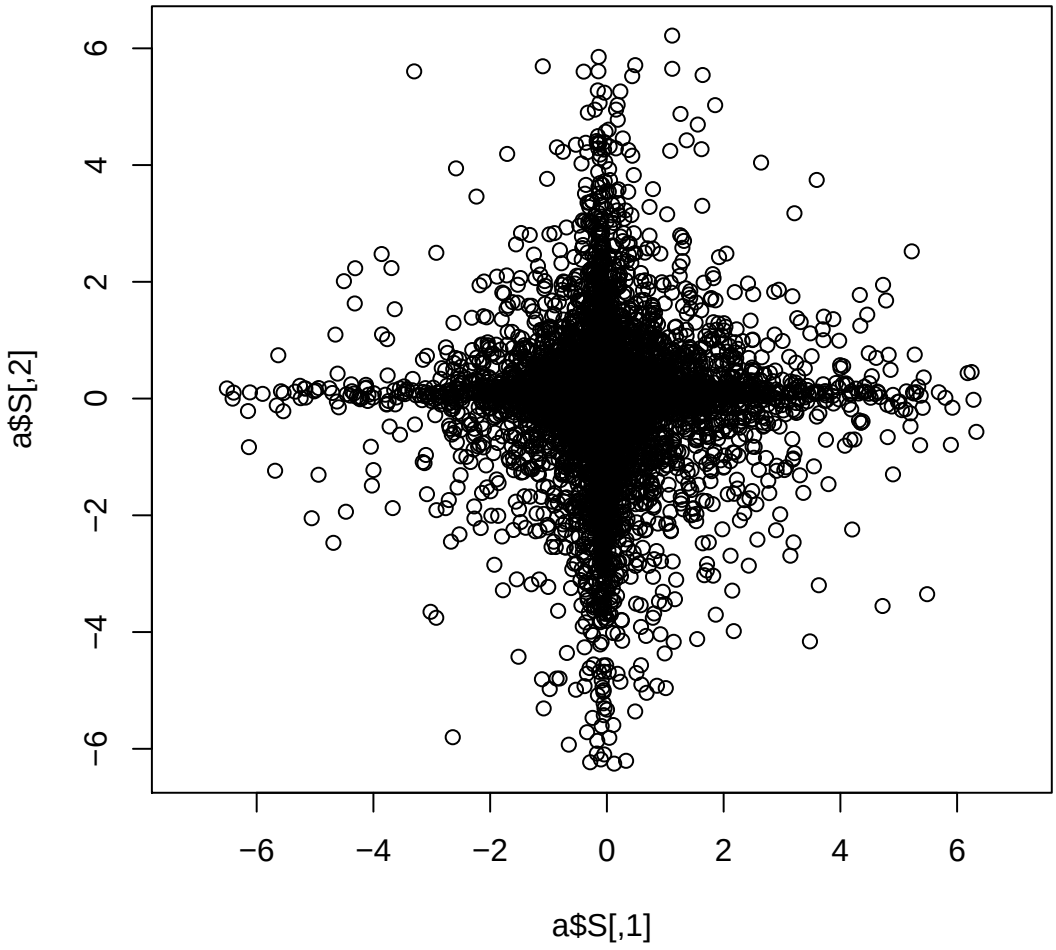


Figure 4.13: Result of `fastICA` on the super-Gaussian mixture.

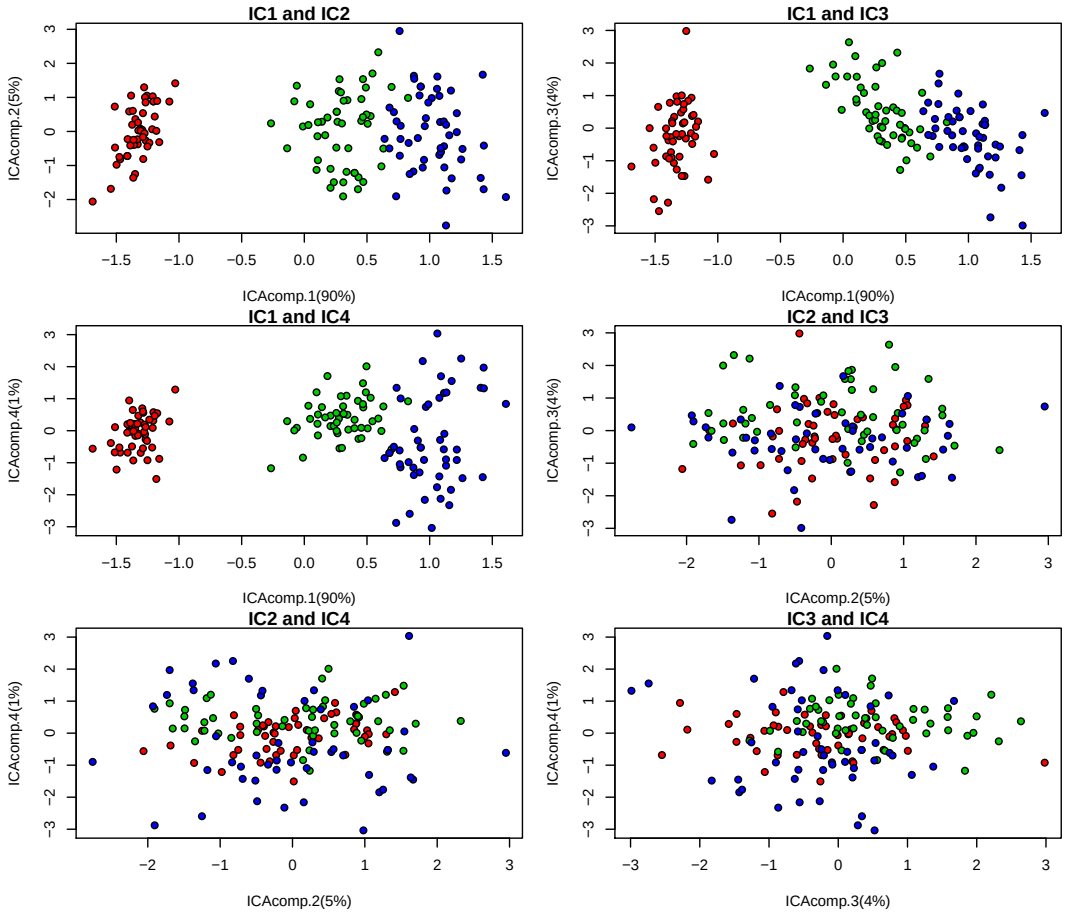


Figure 4.14: ICA applied to Anderson's iris data. The matrix shows scatter plots for pairs of independent components.

4.10 Real World ICA Examples

4.10.1 Iris Data Set

We revisit the iris data set and perform ICA on it. Fig. 4.14 shows scatter plots for pairs of independent components (ICs), more precisely, scatter plots of the projection of the observations to pairs of ICs. We ordered the ICs according to their impact on the observations, which is given by the mixing matrix $\boldsymbol{W}$. We see that the first independent component explains 90% of the variance in the data. Probably IC1 expresses the size of the blossom which is reflected in all four features.

4.10.2 Multiple Tissue Data Set

We apply ICA with 4 components to the multiple tissue microarray data set. Fig. 4.15 shows scatter plots for pairs of independent components, i.e. the projections of the observations to pairs of ICs. IC1 separates the prostate samples (green) and the breast samples (red) from the colon samples

(orange) and the lung samples (blue). Thus, IC1 separates internal organ tissues (colon and lung) from secretory or reproductive organ samples. IC2 separates the prostate samples and the lung samples from the breast samples and the colon samples. IC3 separates the prostate samples and the colon samples from the breast samples and the lung samples. All combinations of the first 3 ICs lead to nice separations except for breast samples and lung samples for which some samples cannot be clearly assigned to one of these two classes. Gene modules may be mutually shared between pairs of tissues samples, which is detected by ICA.

We check which genes are correlated to IC1:

```
"SERPINA7" "LAMB3"    "AR"       "CCNG2"    "KLF5"     "CCL20"
"SLC39A14" "ATP1B1"   "GSTP1"    "LAD1"
```

Interestingly, the androgen receptor (AR) pops up as the 3rd most related gene to IC1. IC1 separates the prostate samples and the breast samples from the other two tissue types. It is known that growth and differentiation of the prostate gland is regulated by androgens. For breast the role of androgens is not as well known, though androgens also regulate breast development. Garay and Park [2012] write

> "However, androgens are known to play a role in normal breast physiology and therefore androgen receptor (AR) signaling is becoming increasingly recognized as an important contributor towards breast carcinogenesis."

Fig. 4.16 shows scatter plots for pairs of independent components for ICA with 8 components. The separation is worse than with 4 components, only the prostate samples (green) are separated by IC1 and IC2. IC3 separates some of the colon samples (orange). The ICs now focus on smaller subgroups. Fig. 4.17 shows scatter plots for pairs of independent components for ICA with 20 components. The result is very similar to the result with 8 components: the prostate samples (green) are separated by IC1 and IC2; IC3 and IC4 separate some of the colon samples (orange). If the data is viewed in more dimensions, then the groups may separate more clearly.

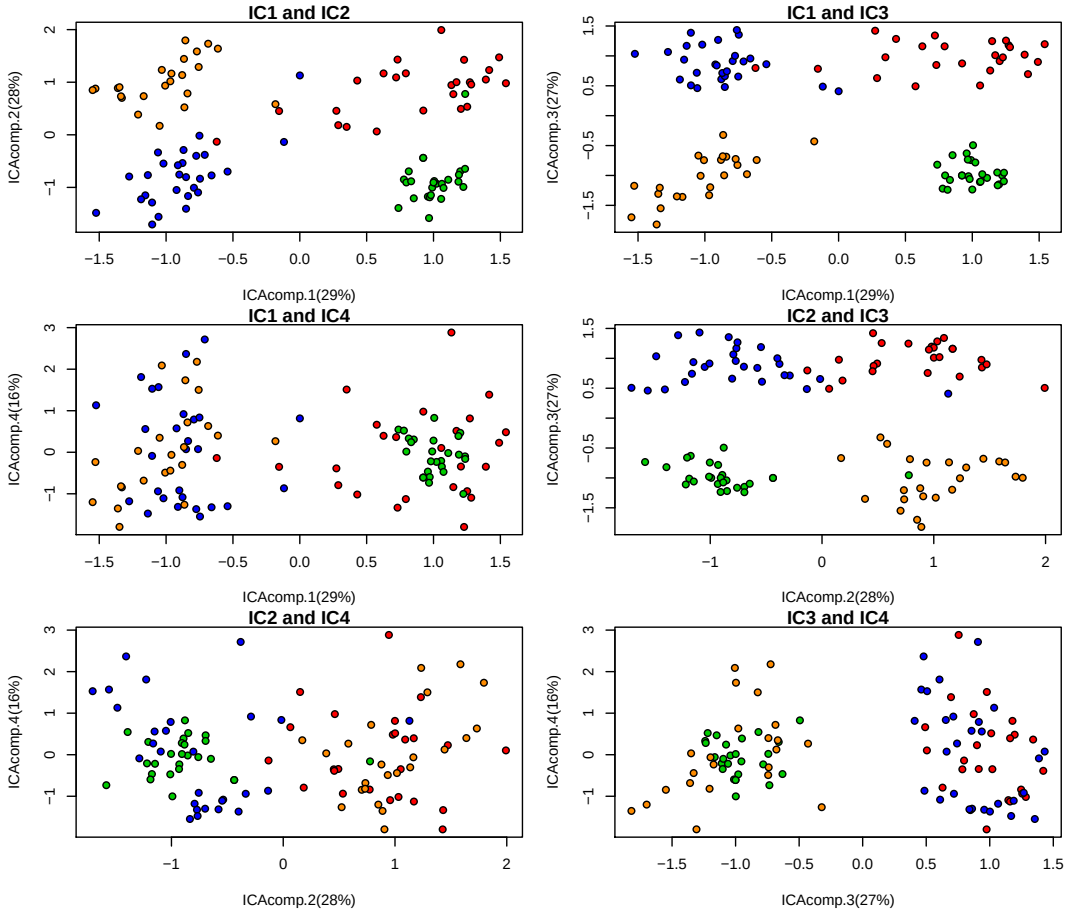


Figure 4.15: fastICA applied to multiple tissue data with 4 components. IC1 separates the prostate samples (green) and the breast samples (red) from the colon samples (orange) and the lung samples (blue). IC2 separates the prostate samples and the lung samples from the breast samples and the colon samples. IC3 separates the prostate samples and the colon samples from the breast samples and the lung samples. All combinations of the first 3 ICs lead to nice separations except for breast samples and lung samples for which some samples cannot be clearly assigned to one of these two classes.

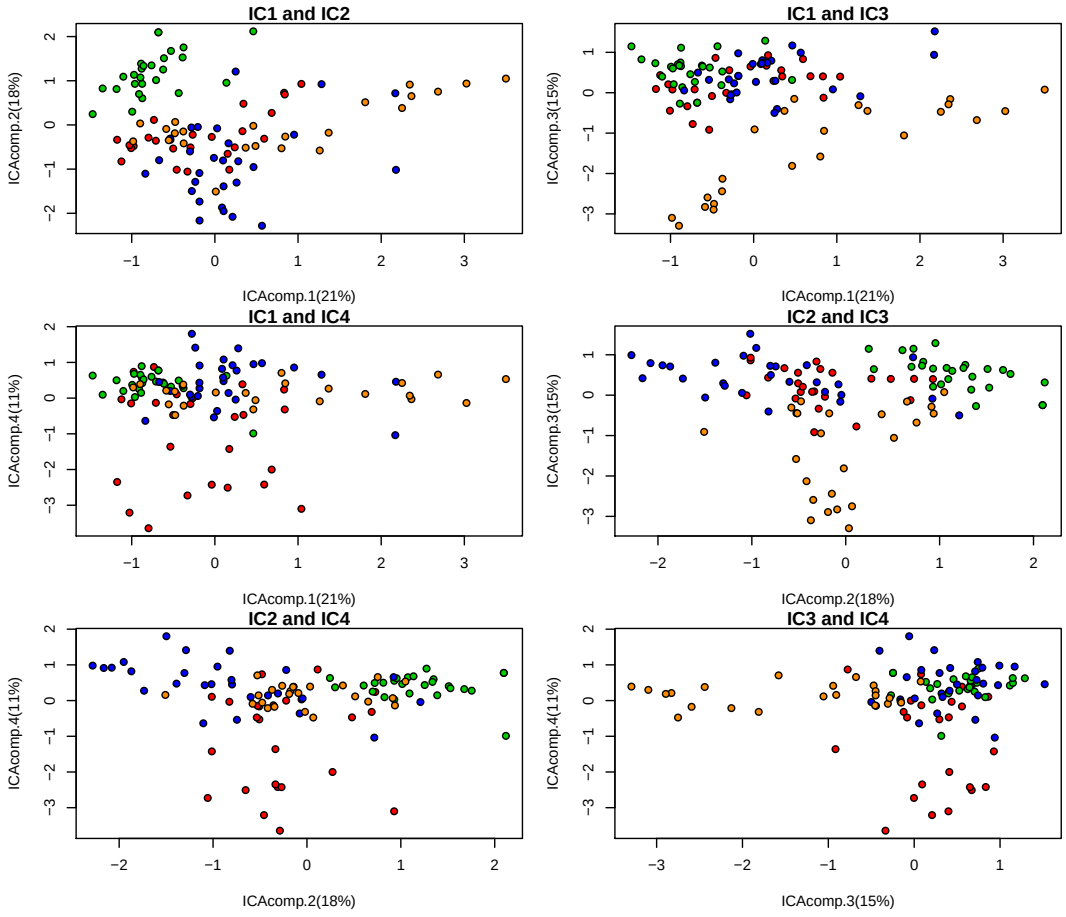


Figure 4.16: fastICA applied to multiple tissue data with 8 components. The separation is worse than with 4 components, only the prostate samples (green) are separated by IC1 and IC2. IC3 separates some of the colon samples (orange). The ICs now focus on smaller subgroups.

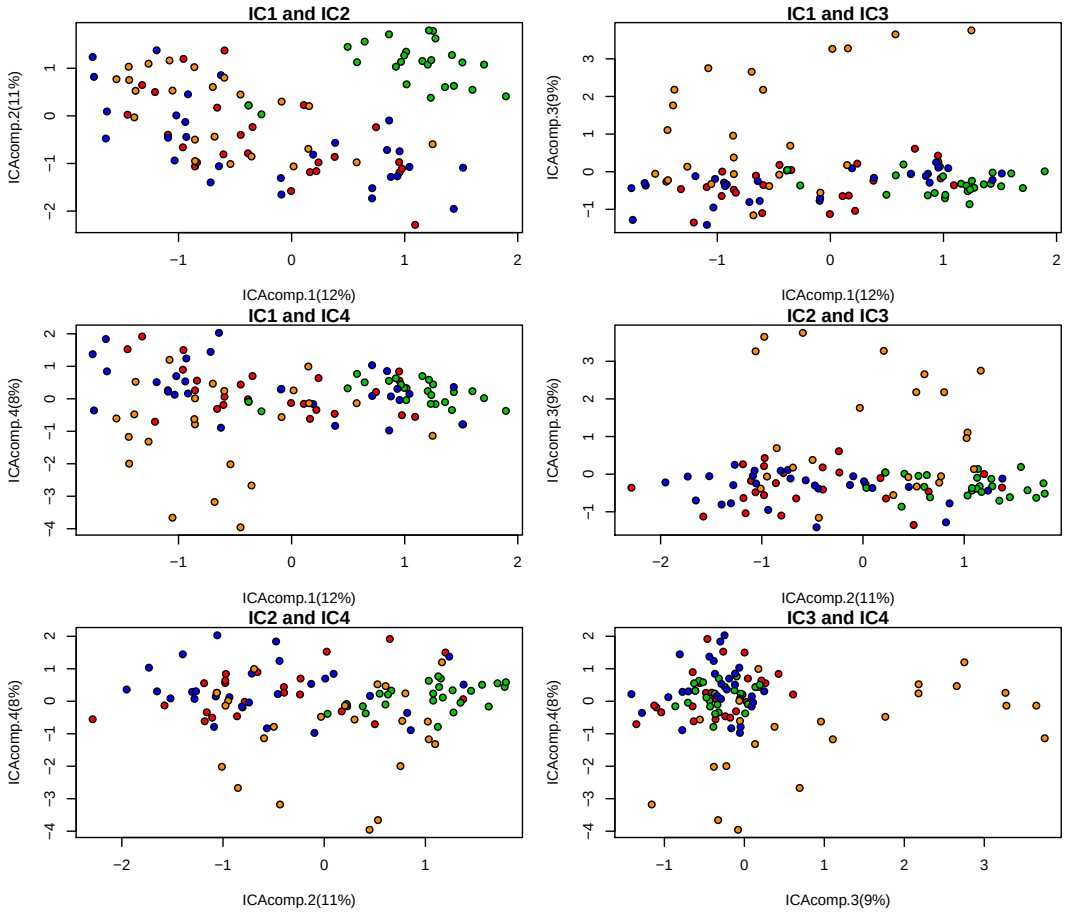


Figure 4.17: fastICA applied to multiple tissue data with 20 components. The result is very similar to the result with 8 components: the prostate samples (green) are separated by IC1 and IC2; IC3 and IC4 separate some of the colon samples (orange). Again, the ICs focus on smaller subgroups.

4.11 Kurtosis Maximization Results in Independent Components

y_1 and y_2 are assumed to be independent from each other and to be super-Gaussian. We show that a linear combination of these signal recovers one of both signals by maximizing the kurtosis.

We assume that y_1 and y_2 are zero centered, symmetric (not skewed), and independent of each other. Therefore we obtain following moments:

$$\begin{aligned}
\mathrm{E}(y_1) &= 0\,, && (4.47)\\
\mathrm{E}(y_2) &= 0\,, && (4.48)\\
\mathrm{E}(y_1^2) &= v_1\,, && (4.49)\\
\mathrm{E}(y_2^2) &= v_2\,, && (4.50)\\
\mathrm{E}(y_1\, y_2) &= 0\,, && (4.51)\\
\mathrm{E}(y_1^3) &= 0\,, && (4.52)\\
\mathrm{E}(y_2^3) &= 0\,, && (4.53)\\
\mathrm{E}(y_1\, y_2^2) &= 0\,, && (4.54)\\
\mathrm{E}(y_2\, y_1^2) &= 0\,, && (4.55)\\
\mathrm{E}(y_1^4) &= m_1\,, && (4.56)\\
\mathrm{E}(y_2^4) &= m_2\,, && (4.57)\\
\mathrm{E}(y_1\, y_2^3) &= 0\,, && (4.58)\\
\mathrm{E}(y_2\, y_1^3) &= 0\,, && (4.59)\\
\mathrm{E}(y_1^2\, y_2^2) &= v_1\, v_2\,. && (4.60)
\end{aligned}$$

We assume that our current reconstruction is a linear combination of these signals:

$$y = a\, y_1 + b\, y_2\,. \tag{4.61}$$

We show: **If y_1 and y_2 are super-Gaussian (have heavy tails) then the maximal kurtosis of y is obtained for $a = 0$ or $b = 0$ that is y is proportional to one y_i.**

The moments of y are:

$$\begin{aligned}
\mathrm{E}(y) &= 0\,, && (4.62)\\
\mathrm{E}(y^2) &= a^2\, v_1 + b^2\, v_2\,, && (4.63)\\
\mathrm{E}(y^3) &= 0\,, && (4.64)\\
\mathrm{E}(y^4) &= a^4\, m_1 + 6\, a^2\, b^2\, v_1\, v_2 + b^4\, m_2\,. && (4.65)
\end{aligned}$$

The kurtosis of y is

$$k = \frac{a^4\, m1 + 6\, a^2\, b^2\, v_1\, v_2 + b^4\, m_2}{\left(a^2\, v_1 + b^2\, v_2\right)^2}\,. \tag{4.66}$$

The derivative of the kurtosis with respect to a is

$$\frac{\partial k}{\partial a} = \frac{4ab^2\left(a^2\left(m_1 - 3v_1^2\right) v_2 - b^2 v_1\left(m_2 - 3v_2^2\right)\right)}{\left(a^2 v_1 + b^2 v_2\right)^3} \tag{4.67}$$

The derivative of the kurtosis with respect to b is

$$\frac{\partial k}{\partial b} = \frac{4a^2 b\left(-a^2\left(m_1 - 3v_1^2\right)v_2 + b^2 v_1\left(m_2 - 3v_2^2\right)\right)}{\left(a^2 v_1 + b^2 v_2\right)^3} \tag{4.68}$$

The derivatives are zero for $a = 0$ or $b = 0$ or

$$a^2 v_2\left(m_1 - 3v_1^2\right) = b^2 v_1\left(m_2 - 3v_2^2\right) . \tag{4.69}$$

The second order derivatives are:

$$\frac{\partial^2 k}{\partial a^2} = \frac{1}{\left(a^2 v_1 + b^2 v_2\right)^4} 4b^2\left(-3a^4 v_1\left(m_1 - 3v_1^2\right)v_2 + \right. \tag{4.70}$$

$$\left. b^4 v_1 v_2\left(-m_2 + 3v_2^2\right) + a^2 b^2\left(5m_2 v_1^2 + 3\left(m_1 - 8v_1^2\right)v_2^2\right)\right) \tag{4.71}$$

$$\frac{\partial^2 k}{\partial a^2} = \frac{1}{\left(a^2 v_1 + b^2 v_2\right)^4} 4a^2\left(a^4 v_1\left(-m_1 + 3v_1^2\right)v_2 + \right. \tag{4.72}$$

$$\left. 3b^4 v_1 v_2\left(-m_2 + 3v_2^2\right) + a^2 b^2\left(3m_2 v_1^2 + \left(5m_1 - 24v_1^2\right)v_2^2\right)\right) \tag{4.73}$$

$$\frac{\partial^2 k}{\partial a\,\partial b} = \frac{1}{\left(a^2 v_1 + b^2 v_2\right)^4} 8ab\left(a^4 v_1\left(m_1 - 3v_1^2\right)v_2 + \right. \tag{4.74}$$

$$\left. b^4 v_1 v_2\left(m_2 - 3v_2^2\right) - 2a^2 b^2\left(m_2 v_1^2 + \left(m_1 - 6v_1^2\right)v_2^2\right)\right) \tag{4.75}$$

$$\frac{\partial^2 k}{\partial a^2}(a, 0) = 0 \tag{4.76}$$

$$\frac{\partial^2 k}{\partial a^2}(0, b) = \frac{4v_1\left(-m_2 + 3v_2^2\right)}{b^2 v_2^3} \tag{4.77}$$

$$\frac{\partial^2 k}{\partial b^2}(0, b) = 0 \tag{4.78}$$

$$\frac{\partial^2 k}{\partial b^2}(a, 0) = \frac{4\left(-m_1 + 3v_1^2\right)v_2}{a^2 v_1^3} \tag{4.79}$$

$$\frac{\partial^2 k}{\partial a\,\partial b}(0, b) = 0 \tag{4.80}$$

$$\frac{\partial^2 k}{\partial a\,\partial b}(a, 0) = 0 \tag{4.81}$$

For the last root of the derivatives we get

$$\hat{a} \;=\; b\sqrt{\frac{v_1(m_2-3v_2^2)}{v_2(m_1-3v_1^2)}} \tag{4.82}$$

and

$$\begin{aligned}\frac{\partial^2 k}{\partial a^2}(\hat{a},b) \;=\; \frac{\partial^2 k}{\partial b^2}(\hat{a},b) \;=\; & \\ \frac{8v_1\left(m_1-3v_1^2\right)^3 v_2^3\left(m_2-3v_2^2\right)}{b^2\left(m_2v_1^2+\left(m_1-6v_1^2\right)v_2^2\right)^3} \;=\; & \\ \frac{8v_1\left(m_1-3v_1^2\right)^3 v_2^3\left(m_2-3v_2^2\right)}{b^2\left(v_1^2\left(m_2-3v_2^2\right)+v_2^2\left(m_1-3v_1^2\right)\right)^3}\,. &\end{aligned} \tag{4.83}$$

This term contains only terms that are larger than zero because the sources are super-Gaussian which means $m_1 > 3v_1^2$ and $m_2 > 3v_2^2$.

The mixed second order derivatives are:

$$\frac{\partial^2 k}{\partial a\,\partial b}(\hat{a},b) \;=\; -\frac{8\left(m_1-3v_1^2\right)^4 v_2^4\left(\frac{v_1\left(m_2-3v_2^2\right)}{\left(m_1-3v_1^2\right)v_2}\right)^{3/2}}{b^2\left(m_2v_1^2+\left(m_1-6v_1^2\right)v_2^2\right)^3} \;= \tag{4.84}$$

$$-\left(\frac{\left(m_1-3v_1^2\right)v_2}{v_1\left(m_2-3v_2^2\right)}\right)^{1/2}\frac{\partial^2 k}{\partial a^2}(\hat{a},b) \tag{4.85}$$

Therefore these derivatives are smaller than zero.

The eigenvalues of the Hessian are proportional to

$$e_1 \;\propto\; 1 \;-\; \left(\frac{\left(m_1-3v_1^2\right)v_2}{v_1\left(m_2-3v_2^2\right)}\right)^{1/2} \tag{4.86}$$

$$e_2 \;\propto\; 1 \;+\; \left(\frac{\left(m_1-3v_1^2\right)v_2}{v_1\left(m_2-3v_2^2\right)}\right)^{1/2}\,. \tag{4.87}$$

It is impossible to make both eigenvalues negative as required for a maximum. Therefore this solution is not a maximum. The maximal solutions are $a = 0$ or $b = 0$ for which the Hessian is negative semidefinite.

In summary, if

$$k_2 \;>\; k_1 \tag{4.88}$$

then $a = 0$ is the maximum of the kurtosis of y. Analogously, if

$$k_1 \;>\; k_2 \tag{4.89}$$

then $b = 0$ is the maximum of the kurtosis of y. That means maximizing the kurtosis gives a unique solution up to permutation and scaling if a linear representation by super-Gaussian distributions is possible.

Chapter 5

Factor Analysis

Factor analysis Jöreskog [1967], Everitt [1984], Neal and Dayan [1997] describes the variability of observations in terms of unobserved latent variables called *factors* and noise. The factors explain correlation between the features or variables while the remaining variance is explained by Gaussian noise. In contrast to the descriptive approach PCA, factor analysis is a generative approach and as such it models both the noise of the observations and their correlation — the latter by factors. Disadvantage of factor analysis is that it has to make assumptions on the distribution like that the factors are Gaussian and the noise is Gaussian. Factor analysis was very successfully applied in bioinformatics, in particular for summarizing microarray data Hochreiter et al. [2006], Talloen et al. [2007, 2010], Hochreiter et al. [2010], Clevert et al. [2011].

5.1 The Factor Analysis Model

We are given the data $\{\boldsymbol{x}\} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ which is assumed to be centered, which can be done by subtracting the mean $\boldsymbol{\mu}$ from the data. The model is

$$\boldsymbol{x} = \boldsymbol{U}\boldsymbol{y} + \boldsymbol{\epsilon}\,, \tag{5.1}$$

where

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \quad \text{and} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Psi})\ . \tag{5.2}$$

The observations $\boldsymbol{x} \in \mathbb{R}^m$, the noise $\boldsymbol{\epsilon} \in \mathbb{R}^m$, the *factors* $\boldsymbol{y} \in \mathbb{R}^l$, the *factor loading matrix* $\boldsymbol{U} \in \mathbb{R}^{m \times l}$, and the noise covariance matrix $\boldsymbol{\Psi}$ is a diagonal matrix from $\mathbb{R}^{m \times m}$.

The data variance is explained through a signal part $\boldsymbol{U}\boldsymbol{y}$ and through a noise part $\boldsymbol{\epsilon}$. The parameters of the model are $\boldsymbol{U}$ and $\boldsymbol{\Psi}$. From the model assumption it follows that

$$\boldsymbol{x} \mid \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{U}\boldsymbol{y}, \boldsymbol{\Psi})\ . \tag{5.3}$$

If $\boldsymbol{y}$ is given, then only the noise $\boldsymbol{\epsilon}$ is a random variable.

The matrix decomposition problem for factor analysis is:

$$\boldsymbol{X} = \boldsymbol{Y}\,\boldsymbol{U}^T + \boldsymbol{\Upsilon}\,. \tag{5.4}$$

Above the model assumptions translate to

$$\frac{1}{n}\,\boldsymbol{Y}^T\boldsymbol{Y} \;=\; \boldsymbol{I} \tag{5.5}$$

$$\boldsymbol{Y}^T\boldsymbol{\Upsilon} \;=\; \boldsymbol{0} \tag{5.6}$$

$$\frac{1}{n}\,\boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon} \;=\; \boldsymbol{\Psi}\,, \tag{5.7}$$

where $\boldsymbol{\Psi}$ is a diagonal matrix. From these equations we obtain:

$$\begin{aligned}\frac{1}{n}\,\boldsymbol{X}^T\boldsymbol{X} \;&=\; \frac{1}{n}\,(\boldsymbol{Y}\,\boldsymbol{U}^T \,+\, \boldsymbol{\Upsilon})^T\,(\boldsymbol{Y}\,\boldsymbol{U}^T \,+\, \boldsymbol{\Upsilon}) \\ &=\; \boldsymbol{U}\,\left(\frac{1}{n}\,\boldsymbol{Y}^T\boldsymbol{Y}\right)\,\boldsymbol{U}^T \,+\, \frac{1}{n}\,\boldsymbol{U}\,\boldsymbol{Y}^T\,\boldsymbol{\Upsilon} \,+\, \frac{1}{n}\,\boldsymbol{\Upsilon}^T\boldsymbol{Y}\,\boldsymbol{U}^T \,+\, \frac{1}{n}\,\boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon} \\ &=\; \boldsymbol{U}\,\boldsymbol{U}^T \,+\, \boldsymbol{\Psi}\,.\end{aligned} \tag{5.8}$$

Therefore, factor analysis is actually a decomposition of the covariance matrix $\boldsymbol{C} = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X}$.

In general the model has fewer factors l than the number m of features of the observations: $m \geq l$. The diagonal form of $\boldsymbol{\Psi}$ is reasonable if the measurements are taken independently and the noise of the components is mutually independent. Therefore, the observations are mutually independent if the factors are known (only the noise is the random component). Therefore, **correlations between observations can only be explained by factors.** This can be seen in the decomposition of the covariance matrix Eq. (5.8): if $\boldsymbol{\Psi}$ is diagonal, then only $\boldsymbol{U}$ can explain covariance between different variables.

We assume that the noise $\boldsymbol{\epsilon}$ and the factors $\boldsymbol{y}$ are independent which need not be true for all applications, e.g. if the noise changes with the signal strength. The parameters $\boldsymbol{U}$ and $\boldsymbol{\Psi}$ can be estimated by maximum likelihood (see Section 5.2). Both parameters explain the variance in the observations $\boldsymbol{x}$ as can be seen at the decomposition of the covariance matrix Eq. (5.8). $\boldsymbol{U}$ explains the dependent part, whereas $\boldsymbol{\Psi}$ explains the independent part of the variance. The factors are very similar to principal components of PCA (see next subsection for a comparison of both methods). Fig. 5.1 depicts the factor analysis model.

Section 5.2 presents the maximum likelihood approach to factor analysis, that is, model selection based on maximum likelihood Jöreskog [1967]. A local maximum of the likelihood is found by the expectation maximization (EM) optimization technique Dempster et al. [1977].

For factor analysis (FA) the factor values are estimated, however, "projection of the data onto the factors" is more complicated than with PCA. We start with a regression setting (see later linear models):

$$\boldsymbol{Y} \;=\; \boldsymbol{X}\,\boldsymbol{A}\,, \tag{5.9}$$

where $\boldsymbol{A}$ is a parameter. The least squares solution (see later linear models) is

$$\hat{\boldsymbol{A}} \;=\; \left(\boldsymbol{X}^T\,\boldsymbol{X}\right)^{-1}\,\boldsymbol{X}^T\boldsymbol{Y}\,. \tag{5.10}$$

The model assumptions (population covariances) and their empirical approximations (sample covariances) are:

$$\boldsymbol{U}\boldsymbol{U}^T \,+\, \boldsymbol{\Psi} \;=\; \mathrm{Var}(\boldsymbol{x}) \;\approx\; \frac{1}{n}\,\boldsymbol{X}^T\,\boldsymbol{X} \tag{5.11}$$

$$\boldsymbol{U} \;=\; \mathrm{Cov}(\boldsymbol{x},\boldsymbol{y}) \;\approx\; \frac{1}{n}\boldsymbol{X}^T\,\boldsymbol{Y}\,. \tag{5.12}$$

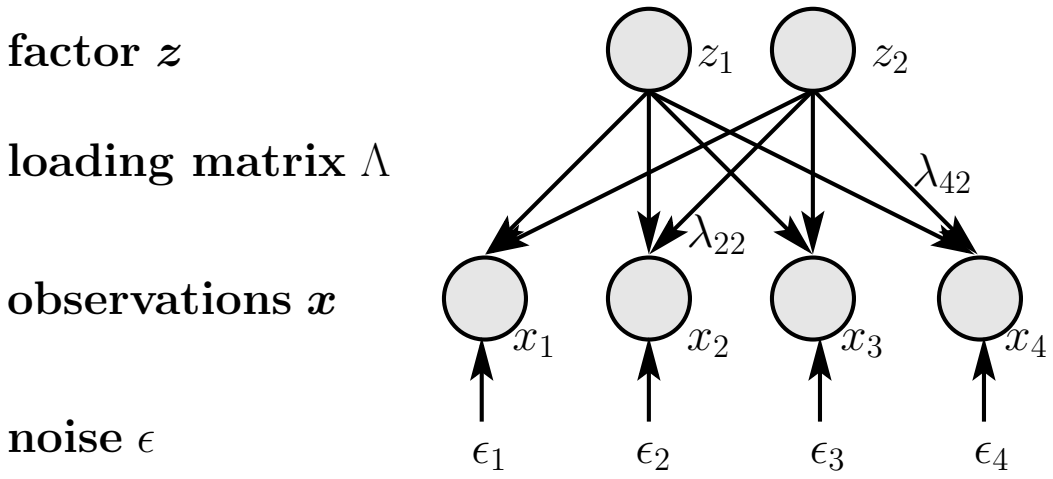


Figure 5.1: The factor analysis model.

The parameter $\boldsymbol{A}$ is estimated using $\mathrm{Var}(\boldsymbol{x})$ and $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{y})$:

$$\hat{\boldsymbol{A}} \;=\; \mathrm{E}\left(\left(\boldsymbol{X}^T\,\boldsymbol{X}\right)^{-1}\right)\,\mathrm{E}\left(\boldsymbol{X}^T\boldsymbol{Y}\right) \tag{5.13}$$

which gives

$$\hat{\boldsymbol{A}} \;=\; \left(\boldsymbol{U}\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\,\boldsymbol{U}\,. \tag{5.14}$$

The projection is therefore

$$\boldsymbol{Y} \;=\; \boldsymbol{X}\;\left(\boldsymbol{U}\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\,\boldsymbol{U}\,, \tag{5.15}$$

or using the matrix inversion lemma

$$\boldsymbol{Y} \;=\; \boldsymbol{X}\,\boldsymbol{\Psi}^{-1}\,\boldsymbol{U}\;\left(\boldsymbol{I} \;+\; \boldsymbol{U}\,\boldsymbol{\Psi}^{-1}\,\boldsymbol{U}^T\right)^{-1}\,. \tag{5.16}$$

The outer product representation for l factors is

$$\boldsymbol{X} \;=\; \sum_{j=1}^{l}\boldsymbol{u}_j\boldsymbol{y}_j^T \;+\; \boldsymbol{\Upsilon}\,, \tag{5.17}$$

where $\boldsymbol{u}_j$ is the j-th column vector of $\boldsymbol{U}$ and $\boldsymbol{y}_j$ is the j-th row vector of $\boldsymbol{Y}$.

We already mentioned that the data variance is explained through signal variance and through noise. The *communality* c_j of an observation variable x_j (the j-th component of $\boldsymbol{x}$) is

$$c_j \;=\; \frac{\mathrm{Var}(x_j) \;-\; \mathrm{Var}(\epsilon_j)}{\mathrm{Var}(x_j)} \;=\; \frac{\sum_{k=1}^{l}\lambda_{jk}^2}{\Psi_{jj} \;+\; \sum_{k=1}^{l}\lambda_{jk}^2} \tag{5.18}$$

which is the proportion in x_j explained by the factors. Here each factor y_t contributes

$$\frac{\lambda_{jt}^2}{\Psi_{jj} \;+\; \sum_{k=1}^{l}\lambda_{jk}^2}\,. \tag{5.19}$$

Like with PCA, the projections onto l factors maximize the variance in the data which can be explained by l factors. However that is only the signal variance but not the noise variance. Furthermore, like with PCA, the factor projections are orthogonal to each other which we know from $\frac{1}{n}\boldsymbol{Y}^T\boldsymbol{Y} = \boldsymbol{I}$. However, the factors are not unique up to orthogonal transformations (rotations). This can be seen by the fact that $\boldsymbol{Y}\boldsymbol{U}^T = \boldsymbol{Y}\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{U}^T = \boldsymbol{Y}'\boldsymbol{U}'^T$ with an orthogonal matrix $\boldsymbol{V}$ and still $\boldsymbol{Y}'^T\boldsymbol{Y}' = \boldsymbol{V}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{V} = \boldsymbol{I}$ holds. Therefore factor analysis is not unique with respect to rotations. Consequently, the projections of the data can be rotated to make the factors more interpretable or to find simpler structures in the data. The following rotations are most common for factor analysis:

- *Varimax rotation*: maximizes the squared loadings of a factor on all the variables; each factor has either large or small loadings of any particular variable; each variable is assigned to a factor.
- *Quartimax rotation*: minimizes the number of factors needed to explain each variable; each factor explains many variables; in most cases not interpretable.
- *Equimax rotation*: compromise between Varimax and Quartimax.

Options for rotations:

```
Orthogonal rotations:
rotate="none"
rotate="varimax"
rotate="quartimax"
rotate="bentlerT"
rotate="geominT"
rotate="bifactor"

Non-orthogonal rotations:
rotate="promax"
rotate="oblimin"
rotate="simplimax"
rotate="bentlerQ"
rotate="geominQ"
rotate="biquartimin"
rotate="cluster"
```

5.2 Maximum Likelihood Factor Analysis

We focus on the maximum likelihood approach to factor analysis which is in most cases Jöreskog [1967] based on the Expectation-Maximization (EM) optimization technique Dempster et al. [1977].

We will now consider the likelihood of the data. Let E denote the expectation of the data (i.e. the factor distribution and the noise distribution is combined), then we obtain for the first two

moments:

$$\begin{aligned} \mathrm{E}(\boldsymbol{x}) \;&=\; \mathrm{E}(\boldsymbol{U}\boldsymbol{y} \;+\; \boldsymbol{\epsilon}) \;=\; \boldsymbol{U}\mathrm{E}(\boldsymbol{y}) \;+\; \mathrm{E}(\boldsymbol{\epsilon}) \;=\; \boldsymbol{0}\,, \\ &\mathrm{E}\left(\boldsymbol{x}\,\boldsymbol{x}^T\right) \;=\; \mathrm{E}\left((\boldsymbol{U}\boldsymbol{y} \;+\; \boldsymbol{\epsilon})(\boldsymbol{U}\boldsymbol{y} \;+\; \boldsymbol{\epsilon})^T\right) \;= \\ &\boldsymbol{U}\mathrm{E}\left(\boldsymbol{y}\,\boldsymbol{y}^T\right)\boldsymbol{U}^T \;+\; \boldsymbol{U}\mathrm{E}\left(\boldsymbol{y}\right)\mathrm{E}\left(\boldsymbol{\epsilon}^T\right) \;+\; \mathrm{E}\left(\boldsymbol{\epsilon}\right)\,\mathrm{E}\left(\boldsymbol{y}^T\right)\,\boldsymbol{U}^T \;+\; \mathrm{E}\left(\boldsymbol{\epsilon}\,\boldsymbol{\epsilon}^T\right) \;= \\ &\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\,. \end{aligned} \tag{5.20}$$

The variance can be computed as

$$\mathrm{var}(\boldsymbol{x}) \;=\; \mathrm{E}\left(\boldsymbol{x}\,\boldsymbol{x}^T\right) \;-\; \left(\mathrm{E}(\boldsymbol{x})\right)^2 \;=\; \boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\,. \tag{5.21}$$

Therefore, the marginal distribution for $\boldsymbol{x}$ is

$$\boldsymbol{x} \;\sim\; \mathcal{N}\left(\boldsymbol{0}\,,\; \boldsymbol{U}\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)\,. \tag{5.22}$$

This means that the observations are Gaussian distributed. This is an assumption of the factor analysis model which can be checked to see whether the model is applicable to a certain problem.

The log-likelihood of the data $\{\boldsymbol{x}\}$ under the model $(\boldsymbol{U}, \boldsymbol{\Psi})$ is

$$\begin{aligned} &\log \prod_{i=1}^{n} (2\pi)^{-m/2} \left|\boldsymbol{U}\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right|^{-1/2} \\ &\exp\left(-\frac{1}{2}\left(\boldsymbol{x}_i^T\left(\boldsymbol{U}\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{x}_i\right)\right)\,, \end{aligned} \tag{5.23}$$

where $|.|$ denotes the absolute value of the determinant of a matrix.

To maximize the likelihood is difficult because no closed form of directly maximizing the likelihood with respect to the parameters is known.

We again apply the EM-algorithm. We introduce a distribution which estimates the hidden states, here the factors.

Using

$$Q_i(\boldsymbol{y}_i) \;=\; p\left(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \boldsymbol{U}, \boldsymbol{\Psi}\right) \tag{5.24}$$

then

$$\begin{aligned} \boldsymbol{y}_i \mid \boldsymbol{x}_i \;&\sim\; \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i}, \boldsymbol{\Sigma}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\right) \\ \boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i} \;&=\; \left(\boldsymbol{x}_i\right)^T\,\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{U} \\ \boldsymbol{\Sigma}_{\boldsymbol{y}_i|\boldsymbol{x}_i} \;&=\; \boldsymbol{I} \;-\; \boldsymbol{U}^T\,\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{U}\,, \end{aligned} \tag{5.25}$$

where we used the fact that

$$\begin{aligned} \boldsymbol{v} \;&\sim\; \mathcal{N}\left(\boldsymbol{\mu}_v, \Sigma_{vv}\right)\;,\; \boldsymbol{u} \;\sim\; \mathcal{N}\left(\boldsymbol{\mu}_u, \Sigma_{uu}\right)\;, \\ \Sigma_{uv} \;&=\; \mathrm{Covar}(\boldsymbol{u}, \boldsymbol{v}) \text{ and } \;\Sigma_{vu} \;=\; \mathrm{Covar}(\boldsymbol{v}, \boldsymbol{u})\,: \\ \boldsymbol{v} \mid \boldsymbol{u} \;&\sim\; \mathcal{N}\left(\boldsymbol{\mu}_v \;+\; \Sigma_{vu}\Sigma_{uu}^{-1}\left(\boldsymbol{u} \;-\; \boldsymbol{\mu}_u\right)\;,\; \Sigma_{vv} \;-\; \Sigma_{vu}\Sigma_{uu}^{-1}\Sigma_{uv}\right) \end{aligned} \tag{5.26}$$

and

$$\mathrm{E}(\boldsymbol{y}\boldsymbol{x}) \;=\; \boldsymbol{U}\,\mathrm{E}(\boldsymbol{y}\,\boldsymbol{y}^T) \;=\; \boldsymbol{U}\;. \tag{5.27}$$

We obtain

$$Q_i(\boldsymbol{y}_i) \;=\; (2\pi)^{-d/2}\left|\boldsymbol{\Sigma}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\right|^{-1/2}\,\exp\left(-\frac{1}{2}\,\left(\boldsymbol{y}_i\;-\;\boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\right)^T\,\boldsymbol{\Sigma}^{-1}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\;-\;\boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\right)\right)\;. \tag{5.28}$$

The EM algorithm for maximum likelihood maximizes in the M-step a lower bound for the log-likelihood:

$$\begin{aligned}\log\left(p(\boldsymbol{x}_i\mid\boldsymbol{U},\boldsymbol{\Psi})\right) \;&=\; \log\left(\int_{\mathbb{R}^p}\frac{Q_i(\boldsymbol{y}_i)\;p(\boldsymbol{x}_i,\boldsymbol{y}_i\mid\boldsymbol{U},\boldsymbol{\Psi})}{Q_i(\boldsymbol{y}_i)}d\boldsymbol{y}_i\right)\;\geq \\ &\int_{\mathbb{R}^p}Q_i(\boldsymbol{y}_i)\;\log\left(\frac{p(\boldsymbol{x}_i,\boldsymbol{y}_i\mid\boldsymbol{U},\boldsymbol{\Psi})}{Q_i(\boldsymbol{y}_i)}\right)d\boldsymbol{y}_i\;.\end{aligned} \tag{5.29}$$

Using the expectation

$$\mathrm{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(f(\boldsymbol{y}_i)\right) \;=\; \int_{\mathbb{R}^p}Q_i(\boldsymbol{y}_i)\;f(\boldsymbol{y}_i)\;d\boldsymbol{y}_i \tag{5.30}$$

and neglecting all terms which are independent of $\boldsymbol{U}$ and $\boldsymbol{\Psi}$, the M-step requires to maximize

$$\begin{aligned}\log\mathcal{L} \;&=\; -\,\frac{m\,n}{2}\log\left(2\pi\right)\;-\frac{m}{2}\log\left|\boldsymbol{\Psi}\right|\;- \\ &\frac{1}{2}\sum_{i=1}^{n}\mathrm{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\left(\boldsymbol{x}_i\;-\;\boldsymbol{U}\boldsymbol{y}_i\right)^T\boldsymbol{\Psi}^{-1}\left(\boldsymbol{x}_i\;-\;\boldsymbol{U}\boldsymbol{y}_i\right)\right)\;.\end{aligned} \tag{5.31}$$

The optimality criteria are

$$\begin{aligned}\frac{1}{n}\nabla_{\boldsymbol{U}}\log\mathcal{L} \;&=\; \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\Psi}^{-1}\,\boldsymbol{U}\,\mathrm{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\,\boldsymbol{y}_i^T\right)\;- \\ &\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\Psi}^{-1}\,\boldsymbol{x}_i\,\mathrm{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\;=\;\boldsymbol{0}\end{aligned} \tag{5.32}$$

and

$$\begin{aligned}\nabla_{\boldsymbol{\Psi}}\log\mathcal{L} \;&=\; -\frac{m}{2}\boldsymbol{\Psi}^{-1}\;+ \\ &\frac{1}{2}\sum_{i=1}^{n}\mathrm{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{\Psi}^{-1}\left(\boldsymbol{x}_i\;-\;\boldsymbol{U}\boldsymbol{y}_i\right)\,\left(\boldsymbol{x}_i\;-\;\boldsymbol{U}\boldsymbol{y}_i\right)^T\boldsymbol{\Psi}^{-1}\right)\;=\;\boldsymbol{0}\;.\end{aligned} \tag{5.33}$$

Solving above equations gives:

$$\boldsymbol{U}^{\mathrm{new}} \;=\; \left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\,\mathrm{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\right)\;\left(\frac{1}{n}\sum_{i=1}^{n}\mathrm{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\,\boldsymbol{y}_i^T\right)\right)^{-1} \tag{5.34}$$

and

$$\boldsymbol{\Psi}^{\text{new}} \;=\; \text{diag}\left(\frac{1}{n}\sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\left(\boldsymbol{x}_i \;-\; \boldsymbol{U}^{\text{new}}\boldsymbol{y}_i\right)\left(\boldsymbol{x}_i \;-\; \boldsymbol{U}^{\text{new}}\boldsymbol{y}_i\right)^T\right)\right) \;= \tag{5.35}$$

$$\text{diag}\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\,\boldsymbol{x}_i^T \;-\; \frac{1}{n}\sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\boldsymbol{x}_i\left(\boldsymbol{U}^{\text{new}}\right)^T \;-\right.$$

$$\left.\frac{1}{n}\sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\boldsymbol{U}^{\text{new}}\boldsymbol{x}_i^T \;+\; \frac{1}{n}\sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\,\boldsymbol{y}_i^T\right)\boldsymbol{U}^{\text{new}}\left(\boldsymbol{U}^{\text{new}}\right)^T\right)\,,$$

where "diag" makes a diagonal matrix from a matrix by setting all non-diagonal elements to zero.

From Eq. (5.34) we obtain

$$\boldsymbol{U}^{\text{new}}\left(\frac{1}{n}\sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\,\boldsymbol{y}_i^T\right)\right) \;=\; \left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\,\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\right)\,. \tag{5.36}$$

and can replace the last term of Eq. (5.35) with the left hand side of above equation which leads to the fact that one term $\frac{1}{n}\sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\boldsymbol{U}^{\text{new}}\boldsymbol{x}_i^T$ cancels in Eq. (5.35). We obtain

$$\boldsymbol{\Psi}^{\text{new}} \;=\; \frac{1}{n}\text{diag}\left(\sum_{i=1}^{n}\boldsymbol{x}_i\,\boldsymbol{x}_i^T \;-\; \sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\boldsymbol{x}_i\left(\boldsymbol{U}^{\text{new}}\right)^T\right)\,. \tag{5.37}$$

This leads to the following EM updates:

E-step: (5.38)

$$\begin{aligned}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right) \;&=\; \boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\\ \text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\,\boldsymbol{y}_i^T\right) \;&=\; \boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\,\boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i}^T \;+\; \boldsymbol{\Sigma}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\end{aligned}$$

M-step: (5.39)

$$\boldsymbol{U}^{\text{new}} \;=\; \left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\,\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\right)\left(\frac{1}{n}\sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\,\boldsymbol{y}_i^T\right)\right)^{-1}$$

$$\boldsymbol{\Psi}^{\text{new}} \;=\; \frac{1}{n}\text{diag}\left(\sum_{i=1}^{n}\boldsymbol{x}_i\,\boldsymbol{x}_i^T \;-\; \sum_{i=1}^{n}\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right)\boldsymbol{x}_i\left(\boldsymbol{U}^{\text{new}}\right)^T\right)\,. \tag{5.40}$$

Speed Ups. To speed up the algorithm especially for $m > l$ the matrix inversion lemma can be used:

$$\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1} \;=\; \boldsymbol{\Psi}^{-1} \;-\; \boldsymbol{\Psi}^{-1}\boldsymbol{U}\left(\boldsymbol{I} \;+\; \boldsymbol{U}^T\boldsymbol{\Psi}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{U}^T\boldsymbol{\Psi}^{-1}\,, \tag{5.41}$$

where $\boldsymbol{\Psi}^{-1}$ can be evaluated very fast because it is a diagonal matrix.

Another speed up is obtained by

$$\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\,\text{E}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\left(\boldsymbol{y}_i\right) \;=\; \left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\left(\boldsymbol{x}_i\right)^T\right)\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{U} \;= \tag{5.42}$$

$$\boldsymbol{C}\,\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{U} \;=\; \boldsymbol{C}\,\left(\boldsymbol{\Psi}^{-1}\boldsymbol{U} \;-\; \boldsymbol{\Psi}^{-1}\boldsymbol{U}\left(\boldsymbol{I} \;+\; \boldsymbol{U}^T\boldsymbol{\Psi}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{U}^T\boldsymbol{\Psi}^{-1}\boldsymbol{U}\right) \;=$$

$$\boldsymbol{C}\,\left(\boldsymbol{U} \;-\; \boldsymbol{U}\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1}\boldsymbol{B}\right)\,,$$

where $\boldsymbol{U} = \boldsymbol{\Psi}^{-1}\boldsymbol{U}$, $\boldsymbol{B} = \boldsymbol{U}^T\boldsymbol{\Psi}^{-1}\boldsymbol{U} = \boldsymbol{U}^T\boldsymbol{U}$, and $\boldsymbol{C}$ is the empirical covariance matrix, which has to be computed only once.

We can also compute

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\Sigma}_{\boldsymbol{y}_i|\boldsymbol{x}_i} \;=\; & \boldsymbol{I} \;-\; \boldsymbol{U}^T\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{U} \;= \\
& \boldsymbol{I} \;-\; \boldsymbol{U}^T\,\boldsymbol{\Psi}^{-1}\boldsymbol{U} \;+\; \boldsymbol{U}^T\,\boldsymbol{\Psi}^{-1}\boldsymbol{U}\left(\boldsymbol{I} \;+\; \boldsymbol{U}^T\boldsymbol{\Psi}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{U}^T\boldsymbol{\Psi}^{-1}\boldsymbol{U} \;= \\
& \boldsymbol{I} \;-\; \boldsymbol{B} \;+\; \boldsymbol{B}\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1}\boldsymbol{B} \;= \\
& \boldsymbol{I} \;-\; \boldsymbol{B} \;+\; \boldsymbol{B}\boldsymbol{B}\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1} \;= \\
& \left(\boldsymbol{I} \;-\; \boldsymbol{B}\right)\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1} \;+\; \boldsymbol{B}\boldsymbol{B}\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1} \;= \\
& \left(\boldsymbol{I} \;+\; \boldsymbol{B} \;-\; \boldsymbol{B} \;-\; \boldsymbol{B}\boldsymbol{B} \;+\; \boldsymbol{B}\boldsymbol{B}\right)\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1} \;= \\
& \left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1}
\end{aligned} \tag{5.43}$$

where we used $\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1}\;\boldsymbol{B} = \boldsymbol{B}\;\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1}$ [Lütkepohl, 1996, Section 3.5.2 (6) (a)] and

$$\begin{aligned}
& \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i}\,\boldsymbol{\mu}_{\boldsymbol{y}_i|\boldsymbol{x}_i}^T \;= \\
& \boldsymbol{U}^T\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\left(\boldsymbol{x}_i\right)^T\right)\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{U} \;= \\
& \boldsymbol{U}^T\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{C}\;\left(\boldsymbol{U}\,\boldsymbol{U}^T \;+\; \boldsymbol{\Psi}\right)^{-1}\boldsymbol{U} \;= \\
& \left(\boldsymbol{U} \;-\; \boldsymbol{U}\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1}\boldsymbol{B}\right)^T\boldsymbol{C}\left(\boldsymbol{U} \;-\; \boldsymbol{U}\left(\boldsymbol{I} \;+\; \boldsymbol{B}\right)^{-1}\boldsymbol{B}\right)\,.
\end{aligned} \tag{5.44}$$

Using these equations the E-step and the M-step can be unified and all sums $\sum_{i=1}^{n}$ are removed and the matrix $\boldsymbol{C}$ can be computed once at the beginning of the iterative procedure.

MAP factor analysis. This algorithm can be generalized to a maximum a posteriori method with posterior $p(\boldsymbol{U}, \boldsymbol{\Psi} \mid \{\boldsymbol{x}\})$ which is proportional to the product between the likelihood $p(\{\boldsymbol{x}\} \mid \boldsymbol{U}, \boldsymbol{\Psi})$ and the prior $p(\boldsymbol{U})$:

$$p(\boldsymbol{U}, \boldsymbol{\Psi} \mid \{\boldsymbol{x}\}) \;\propto\; p(\{\boldsymbol{x}\} \mid \boldsymbol{U}, \boldsymbol{\Psi})\,p(\boldsymbol{U})\,, \tag{5.45}$$

therefore up to a constant independent of the parameters the log-posterior is

$$\log\left(p(\boldsymbol{U}, \boldsymbol{\Psi} \mid \{\boldsymbol{x}\})\right) \;=\; \log\left(p(\{\boldsymbol{x}\} \mid \boldsymbol{U}, \boldsymbol{\Psi})\right) \;+\; \log\left(p(\boldsymbol{U})\right)\,. \tag{5.46}$$

An example for the prior on λ_j is a rectified Gaussian $\mathcal{N}_{\text{rect}}\left(\mu_\Lambda, \sigma_\Lambda\right)$ in order to allow only positive factor loading values which assume that the factors are only additive:

$$y_j \;\sim\; \mathcal{N}\left(\mu_\Lambda, \sigma_\Lambda\right)\lambda_j \;=\; \max\{y_j, 0\}\,. \tag{5.47}$$

MAP factors. The E-step gives also the most probable values for the factors $\boldsymbol{y}$. This can be important for analyzing data and extracting hidden causes.

5.3 Factor Analysis vs. PCA and ICA

We consider factor analysis and compare it to PCA:

factor analysis	principal component analysis
causes of the data	geometrical abstractions
explain common variances	explain all variance
variance shared	first l with max. variance
scale invariant	not scale invariant
additive noise (variance lost)	no noise
solution not unique	solution unique
model assumptions	no assumptions
solution depends on l	first l unique
projection uses noise	no noise
no ranking	ranked by eigenvalues

We now compare factor analysis to ICA:

factor analysis	independent component analysis
additive noise	no noise
solution not unique	unique up to scale and permutation
assumption: Gauss	assumption: super-Gauss
projection averaged over noise	no noise
solution depends on l	does not depend on l

5.4 Artificial Factor Analysis Examples

We compare factor analysis and ICA on a 50-dimensional data set with linearly mixed super-Gaussians.

First we generate a 50-dimensional data set with super-Gaussians. Fig. 5.2 shows the first two components of the original data of the 50-dimensional super-Gaussian.

Next the 50-dimensional super-Gaussian is mixed. Fig. 5.3 shows the first two components of the linear mixing of 50 super-Gaussians.

First we apply factor analysis to this mixture of 50 super-Gaussians. Fig. 5.4 shows the first two components of the results of ICA applied to the mixture of 50-dimensional super-Gaussian.

Next we apply factor analysis to this 50-dimensional data set. Fig. 5.5 shows the first two components of the results of factor analysis applied to the mixture of 50 super-Gaussians. The demixing does not work as well as with ICA. This was expected because factor analysis assumes normally distributed factors while ICA assumes super-Gaussian components. Thus, the task is suited for ICA but not for factor analysis. Factor analysis with subsequent rotations that maximize some ICA contrast functions would lead to better results concerning separation of the sources.

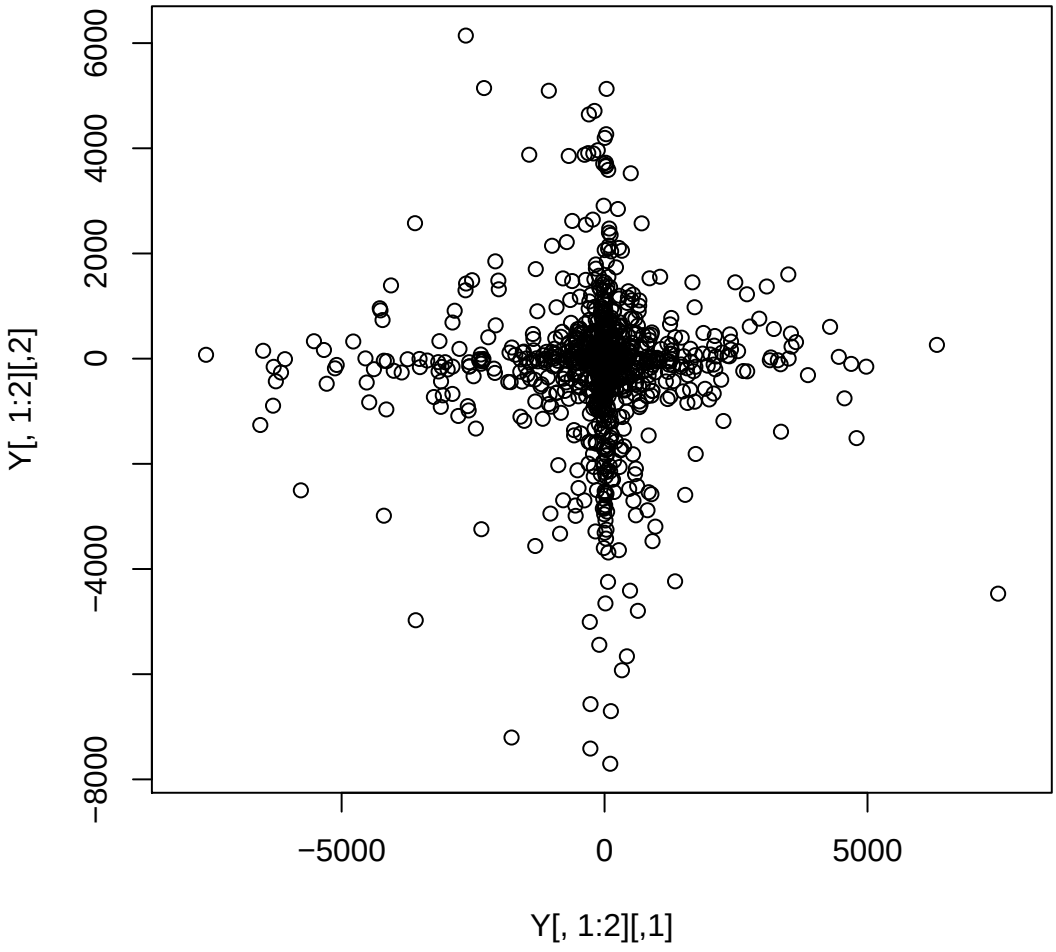


Figure 5.2: The first two components of the original data of the 50-dimensional super-Gaussian.

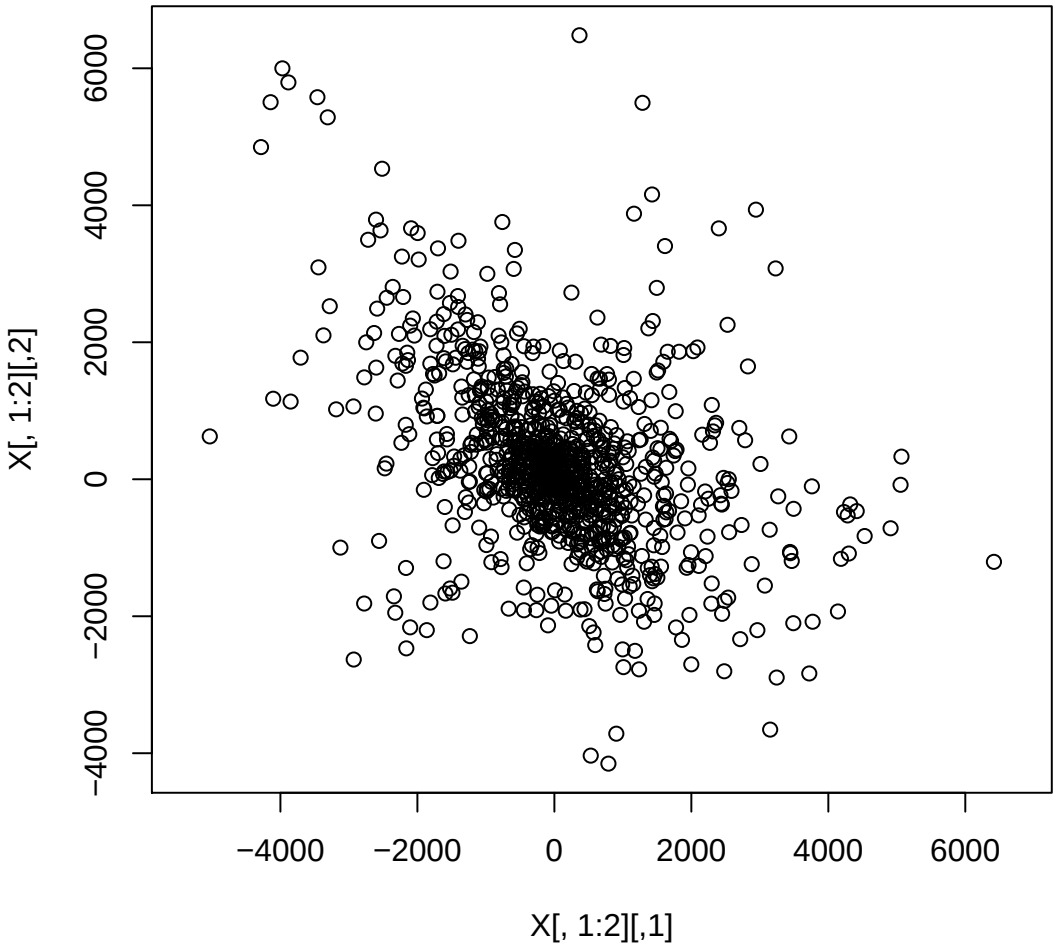


Figure 5.3: First two components of the linear mixing of the 50-dimensional super-Gaussian.

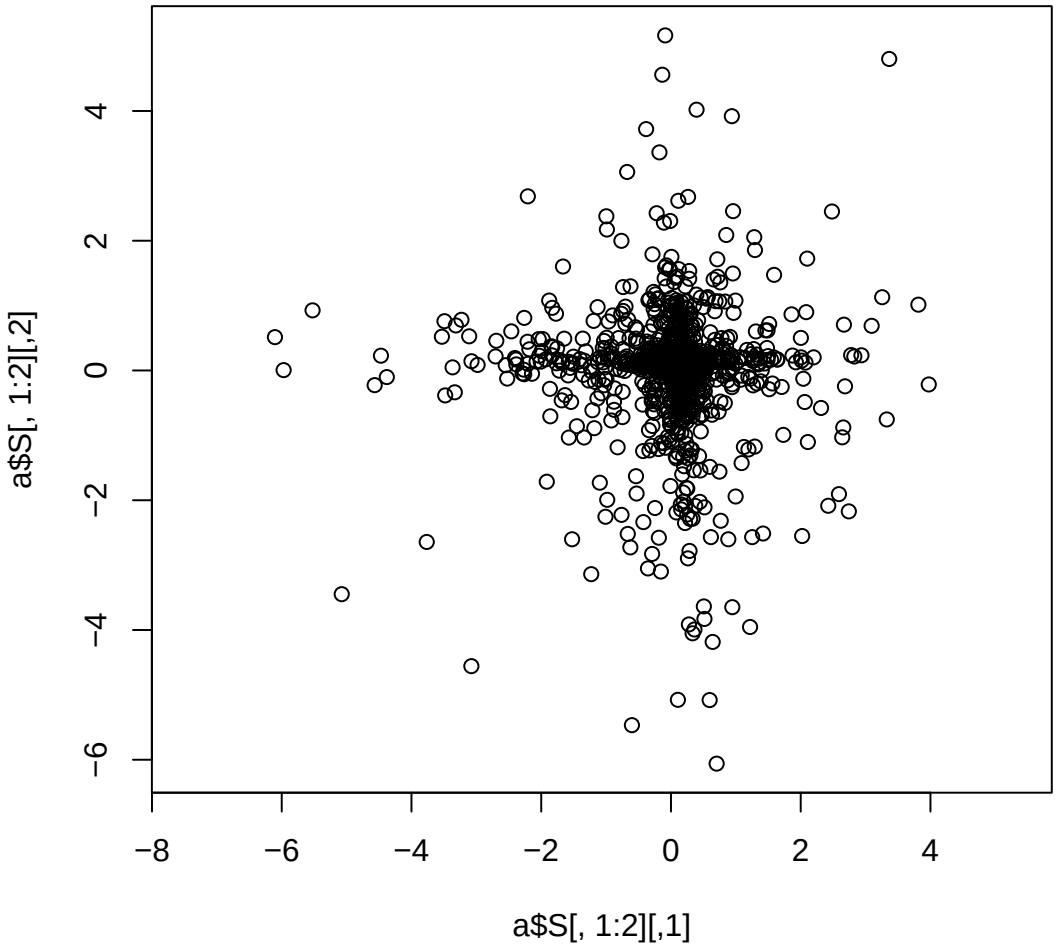


Figure 5.4: First two components of the results of `fastICA` applied to the mixture of the 50-dimensional super-Gaussian.

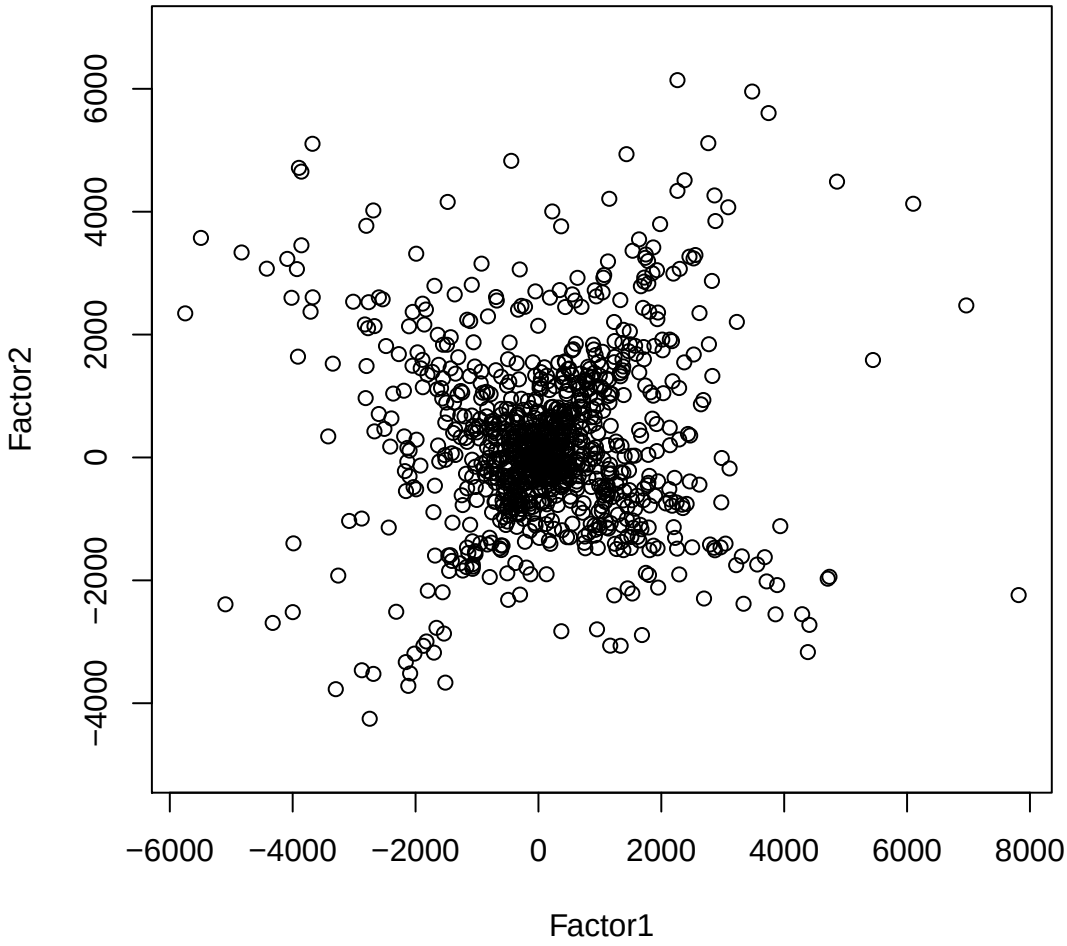


Figure 5.5: The first two components of the results of factor analysis applied to the mixture of 50 super-Gaussians.

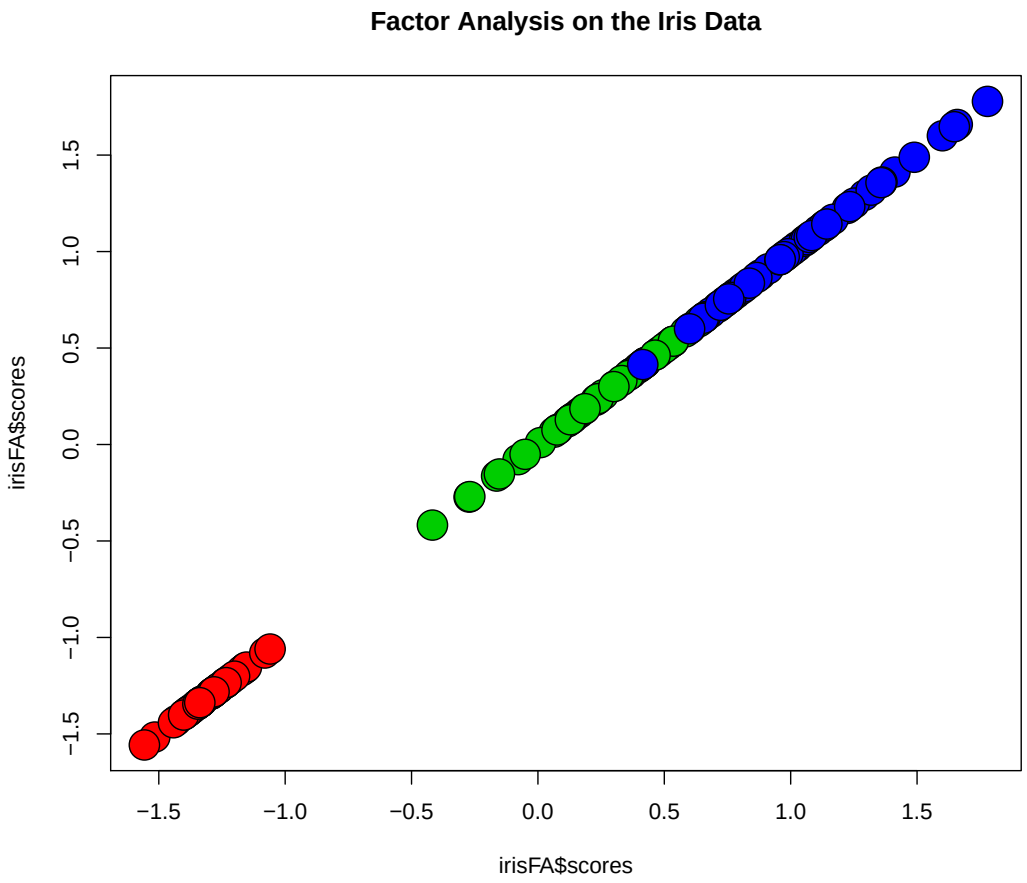


Figure 5.6: Factor analysis applied to Anderson's iris data. Scatter plot of the factor against itself.

5.5 Real World Factor Analysis Examples

5.5.1 Iris Data Set

We revisit the iris data set and perform factor analysis on this data. We computed one factor by factor analysis. Fig. 5.6 shows the factor extracted by factor analysis. The species can be quite well separated except for one flower.

5.5.2 Multiple Tissue Data Set

We apply factor analysis to the multiple tissue microarray data set. Since for this data set $m > n$, we first select the n features with largest variance, in order to ensure a full rank covariance matrix. Then we call factor analysis with 4 components and then compute the scoring.

Fig. 5.7 shows scatter plots for pairs of factors from the factor analysis without any rotation. Factor 1 (FA1) separates prostate samples (green) from the rest. FA2 separates breast samples

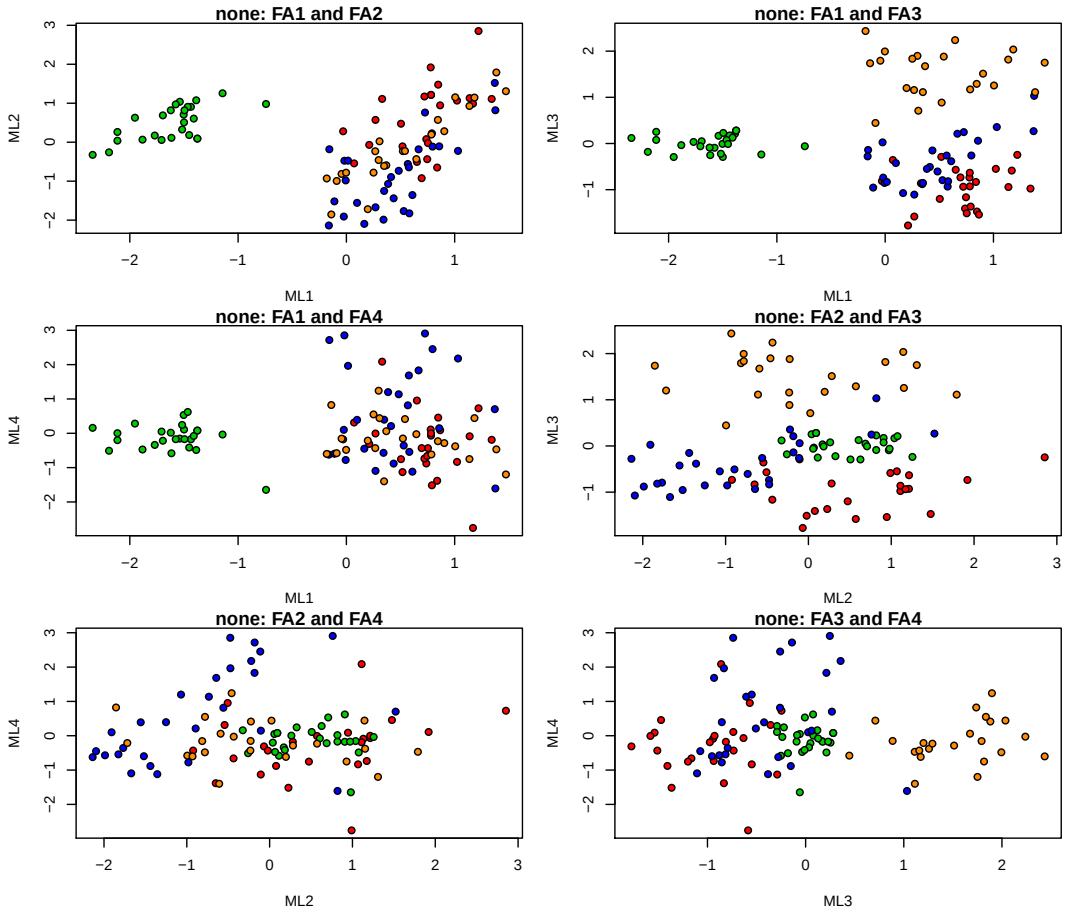


Figure 5.7: Factor analysis applied to multiple tissue data with 4 components and no rotation. Factor 1 (FA1=ML1) separates prostate samples (green) from the rest. FA2 (ML2) separates breast samples (red) from the lung samples (blue), however this separation is not very good. FA3 (ML3) separates the colon samples (orange) from the rest. FA4 (ML4) separates part of the lung samples (blue) from the rest.

(red) from the lung samples (blue), however this separation is not very good. FA3 separates the colon samples (orange) from the rest. FA4 separates part of the lung samples (blue) from the rest. Fig. 5.8 shows scatter plots for pairs of factors from the factor analysis with rotation “varimax”. Fig. 5.9 shows scatter plots for pairs of factors from the factor analysis with rotation “quartimax”. The results of both rotations are very similar to each other. The separation is slightly worse than without rotations.

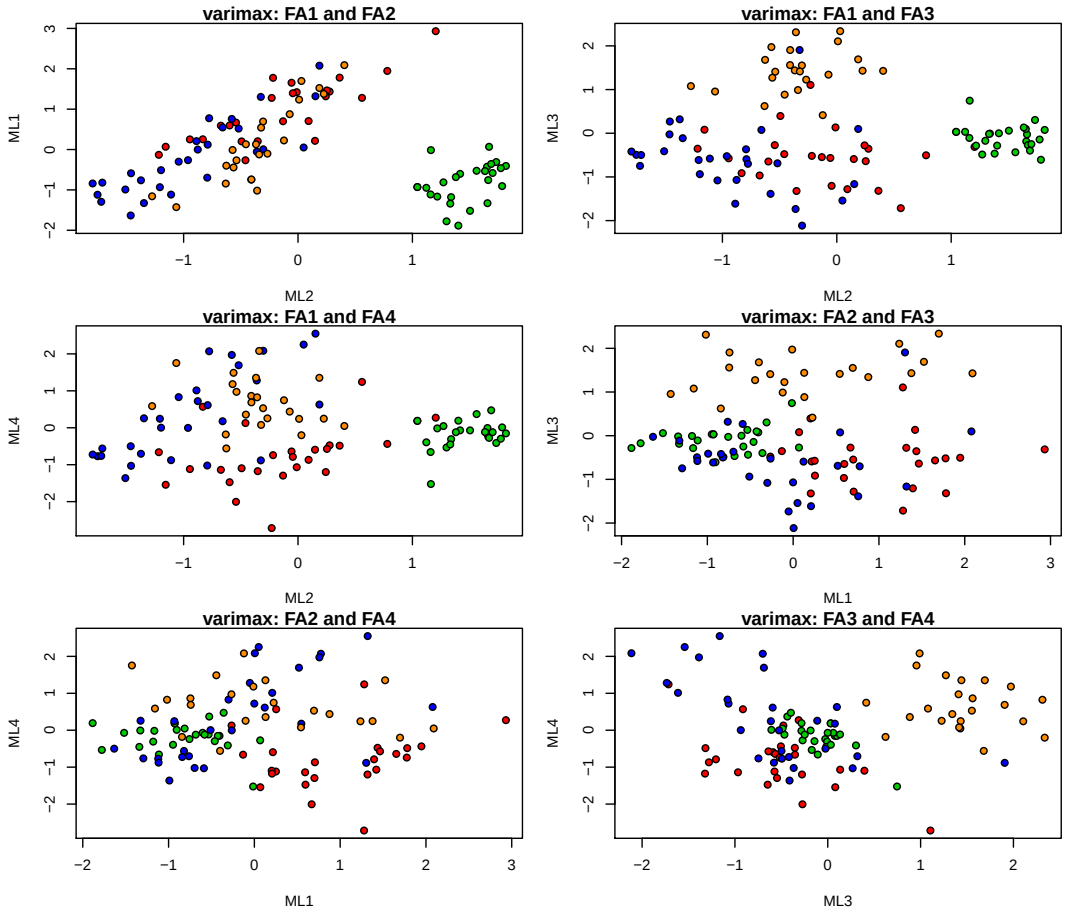


Figure 5.8: Factor analysis applied to multiple tissue data with 4 components and "varimax" rotation. Factor 1 (FA1=ML2) separates prostate samples (green) from the rest. FA2 (ML1) separates breast samples (red) from the lung samples (blue), however this separation is not very good. FA3 (ML3) separates the colon samples (orange) from the rest. FA4 (ML4) separates part of the lung samples (blue) from the rest.

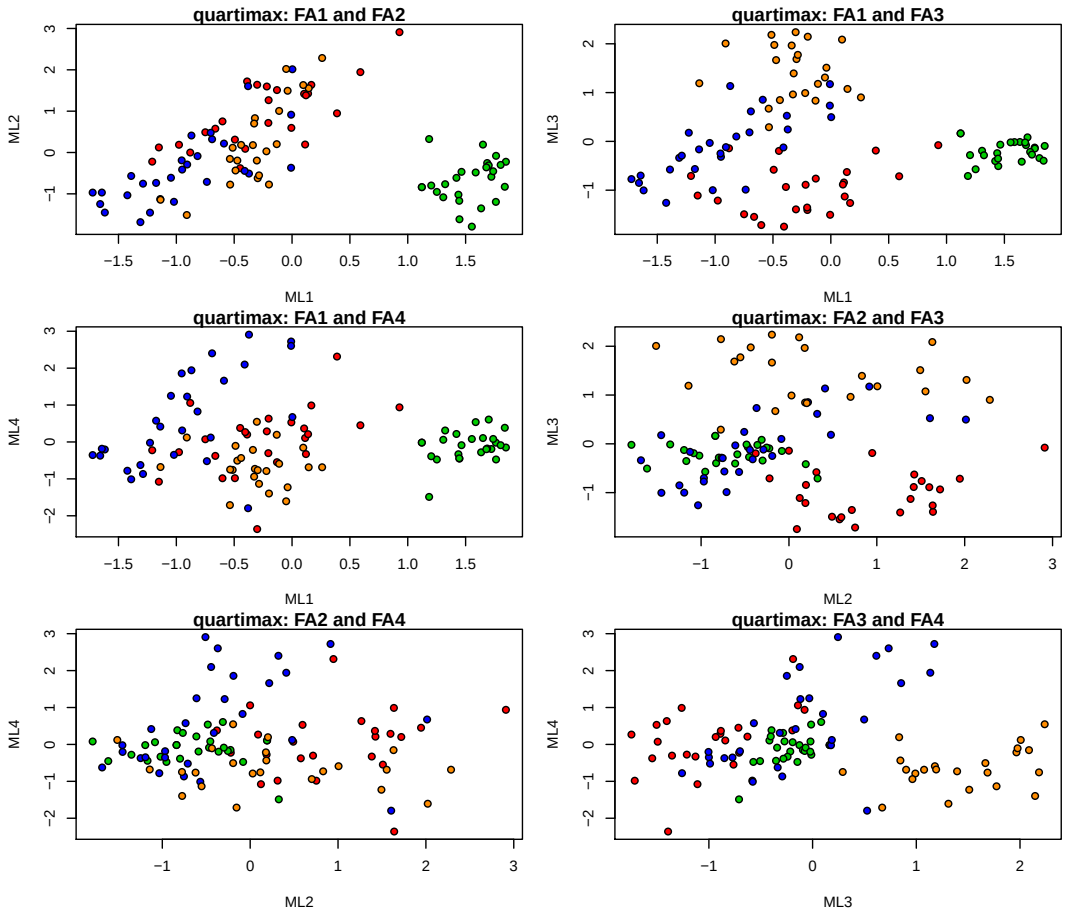


Figure 5.9: Factor analysis applied to multiple tissue data with 4 components and "quartimax" rotation. Factor 1 (FA1) separates prostate samples (green) from the rest. FA2 separates breast samples (red) from the lung samples (blue), however this separation is not very good. FA3 separates the colon samples (orange) from the rest. FA4 separates part of the lung samples (blue) from the rest.

Chapter 6

Scaling and Projection Methods

We consider methods that project the data to a low-dimensional space. This is also called "scaling". The goal is either to visualize the data or to represent the data in a low-dimensional space for further processing. The low-dimensional space allows to perform model selection using low-complex model classes. In a low-dimensional space only the main structures of the data can be captured, therefore, it is assumed that noise and outliers are not represented.

6.1 Projection Pursuit

Projection pursuit Friedman and Tukey [1974], Friedman and Stuetzle [1981], Huber [1985], Friedman [1987], Jones [1987], Jones and Sibson [1987], Zhao and Atkeson [1996], Intrator [1993] attempts to find "interesting" projections of the data in order to visualize or cluster the data. "Interesting" is defined as the least Gaussian distribution. The central question is how to define non-Gaussianity. If the covariance of a zero mean variable $\boldsymbol{y}$ is fixed, then a Gaussian distribution maximizes the entropy $H(\boldsymbol{y})$. Then for $t = \boldsymbol{w}^T\boldsymbol{x}$ the vector $\boldsymbol{w}$ must be found which minimizes $H(t)$ if t is normalized to zero mean and unit variance. However, the density of $t = \boldsymbol{w}^T\boldsymbol{x}$ is difficult to estimate. The entropy is minimized by finding unimodal super-Gaussians or by finding multimodal distributions.

Other more practical measures of non-Gaussianity have been given in paragraph "Non-Gaussianity" in the Subsection 4.2 of ICA. Besides the kurtosis, different contrast functions are discussed which measure non-Gaussianity. Actually, ICA provides examples for projection pursuit.

6.2 Multidimensional Scaling

6.2.1 The Method

Multidimensional Scaling (MDS) Torgerson [1958], Gower [1966] aims at representing data points $\boldsymbol{x}$ by $\boldsymbol{y}$ in a lower dimensional space so that the distances between the $\boldsymbol{y}$'s correspond to the distances (dissimilarities) between the $\boldsymbol{x}$'s.

We define

$$\boldsymbol{y}_i = f(\boldsymbol{x}_i; \boldsymbol{w}) \tag{6.1}$$
$$\delta_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \tag{6.2}$$
$$d_{ij} = \|\boldsymbol{y}_i - \boldsymbol{y}_j\| \ . \tag{6.3}$$

The goal is to define a measure which measures the difference between δ and d. In most cases a weighted sum of the absolute differences between δ_{ij} and d_{ij} or their squared values are used. Measures, which have previously been used, are

$$R_1(d, \delta) = \frac{\sum_{i<j} (d_{ij} - \delta_{ij})^2}{\sum_{i<j} \delta_{ij}^2} \propto \sum_{i<j} (d_{ij} - \delta_{ij})^2 \ , \tag{6.4}$$

$$R_2(d, \delta) = \sum_{i<j} \left(\frac{d_{ij} - \delta_{ij}}{\delta_{ij}} \right)^2 \ , \tag{6.5}$$

$$R_3(d, \delta) = \frac{1}{\sum_{i<j} \delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}} \propto \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}} \ , \tag{6.6}$$

where "$\propto$" means that factors which are constant in the parameters $\boldsymbol{w}$ are removed.

The measure R_1 is called "Kruskal's measure" and is basically the mean squared error and penalizes large errors even if the δ_{ij} are large. The measure R_2 measures the fractional errors (relative errors) but small δ_{ij} may increase the relative error. R_3 is called "Sammon mapping" and is a compromise of R_1 and R_2.

The derivatives, which are used in gradient-based methods, are

$$\frac{\partial}{\partial \boldsymbol{y}_k} R_1(d, \delta) = \frac{2}{\sum_{i<j} \delta_{ij}^2} \sum_{j \neq k} (d_{kj} - \delta_{kj}) \frac{\boldsymbol{y}_k - \boldsymbol{y}_j}{d_{kj}} \tag{6.7}$$

$$\frac{\partial}{\partial \boldsymbol{y}_k} R_2(d, \delta) = 2 \sum_{j \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}^2} \frac{\boldsymbol{y}_k - \boldsymbol{y}_j}{d_{kj}} \tag{6.8}$$

$$\frac{\partial}{\partial \boldsymbol{y}_k} R_3(d, \delta) = \frac{2}{\sum_{i<j} \delta_{ij}} \sum_{j \neq k} \frac{d_{kj} - \delta_{kj}}{\delta_{kj}} \frac{\boldsymbol{y}_k - \boldsymbol{y}_j}{d_{kj}} \ . \tag{6.9}$$

If the measures R are viewed as potential functions, then these derivatives with respect to points $\boldsymbol{y}_k$ can be considered as forces on $\boldsymbol{y}_k$. Also other supervised and unsupervised methods can be derived from potential functions and forces Bashkirov et al. [1964], Perrone and Cooper [1995], Hochreiter and Mozer [2001b,a], Hochreiter et al. [2003], Hochreiter and Obermayer [2005].

6.2.2 Examples

Fig. 6.1 shows an example for multidimensional scaling from Duda et al. [2001].

We perform classical *metric multidimensional scaling* of a data matrix, which is known as *principal coordinates analysis* Torgerson [1958], Gower [1966]. Fig. 6.2 shows the result of classical multidimensional scaling applied to multiple tissue data and down-projected to 2 dimensions.

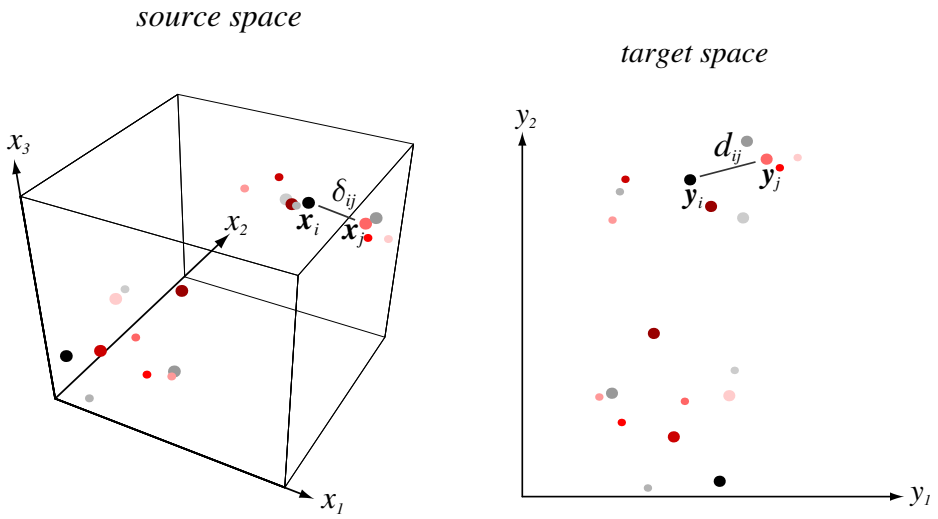


Figure 6.1: Example for multidimensional scaling. Points $\boldsymbol{x}$ from a 3-dimensional space (left) are mapped by multidimensional scaling to a 2-dimensional space (right). From Duda et al. [2001], Copyright © 2001 John Wiley & Sons, Inc.

The 101 features with largest variance are selected. Fig. 6.3 shows the same result but for the 13 features with largest variance. In both cases the results are almost identical to PCA with the first two components.

The R function `isoMDS()` implements Kruskal's non-metric multidimensional scaling, which is the measure R_1 from above. We produce MDS with Kruskal's measure for the 101 features with the largest variance (see Fig. 6.4) and for the 13 features with the largest variance (see Fig. 6.5). Kruskal's is better for separating breast samples (red) from the lung samples (blue). However, the separation of the colon samples (orange) gets worse. Also the prostate samples (green) are less clearly separated.

The next MDS procedure is Sammon's non-linear mapping, which was the measure R_3 from above. The results of Sammon's mapping for the 101 features with the largest variance are shown in Fig. 6.6 and for the 13 features with the largest variance in Fig. 6.7. The separation of the classes is worse than with metric or Kruskal's measure. Some data points get separated from their corresponding classes. For 13 features the clusters are more spread out and there is a relatively large distance between the points. The later makes it more difficult to identify cluster.

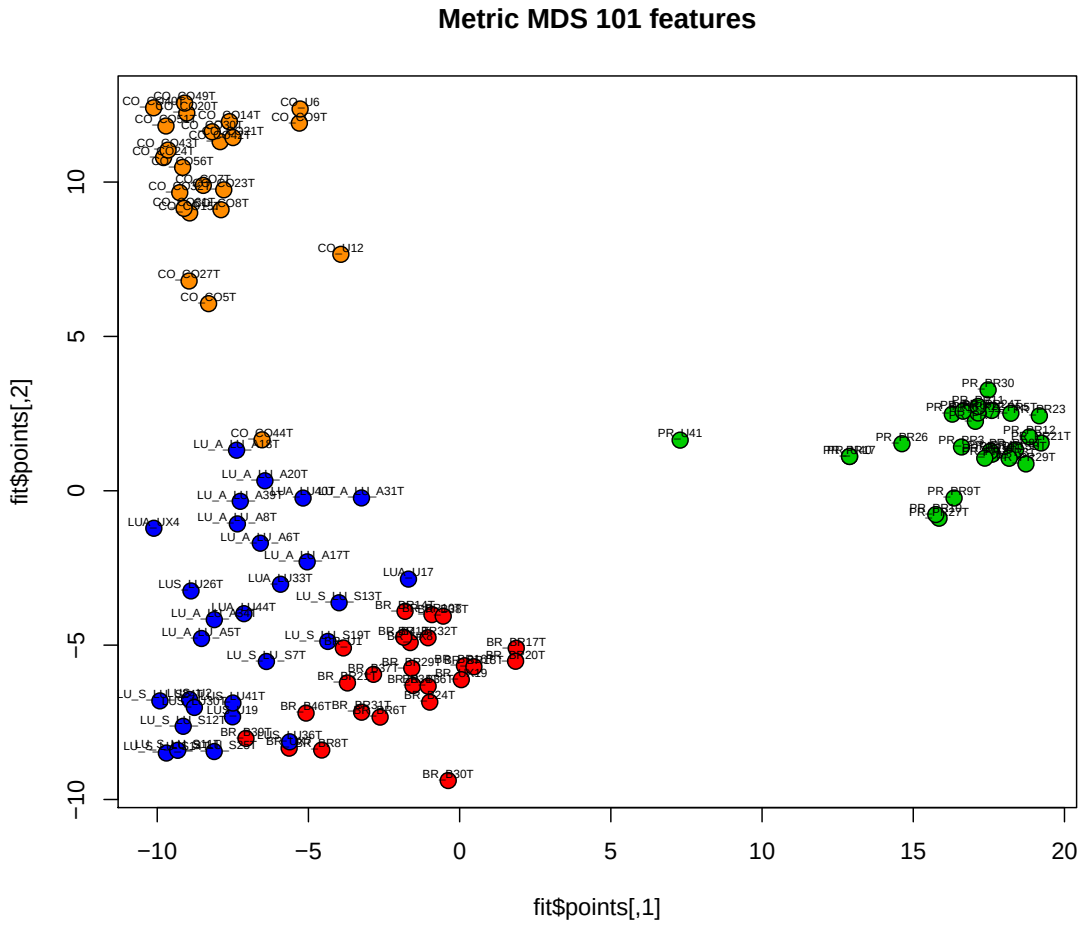


Figure 6.2: Classical metric multidimensional scaling applied to multiple tissue data and down-projected to 2 dimensions. The 101 features with largest variance are selected.

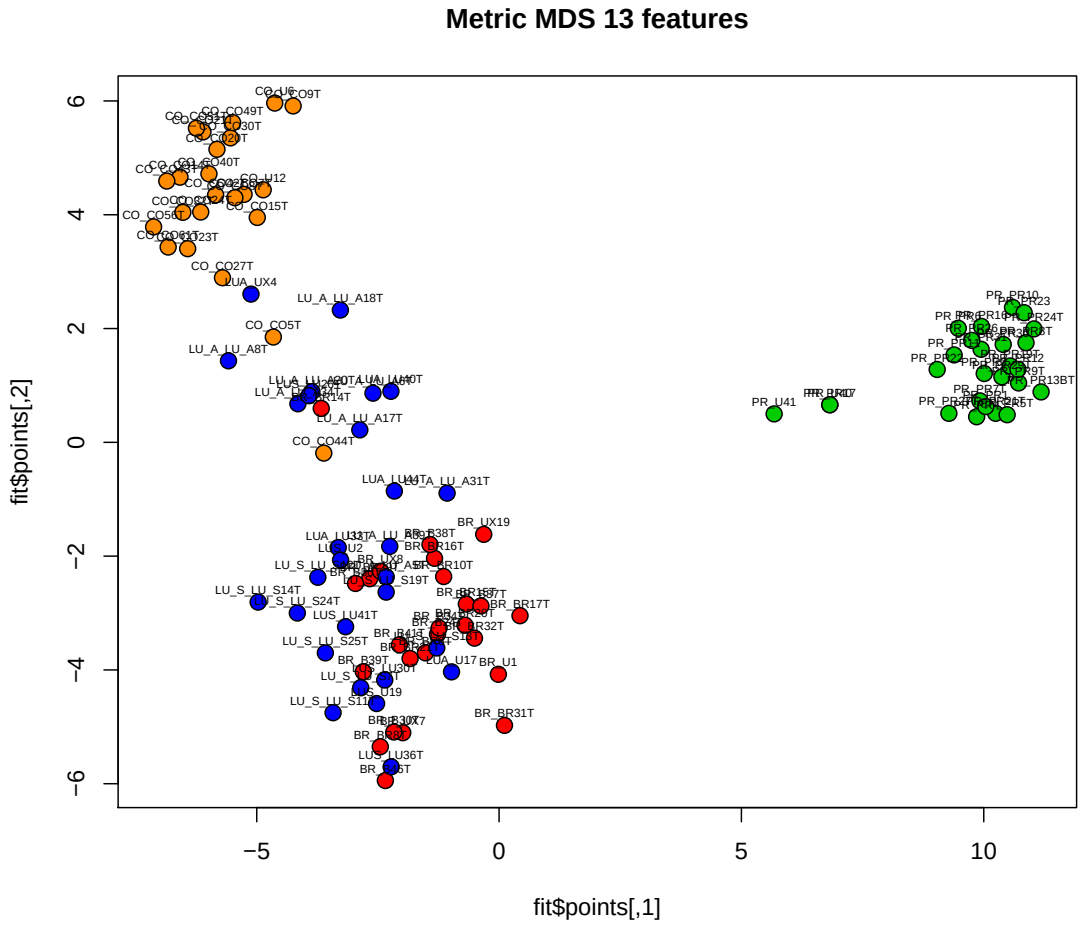


Figure 6.3: Classical metric multidimensional scaling applied to multiple tissue data and down-projected to 2 dimensions. The 13 features with largest variance are selected.

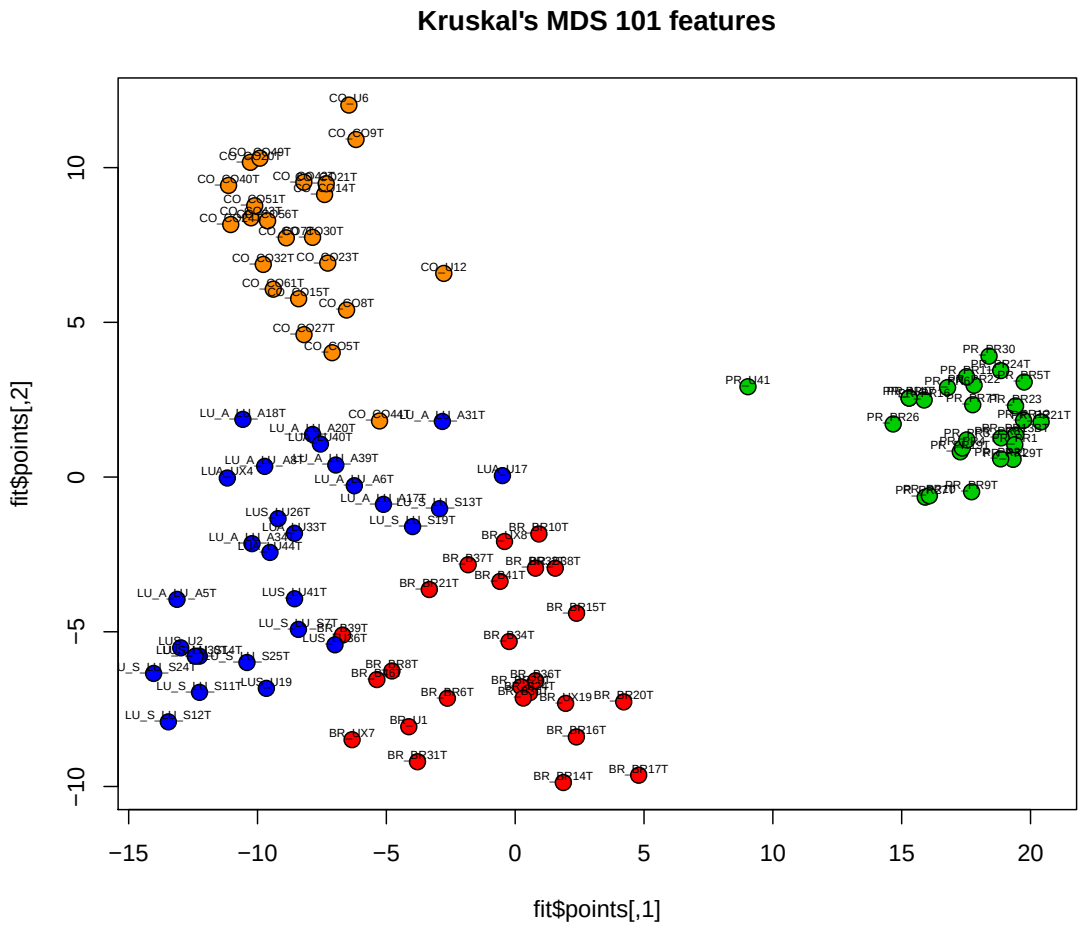


Figure 6.4: Multidimensional scaling with Kruskal's measure applied to multiple tissue data down-projected to 2 dimensions. The 101 features with largest variance are selected.

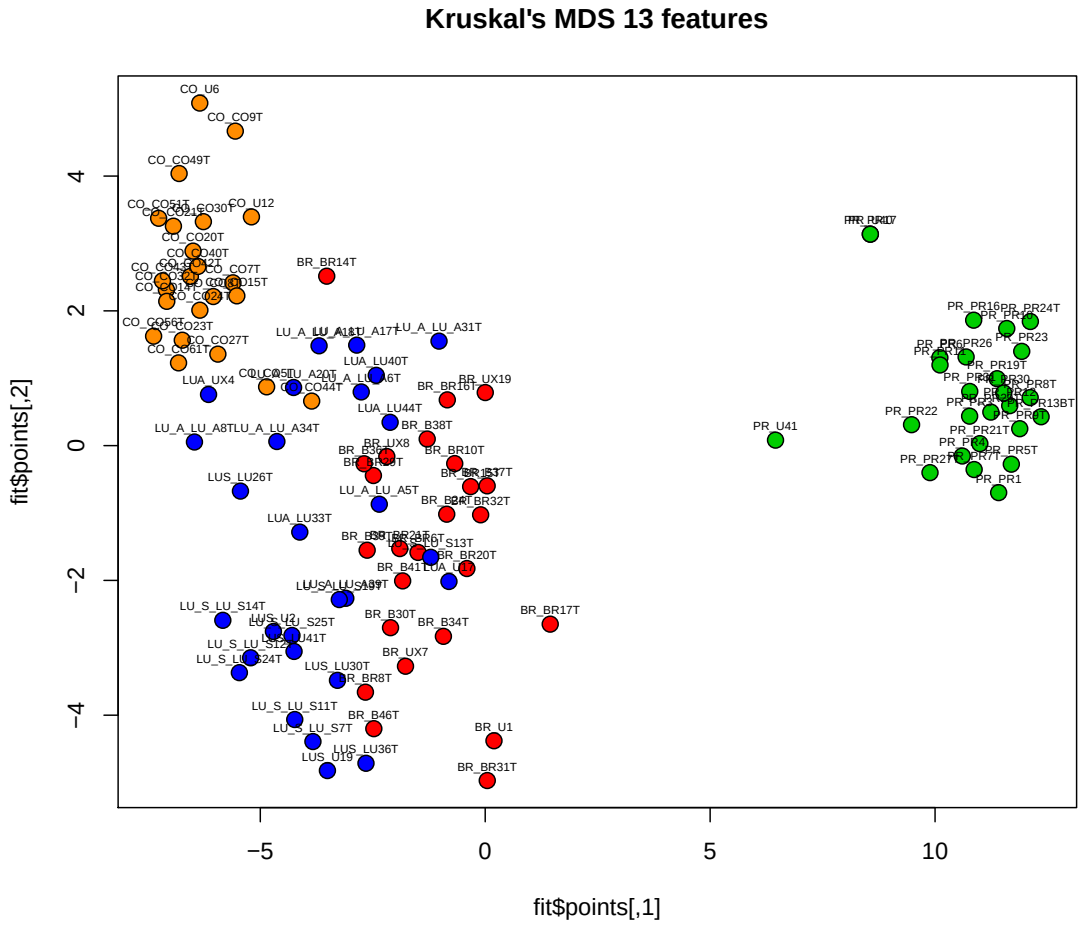


Figure 6.5: Multidimensional scaling with Kruskal's measure applied to multiple tissue data down-projected to 2 dimensions. The 13 features with largest variance are selected.

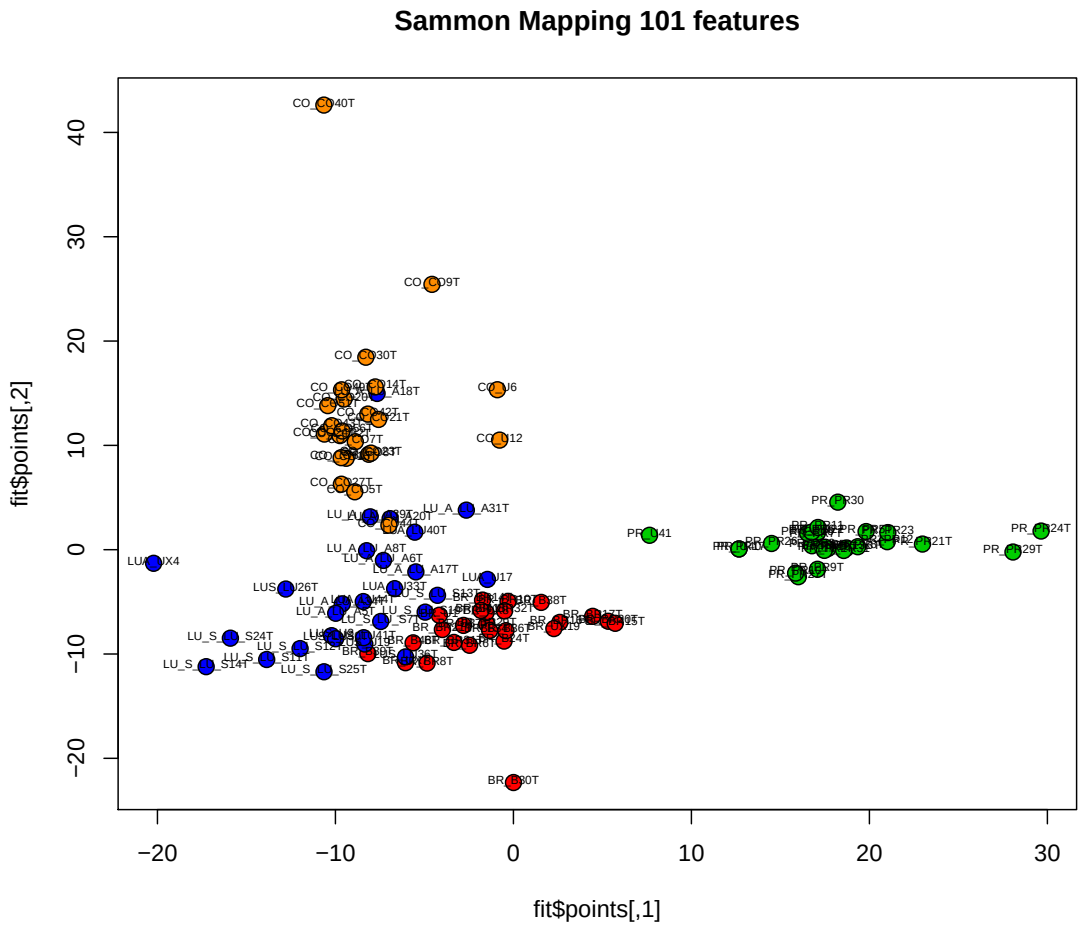


Figure 6.6: Sammon' mapping applied to multiple tissue data down-projected to 2 dimensions. The 101 features with largest variance are selected.

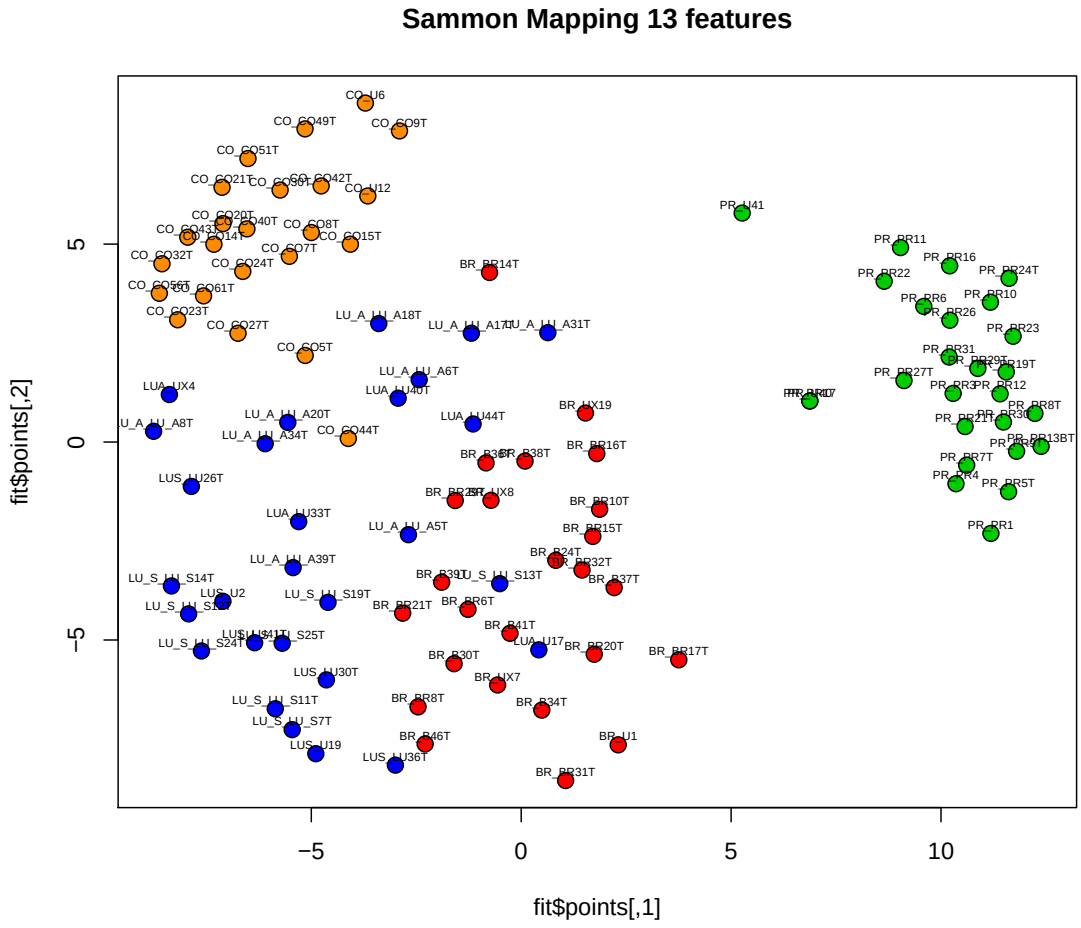


Figure 6.7: Sammon' mapping applied to multiple tissue data down-projected to 2 dimensions. The 13 features with largest variance are selected.

6.3 Non-negative Matrix Factorization

6.3.1 The Method

Non-negative matrix factorization (NFM) Lee and Seung [1999, 2001], Hoyer [2004] is a matrix factorization method where all matrix entries are assumed to be positive: both of the observation matrix and the decomposition matrices. In contrast to PCA, ICA, or factor analysis, the non-negativity constraints make the representation of the observations purely additive (allowing no subtractions). A motivation for NFM is a parts-based representation: only parts are added to the observation but never subtracted. For example, to images objects can be added but non-existing objects are not subtracted.

Non-negative matrix factorization is concerned with computing a multiplicative decomposition of a *positive* data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ into *two positive matrices* $\boldsymbol{Y} \in \mathbb{R}^{n \times l}$, and $\boldsymbol{U} \in \mathbb{R}^{m \times l}$ in the following way:

$$\boldsymbol{X} \;=\; \boldsymbol{Y}\,\boldsymbol{U}^T \;=\; \sum_{k=1}^{l} \boldsymbol{y}_k\,\boldsymbol{u}_k^T\,, \tag{6.10}$$

where $0 \leq X_{ij}$, $0 \leq Y_{ik} = [\boldsymbol{y}_k]_i$, and $0 \leq U_{jk} = [\boldsymbol{u}_k]_j$. The right hand side of this equation expresses the model as the sum of outer products of column vectors $\boldsymbol{y}_j \in \mathbb{R}^n$ and the $\boldsymbol{u}_j \in \mathbb{R}^m$.

First we consider NFM with the Kullback-Leibler divergence as objective for the reconstruction error Lee and Seung [1999, 2001]. The objective is the Kullback-Leibler distance between two matrices:

$$\mathrm{D}(\boldsymbol{A} \parallel \boldsymbol{B}) \;=\; \sum_{ij} \left(A_{ij}\,\log \frac{A_{ij}}{B_{ij}} \;+\; A_{ij} \;-\; B_{ij} \right) \tag{6.11}$$

It is only the Kullback-Leibler divergence if $\sum_{ij} A_{ij} = \sum_{ij} B_{ij} = 1$, because in this case the matrices can be viewed as probabilities as they are positive according to our assumptions.

We minimize the Kullback-Leibler divergence $\mathrm{D}(\boldsymbol{X} \parallel \boldsymbol{Y}\,\boldsymbol{U}^T)$ by gradient descent, which gives the following update rules:

$$Y_{ik} \;=\; Y_{ik} \frac{\sum_{j=1}^{m} U_{jk}\,X_{ij} \;/\; \left(\boldsymbol{Y}\,\boldsymbol{U}^T\right)_{ij}}{\sum_{j=1}^{m} U_{jk}} \tag{6.12}$$

and

$$U_{jk} \;=\; U_{jk} \frac{\sum_{i=1}^{n} Y_{ik}\,X_{ij} \;/\; \left(\boldsymbol{Y}\,\boldsymbol{U}^T\right)_{ij}}{\sum_{i=1}^{n} Y_{ik}}\,. \tag{6.13}$$

The second approach to non-negative matrix factorization uses the Euclidean distance as objective to measure the reconstruction error Lee and Seung [2001], Paatero and Tapper [1997]. The objective is the Euclidean distance (Frobenius norm) between two matrices:

$$\|\boldsymbol{A} \;-\; \boldsymbol{B}\|_F^2 \;=\; \sum_{ij} (A_{ij} \;-\; B_{ij})^2\,. \tag{6.14}$$

We minimize the Euclidean distance $\|\boldsymbol{X} - \boldsymbol{Y}\,\boldsymbol{U}^T\|_F^2$ by the following update rules:

$$Y_{ik} = Y_{ik}\frac{(\boldsymbol{X}\,\boldsymbol{U})_{ik}}{(\boldsymbol{Y}\,\boldsymbol{U}^T\,\boldsymbol{U})_{ik}} \tag{6.15}$$

and

$$U_{jk} = U_{jk}\frac{\left(\boldsymbol{Y}^T\,\boldsymbol{X}\right)_{kj}}{(\boldsymbol{Y}^T\,\boldsymbol{Y}\,\boldsymbol{U}^T)_{kj}}\,. \tag{6.16}$$

The factor of the first multiplicative update rule is obtained by multiplying

$$\boldsymbol{X} = \boldsymbol{Y}\,\boldsymbol{U}^T \tag{6.17}$$

from the right by $\boldsymbol{U}$ while the second multiplicative update rule is obtained by multiplying from the left by $\boldsymbol{Y}^T$. For a fixed point, the left and the right hand side have to be equal, however for small l that is not possible. The update rules for both approaches (Kullback-Leibler divergence and Euclidean distance) have been proved to converge to a local optimum Lee and Seung [2001].

NFM has been extended to *sparse non-negative matrix factorization* which combines non-negativity and sparseness Hoyer [2004]. In particular solutions become unique if sparseness constraints are introduced. Sparseness can be imposed on both decomposition matrices. If the matrix $\boldsymbol{Y}$ is considered as an indicator matrix of which parts are present in the observations, then sparseness means that only few parts are present in one observation. If the matrix $\boldsymbol{U}$ is considered as how the parts are characterized or measured by the observations, then sparseness means that only few measurements characterize one part. For example, in gene expression $\boldsymbol{X}$ is a samples-genes measurement matrix. These measurements may be decomposed into pathways and their activation. $\boldsymbol{Y}$ may indicate which pathway is activated in which sample. Sparseness means that only few pathways are activated in a particular sample. For gene expression $\boldsymbol{U}$ may indicate which genes belong to which pathway. Sparseness means that only few genes belong to a pathway.

6.3.2 Examples

Fig. 6.8 shows an example from Lee and Seung [1999] of non-negative matrix factorization (NMF). NFM learns parts-based representations of faces, whereas vector quantization (VQ) and principal components analysis (PCA) learn holistic representations.

We generated positive toy data and apply different non-negative matrix factorization methods to it. Fig. 6.9 shows the noise data (left) and the noise free data of this non-negative matrix factorization problem. The data contains blocks of patterns. These blocks should be recognized by the NFM methods, where $\boldsymbol{Y}$ and $\boldsymbol{U}$ indicate which rows and which columns, respectively, belong to a block. For visualization purposes only, the blocks are constructed by adjacent row or column elements. In real data there may be such blocks but no row and column sorting can visualize them simultaneously — only two blocks can always be visualized simultaneously by proper row and column sorting.

First we applied NMF with the Kullback-Leibler divergence as objective to the data and then plotted the results. Fig. 6.10 shows the results for the Kullback-Leibler divergence as objective. The right panel shows the reconstructed data while the left panel shows the reconstruction error.

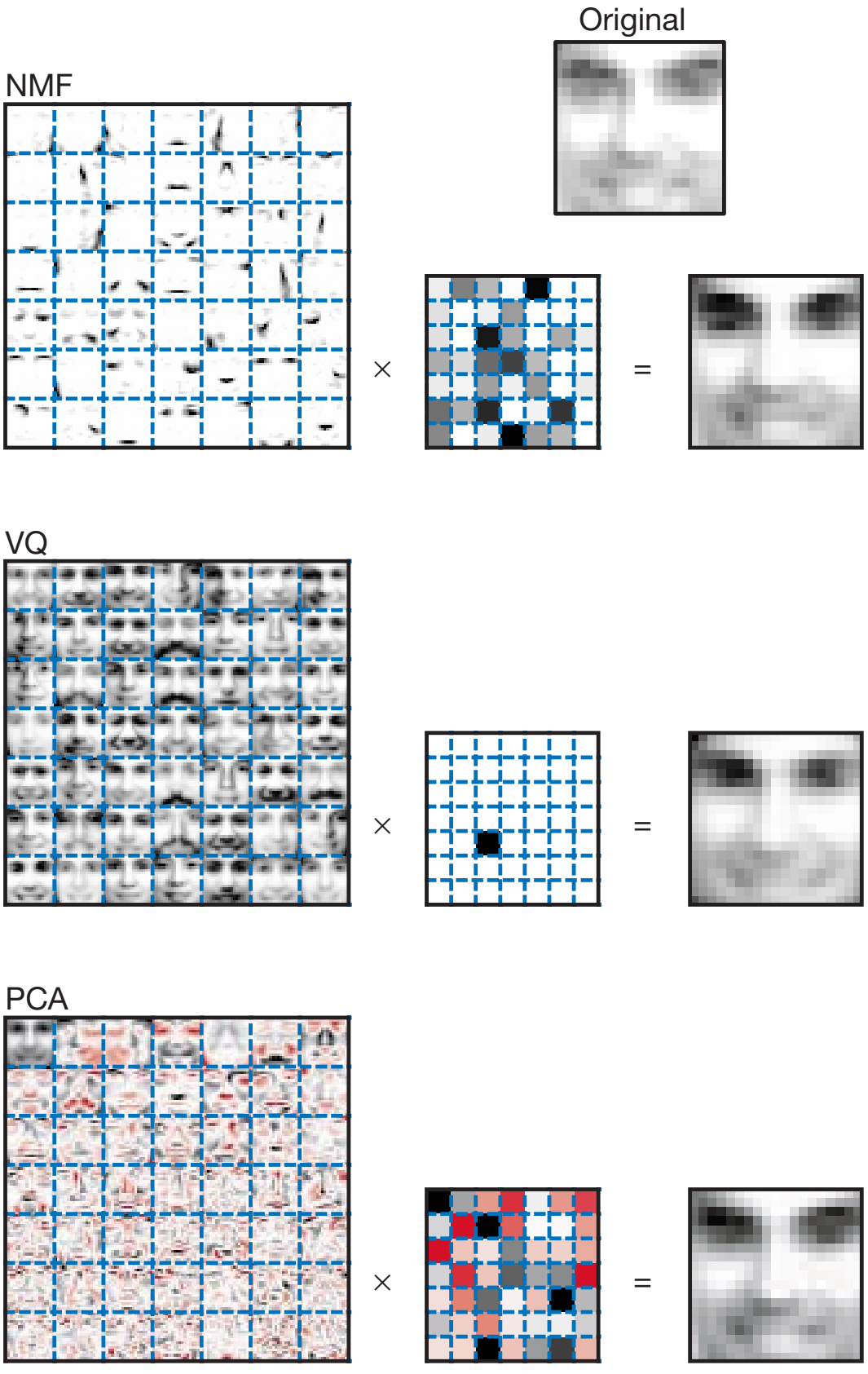


Figure 6.8: Non-negative matrix factorization (NMF) learns parts-based representations of faces, whereas vector quantization (VQ) and principal components analysis (PCA) learn holistic representations. Figure is from Lee and Seung [1999].

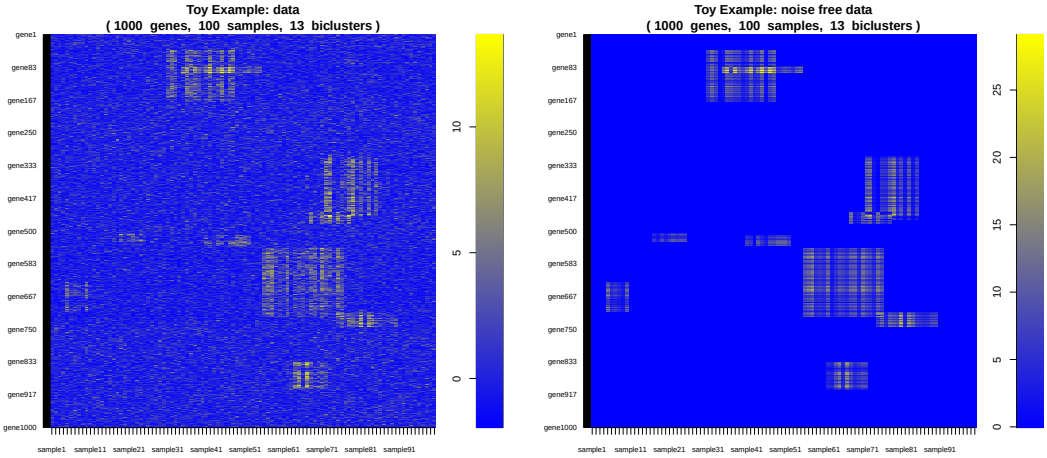


Figure 6.9: Non-negative toy data (left) and noise-free data (right) for non-negative matrix factorization. The data contains blocks of patterns which is actually a biclustering problem. For visualization the blocks are constructed by adjacent row or column elements.

The original matrix is scaled and, therefore, the range of the error may be different for different methods (e.g. the biclustering method FABIA scales the data). Fig. 6.11 shows the matrix $\boldsymbol{Y}$ at the left panel and the matrix $\boldsymbol{U}^T$ at the right panel. The blocks are not as well detected as with FABIA (see below).

Next we applied NMF with the Euclidean distance as objective to the data and plot the results. Fig. 6.12 shows the results for the Euclidean distance as objective. The right panel shows the reconstructed data while the left panel shows the reconstruction error. Fig. 6.13 shows the matrix $\boldsymbol{Y}$ at the left panel and the matrix $\boldsymbol{U}^T$ at the right panel. The blocks are not as well detected as with FABIA (see below).

Then we applied NMF with a sparseness constraint Hoyer [2004] to the data and plot the results. Fig. 6.14 shows the reconstructed data and error while Fig. 6.15 shows the matrices into which the data matrix was factorized. Not all blocks are detected, because too much sparseness was enforced onto the matrix factorization because it was difficult to properly adjust the parameter which controls sparseness.

Further we applied matrix factorization with a sparseness constraint Hoyer [2004] to the data and plot the results. That means we did not enforce non-negativity. Fig. 6.16 shows the reconstructed data and error while Fig. 6.17 shows the matrices into which the data matrix was factorized. Also in this case not all blocks are detected, because too much sparseness was enforced onto the matrix factorization because it was difficult to properly adjust the parameter which controls sparseness.

Finally, we applied the biclustering method FABIA Hochreiter et al. [2010] to the data and then plotted the results. FABIA is based on a sparse factor analysis model, where both the factors and the loadings are sparse. Thus, both the matrix $\boldsymbol{Y}$ as well as the matrix $\boldsymbol{U}$ are sparse if they are the result of FABIA. Fig. 6.18 shows the reconstructed data and error. Fig. 6.19 shows the matrices into which the data matrix was factorized. The original matrix is scaled for some methods like FABIA and, therefore, the range of the error may be different for different methods.

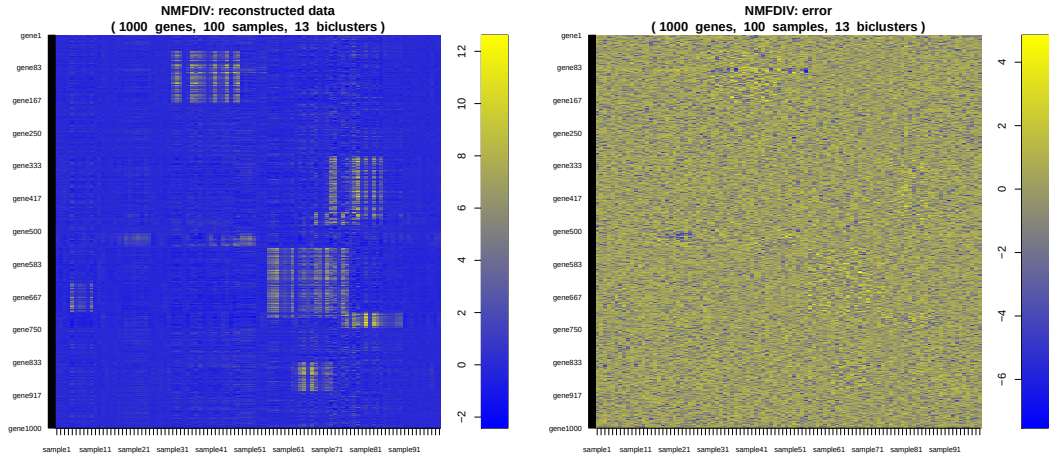


Figure 6.10: Non-negative matrix factorization using the Kullback-Leibler divergence as objective. Left: reconstructed data. Right: reconstruction error.

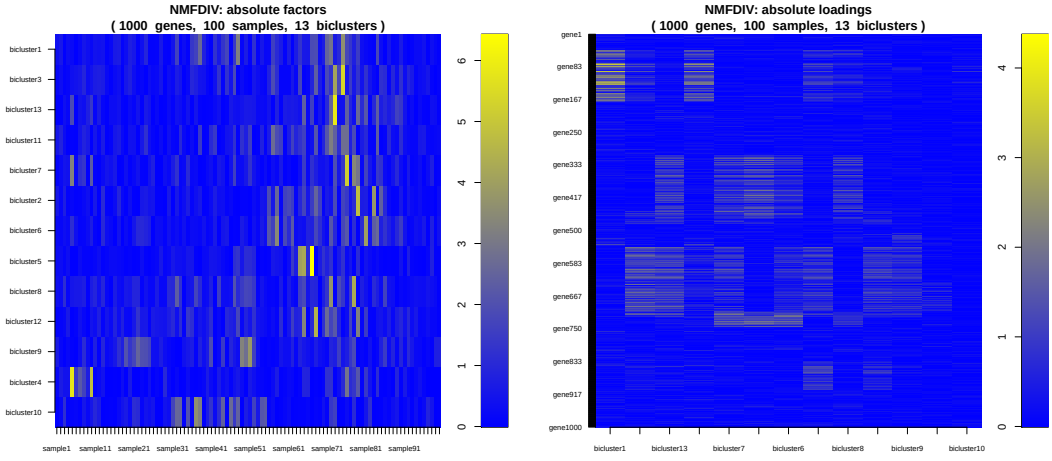


Figure 6.11: Non-negative matrix factorization using the Kullback-Leibler divergence as objective. Left: $\boldsymbol{Y}$ matrix. Right: $\boldsymbol{U}^T$ matrix.

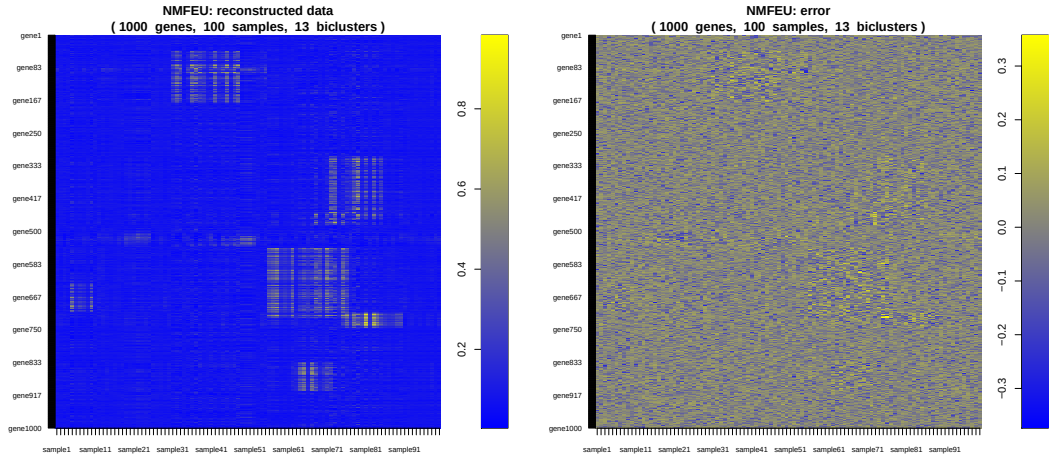


Figure 6.12: Non-negative matrix factorization using the Euclidean distance as objective. Left: reconstructed data. Right: reconstruction error.

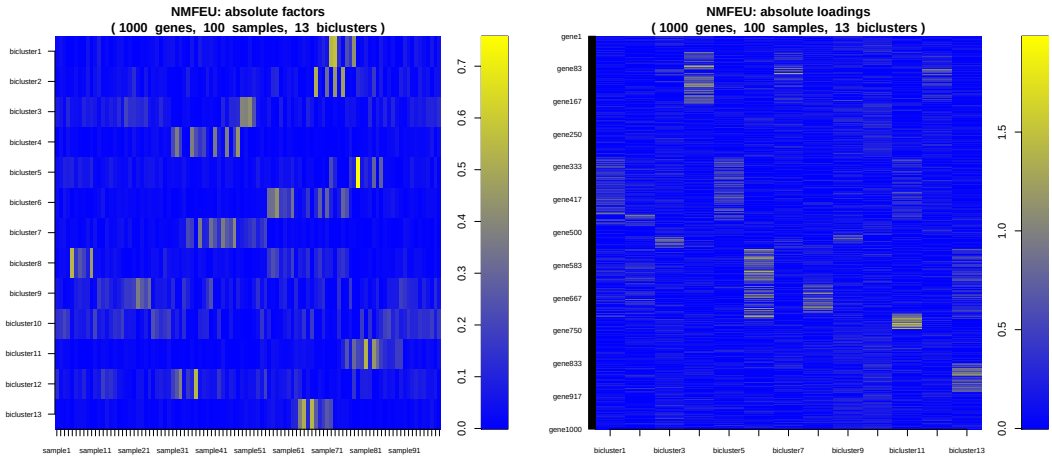


Figure 6.13: Non-negative matrix factorization using the Euclidean distance as objective. Left: $\boldsymbol{Y}$ matrix. Right: $\boldsymbol{U}^T$ matrix.

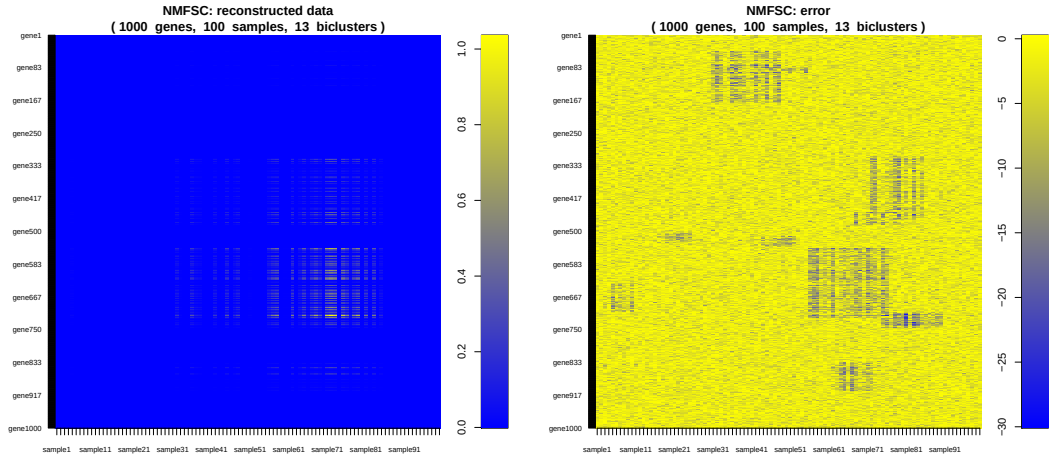


Figure 6.14: Non-negative matrix factorization using a sparseness constraint. Left: reconstructed data. Right: reconstruction error.

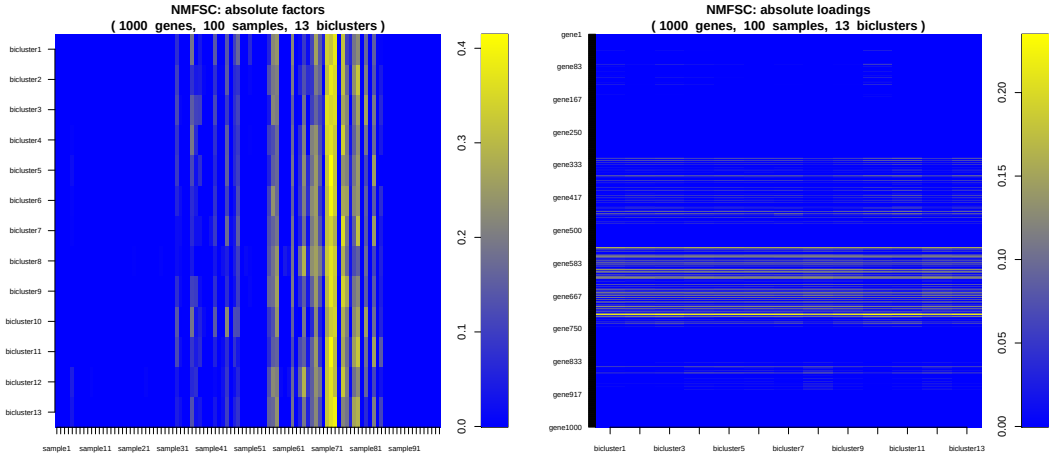


Figure 6.15: Non-negative matrix factorization using a sparseness constraint. Left: $\boldsymbol{Y}$ matrix. Right: $\boldsymbol{U}^T$ matrix.

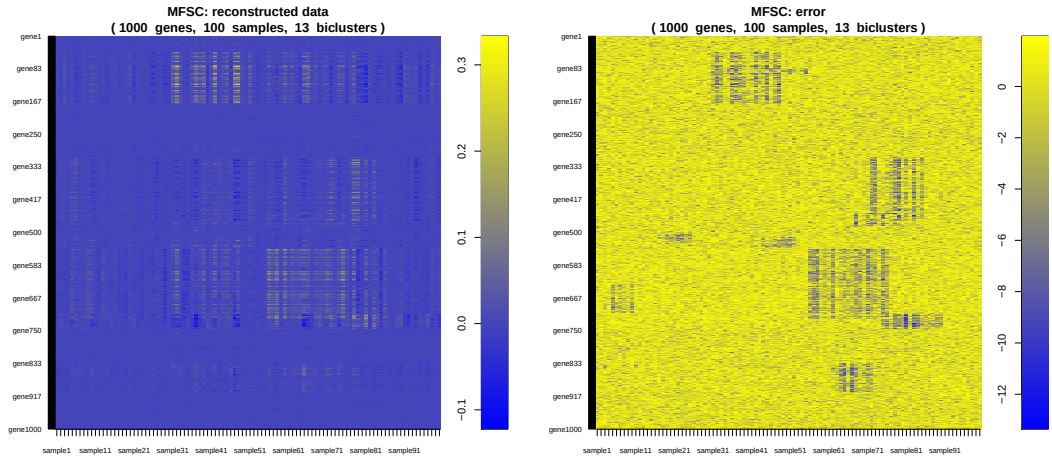


Figure 6.16: Matrix factorization using a sparseness constraint. Left: reconstructed data. Right: reconstruction error.

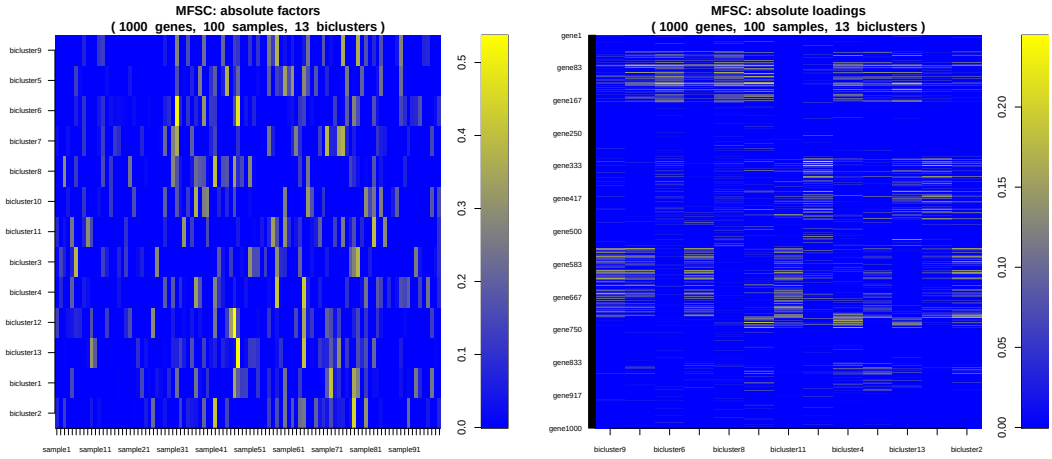


Figure 6.17: Matrix factorization using a sparseness constraint. Left: $\boldsymbol{Y}$ matrix. Right: $\boldsymbol{U}^T$ matrix.

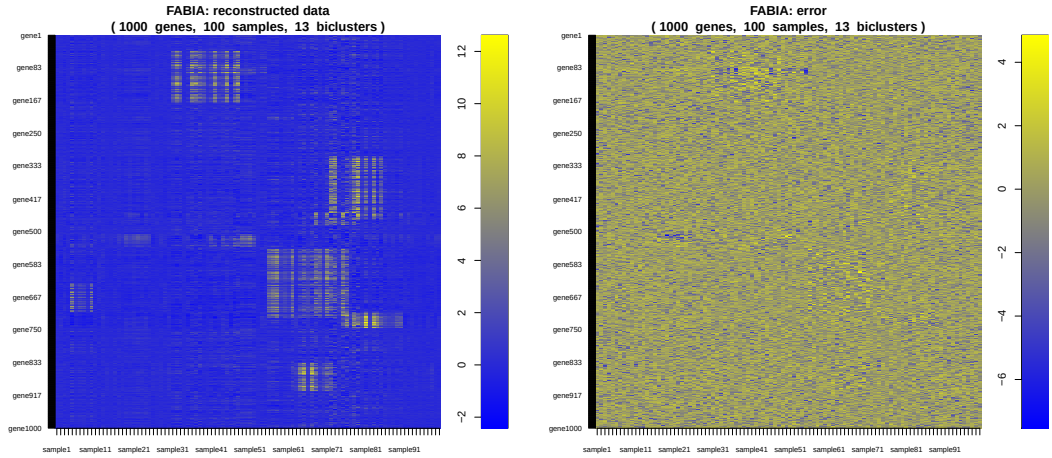


Figure 6.18: FABIA biclustering. Left: reconstructed data. Right: reconstruction error.

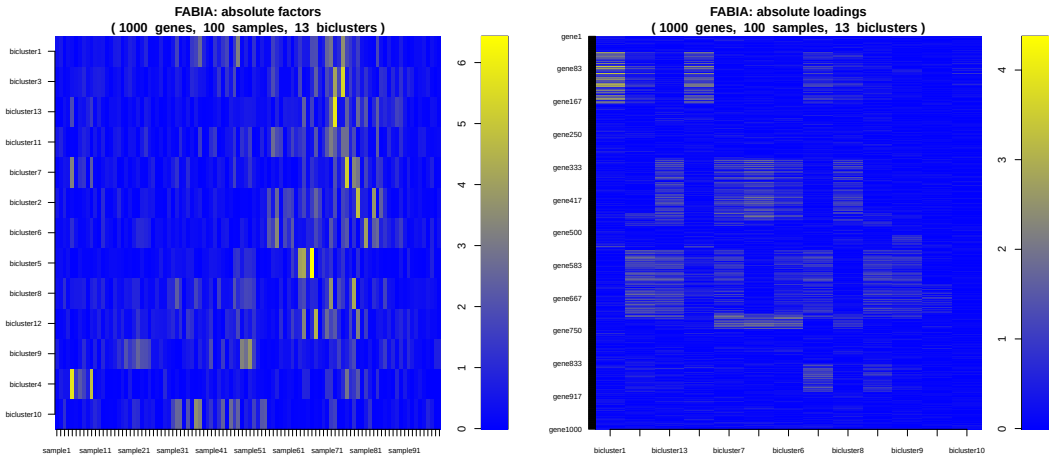


Figure 6.19: FABIA biclustering. Left: $\boldsymbol{Y}$ matrix. Right: $\boldsymbol{U}^T$ matrix.

6.4 Locally Linear Embedding

6.4.1 The Method

Locally linear embedding (LLE) computes low-dimensional, neighborhood-preserving embeddings / representations of high-dimensional observations. LLE maps observations into a single global coordinate system of lower dimensionality. In contrast to previous projection methods, except multidimensional scaling, LLE performs nonlinear mappings.

The objective of LLE

$$\varepsilon(\boldsymbol{W}) \;=\; \sum_i \left\| \boldsymbol{x}_i \;-\; \sum_{j=1}^{k} W_{ij} \boldsymbol{x}_j \right\|^2 \tag{6.18}$$

is minimized with respect to $\boldsymbol{W}$ by constrained least squares using only neighbors $\boldsymbol{x}_j$ of $\boldsymbol{x}_i$ and enforcing $\sum_{j=1}^{k} W_{ij} = 1$. The solutions W_{ij} of this problem are invariant to rotations, rescalings, and translations of data point $\boldsymbol{x}_i$ and its neighbors $\boldsymbol{x}_j$. This means that $\boldsymbol{x}_i$ is represented as a weighted sum of its neighbors.

Down-projection optimizes the objective

$$\Phi(\boldsymbol{Y}) \;=\; \sum_i \left\| \boldsymbol{y}_i \;-\; \sum_{j=1}^{k} W_{ij} \boldsymbol{y}_j \right\|^2 \;, \tag{6.19}$$

where the W_{ij} are fixed but the $\boldsymbol{y}_i$ are optimized. This means the representation of $\boldsymbol{x}_i$ by its neighbors is now transferred to $\boldsymbol{y}_i$ which should have the same representation by its neighbors. This quadratic problem is solved by a sparse eigenvalue problem.

The objective $\Phi(\boldsymbol{Y})$ can be represented as

$$\Phi(\boldsymbol{Y}) \;=\; \sum_{ij} M_{ij}\; \boldsymbol{y}_i^T \boldsymbol{y}_j \;, \tag{6.20}$$

where

$$M_{ij} \;=\; \delta_{ij} - W_{ij} \;-\; W_{ji} \;+\; \sum_k W_{ki}\, W_{kj} \;, \tag{6.21}$$

where δ_{ij} is the Kronecker delta which is 1 for $i = j$ and 0 otherwise. The matrix $\boldsymbol{M}$ can be represented by

$$\boldsymbol{M} \;=\; (\boldsymbol{I} \;-\; \boldsymbol{W})^T \;(\boldsymbol{I} \;-\; \boldsymbol{W}) \;. \tag{6.22}$$

The optimal embedding is found by the bottom d eigenvectors of this matrix, except the last one.

Fig. 6.20 depicts the steps of the LLE method. Alg. 6.1 presents a pseudo code for LLE.

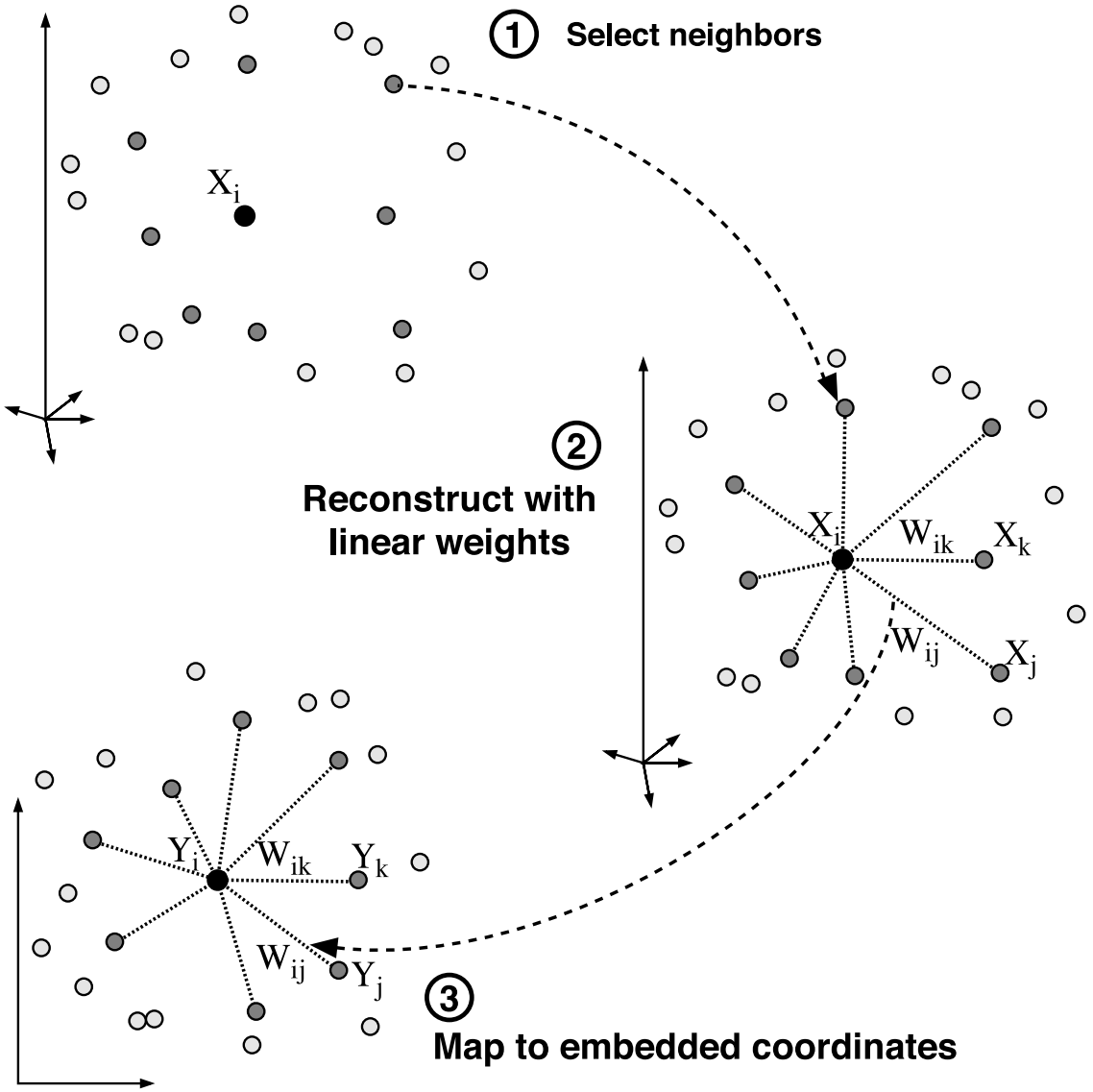


Figure 6.20: Steps of locally linear embedding: (1) Assign neighbors to each observation $\boldsymbol{x}_i$ by using k nearest neighbors. (2) Compute the weights W_{ij} that best linearly reconstruct $\boldsymbol{x}_i$ from its neighbors, solving the constrained least-squares problem in Eq. (6.18). Compute the low-dimensional embedding vectors $\boldsymbol{y}_i$ which is also reconstructed by W_{ij} from its neighbors, minimizing Eq. (6.19) by finding the smallest eigenmodes of the sparse symmetric matrix in Eq. (6.21).

Algorithm 6.1 Locally linear embedding

Given: $\boldsymbol{X}$: n by m matrix consisting of n data items in m dimensions, dimension of embedding space l, k number of neighbors, distance measure

Find neighbors in $\boldsymbol{X}$ space

for $(i = 1\ ;\ i\ \geq\ n\ ;\ i++)$ **do**

compute the distance from $\boldsymbol{x}_i$ to every other point $\boldsymbol{x}_j$

find the k smallest distances

assign the corresponding points to be neighbors of $\boldsymbol{x}_i$

end for

Solve for reconstruction weights $\boldsymbol{W}$

for $(i = 1\ ;\ i\ \geq\ n\ ;\ i++)$ **do**

create matrix $\boldsymbol{Z}$ consisting of all neighbors of $\boldsymbol{x}_i$ [d]

subtract $\boldsymbol{x}_i$ from every row of $\boldsymbol{Z}$

compute the local covariance $\boldsymbol{C} = \boldsymbol{Z}^T\boldsymbol{Z}$ [e]

solve linear system $\boldsymbol{C}\boldsymbol{w} = \boldsymbol{1}$ for $\boldsymbol{w}$ [f]

set $W_{ij} = 0$ if j is not a neighbor of i

set the remaining elements in the i-th row of $\boldsymbol{W}$ equal to $\boldsymbol{w}/\sum_j(w_j)$;

end for

Compute embedding coordinates $\boldsymbol{Y}$ using weights $\boldsymbol{W}$

create sparse matrix $\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{W})^T(\boldsymbol{I} - \boldsymbol{W})$

find bottom $l + 1$ eigenvectors of $\boldsymbol{M}$ (corresponding to the $d + 1$ smallest eigenvalues)

set the q-th *column* of $\boldsymbol{Y}$ to be the $q + 1$ smallest eigenvector (discard the bottom eigenvector $\boldsymbol{1} = (1, 1, 1, 1...)$ with eigenvalue zero)

Result $\boldsymbol{Y}$: n by l matrix consisting of $l < m$ dimensional embedding coordinates.

Comments:

[a] Notation $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ denote the i-th row of $\boldsymbol{X}$ and $\boldsymbol{Y}$ (in other words the data and embedding coordinates of the i-th point),
$\boldsymbol{M}^T$ denotes the transpose of matrix $\boldsymbol{M}$,
$\boldsymbol{I}$ is the identity matrix,
$\boldsymbol{1}$ is a column vector of all ones

[b] This can be done in a variety of ways, for example above we compute the k nearest neighbors using Euclidean distance. Other methods such as epsilon-ball include all points within a certain radius or more sophisticated domain specific and/or adaptive local distance metrics.

[c] Even for simple neighborhood rules like KNN or epsilon-ball using Euclidean distance, there are highly efficient techniques for computing the neighbors of every point, such as KD trees.

[d] $\boldsymbol{Z}$ consists of all rows of $\boldsymbol{X}$ corresponding to the neighbors of $\boldsymbol{x}_i$ but not $\boldsymbol{x}_i$ itself

[e] If $k > m$, the local covariance will not have full rank, and it should be regularized by setting $\boldsymbol{C} = \boldsymbol{C} + \epsilon\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and ϵ is a small constant of order 1e-3 $\mathrm{trace}(\boldsymbol{C})$. This ensures that the system to be solved in step 2 has a unique solution.

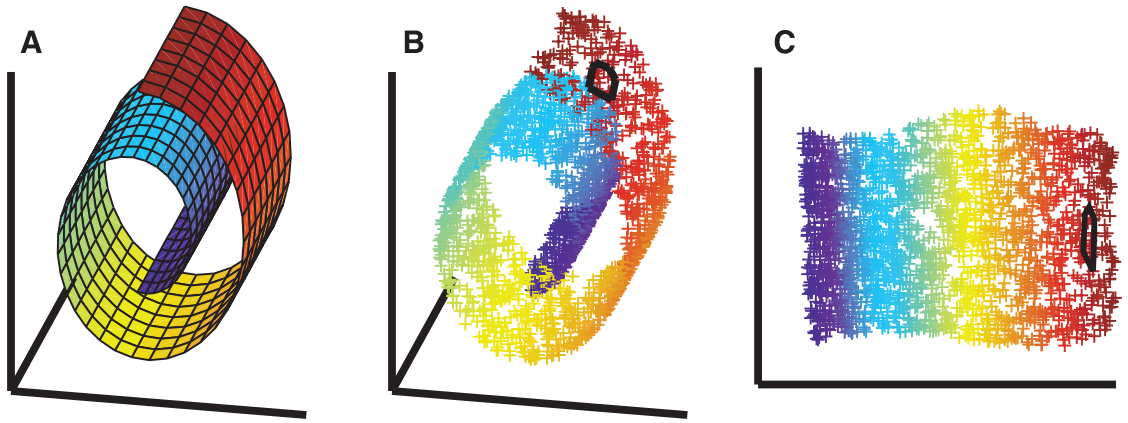


Figure 6.21: The "Swiss roll" data illustrates the Locally Linear Embedding method. (A) Original manifold on which the data points are located. The color coding illustrates the neighborhood which is preserved by LLE. Black outlines in the original data (B) and the mapped data (C) show the neighborhood of a single point.

6.4.2 Examples

Examples for the application of LLE can be found in Fig. 6.21 for the Swiss Roll data set and in and Fig. 6.22 for a data set of face images.

We test LLE on the "S" curve data. The points of this 3-dimensional data set form an "S" if the data is viewed from the right perspective (see Fig. 6.23 and Fig. 6.24). LLE is able to embed the points on the manifold into a 2-dimensional space (see Fig. 6.25).

We also applied LLE to the multiple tissue data set. We selected the 101 features with largest variance and then called LLE with different parameters k. Fig. 6.26 shows LLE results for the multiple tissue data set with different parameter settings. The results are not as good as with other methods because the observations seem not to be located on a manifold in the high-dimensional space. That was to be expected.

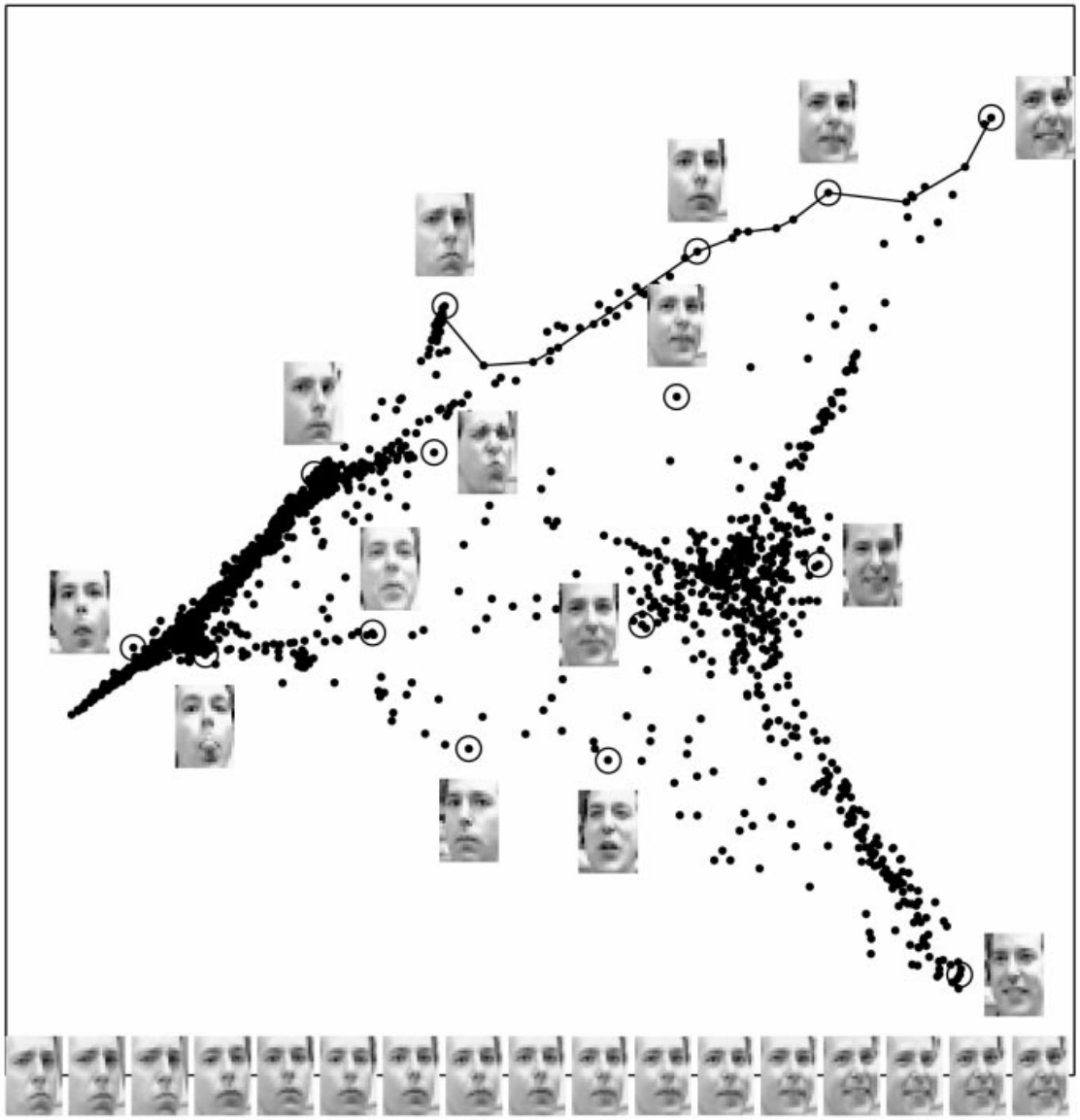


Figure 6.22: Images of faces mapped by LLE onto the embedding space described by the first two coordinates of LLE. Representative faces are shown next to circled points in different parts of the space. The bottom images correspond to points along the top-right path (linked by solid line).

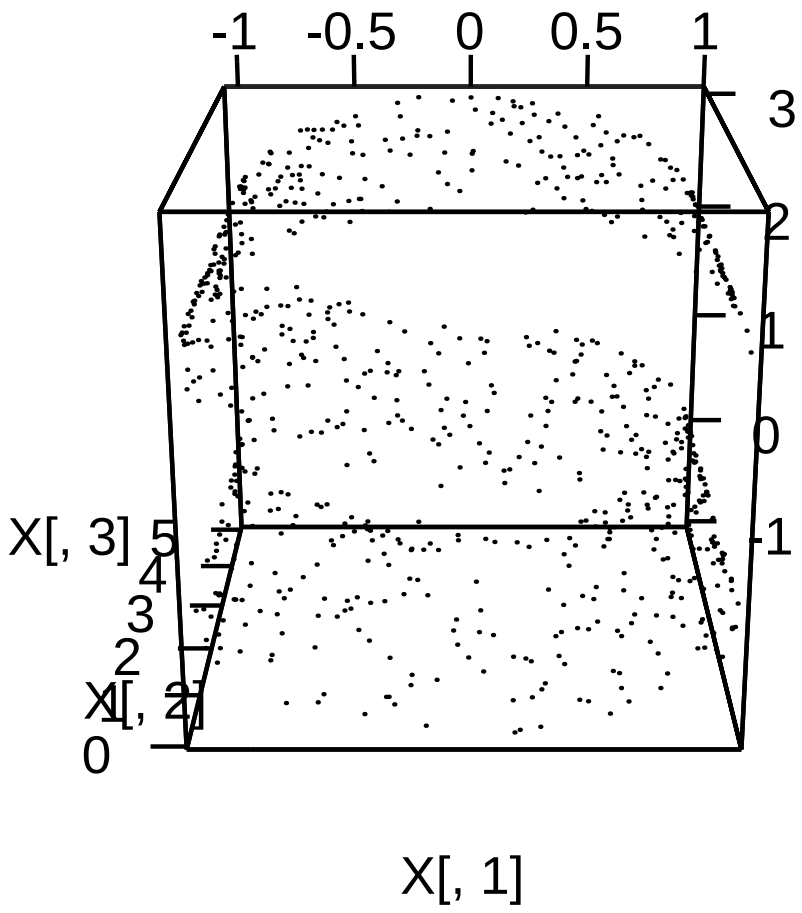


Figure 6.23: Data of points that form a "S" if viewed from the right angle. Here is the original view on the data.

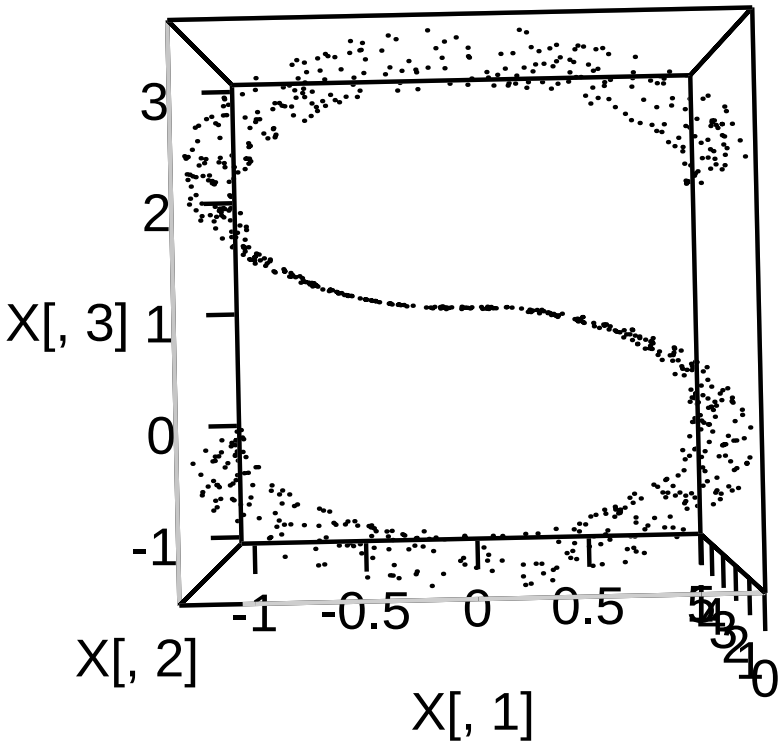


Figure 6.24: Data of points that form a “S” if viewed from the right angle. Here is a view which makes the “S” visible.

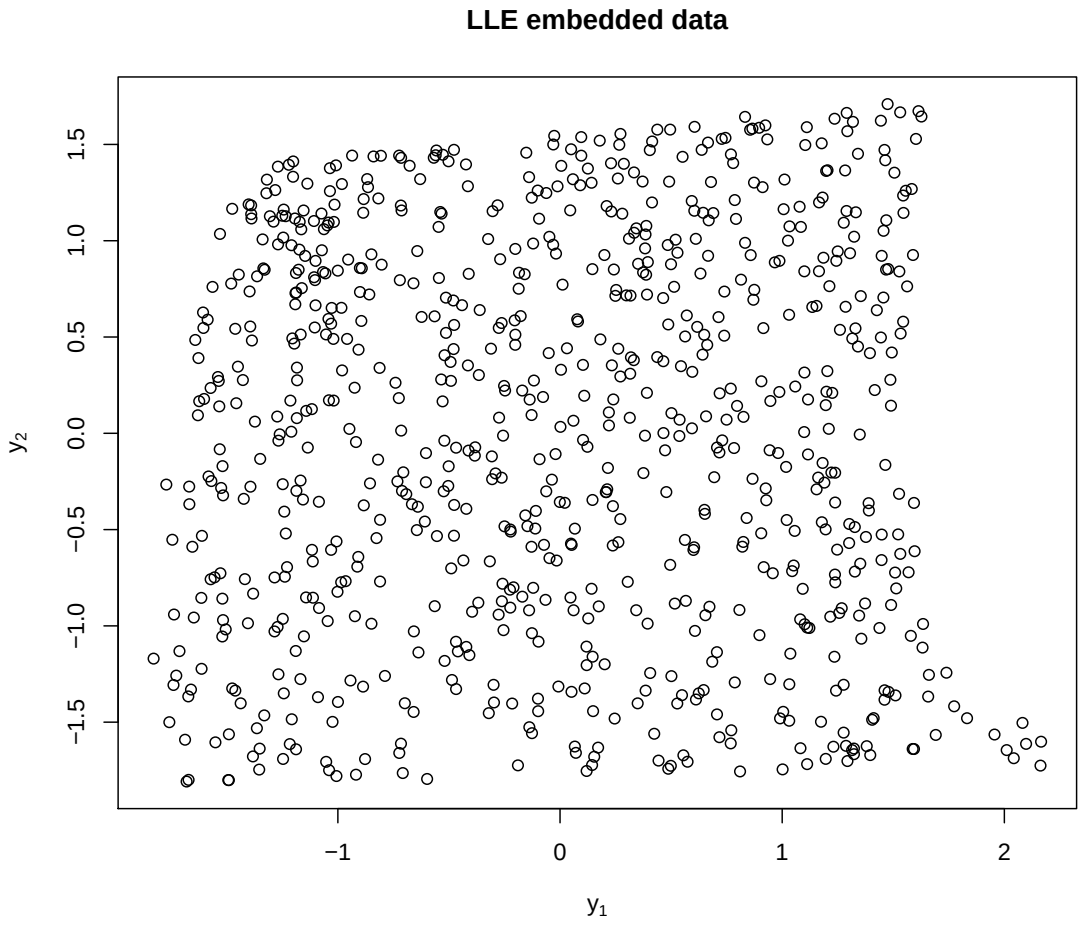


Figure 6.25: Result of LLE on the "S" curve data. The embedding into a 2-dimensional space is shown.

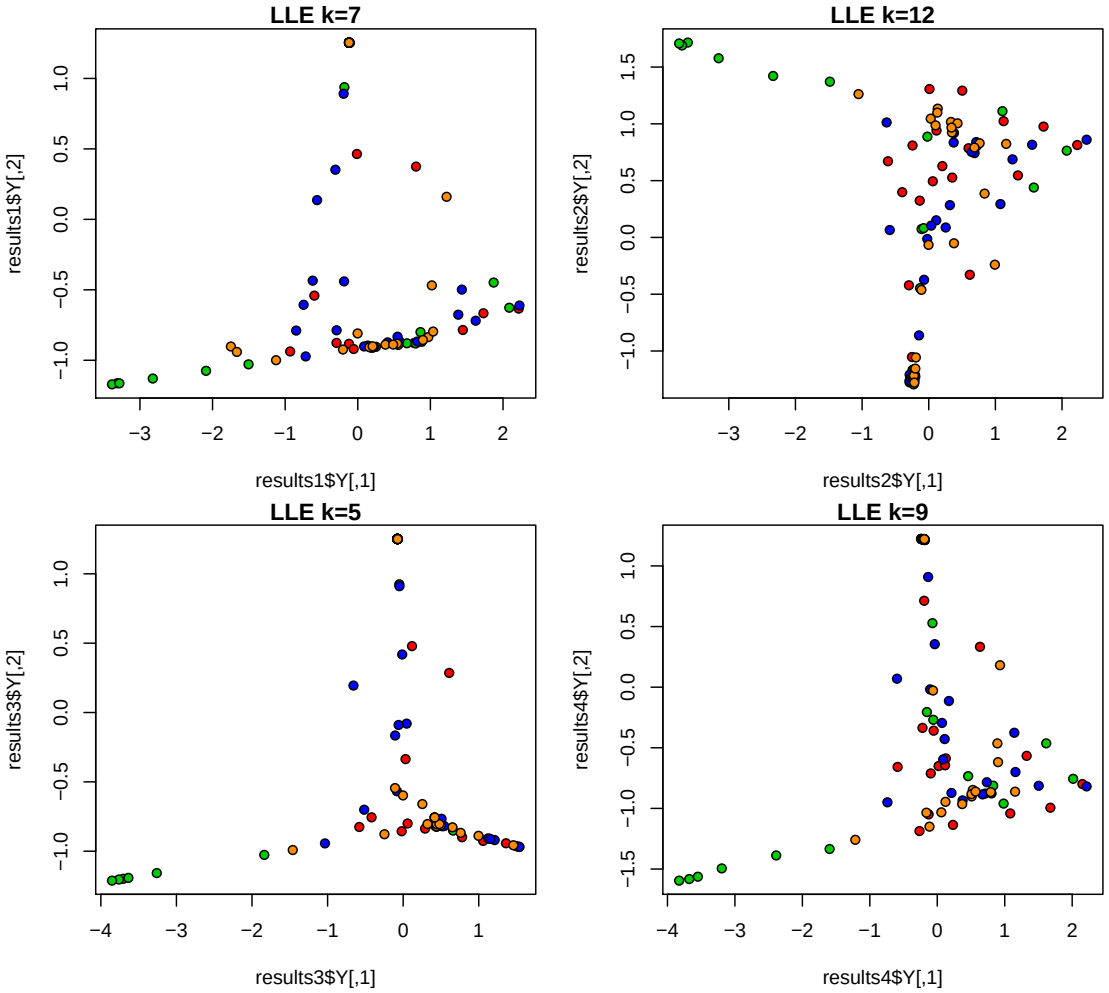


Figure 6.26: Result of LLE applied to the multiple tissue data set. The plots show the result of LLE for different parameter settings $k = 7$ (the optimal value with minimal ρ), $k = 12$, $k = 5$, and $k = 9$. LLE is not suited to this data set.

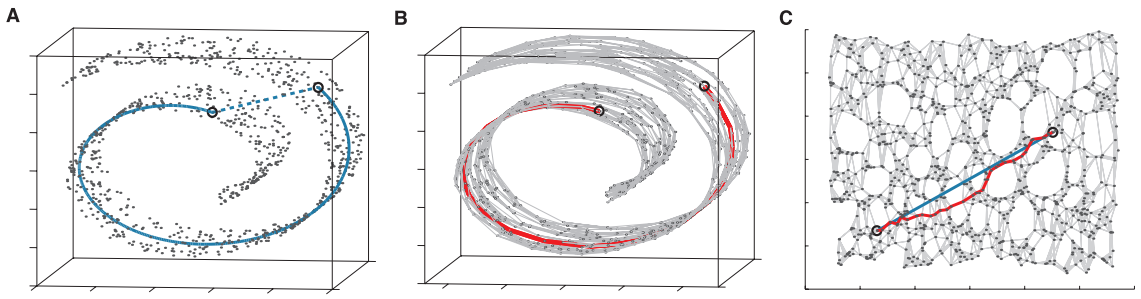


Figure 6.27: The "Swiss roll" data illustrates the Isomap method and the geodesic paths for non-linear dimensionality reduction. (A) The Euclidean distance of two points (circled) on a nonlinear manifold does not reflect their intrinsic similarity given by their geodesic distance. (B) Isomap constructs a neighborhood graph G which is used to approximate the geodesic path as the shortest path between the points in G (red). (C) The two-dimensional embedding recovered by Isomap.

6.5 Isomap

6.5.1 The Method

Very similar to LLE, also Isomap is a low-dimensional embedding method which computes a quasi-isometric, low-dimensional embedding of a set of high-dimensional observations. As LLE, Isomap is a non-linear projection method.

Metric MDS performs low-dimensional embedding based on the pairwise distance between data points using the Euclidean distance. In contrast to metric MDS, Isomap uses geodesic distance induced by a neighborhood graph embedded in the classical scaling. Isomap measures geodesic distances which are the shortest distances on a manifold in the high-dimensional space. Isomap approximates the geodesic distance by the sum of edge weights along the shortest path between two nodes and, thereby, assuming to stay on the manifold. See Fig. 6.27 for the Isomap method. The shortest path can be computed using Dijkstra's algorithm. The largest l eigenvectors of the geodesic distance matrix are coordinates in the l-dimensional projected space. This procedure is similar to PCA on the geodesic distance matrix, i.e. local PCA.

The connectivity of each data point in the neighborhood graph is defined as its nearest k Euclidean neighbors in the high-dimensional space. The connectivity is prone to "short-circuit errors" for too large k with respect to the manifold structure or to noise in the data. "Short-circuit errors" lead to a jump across the space to another location on the manifold.

The doubly centered geodesic distance matrix $\tau(\boldsymbol{D})$ in Isomap is a function of the geodesic distance matrix $\boldsymbol{D}$:

$$\tau(\boldsymbol{D}) \;=\; -\frac{1}{2}\,\boldsymbol{H}\,\boldsymbol{D}^2\,\boldsymbol{H}\;, \tag{6.23}$$

where $\boldsymbol{D}^2 = D_{ij}^2 = D_{ji}^2$ is the element-wise square of the geodesic distance matrix $\boldsymbol{D} = [D_{ij}]$, $\boldsymbol{H}$ is the centering matrix

$$\boldsymbol{H} \;=\; \boldsymbol{I}_n \;-\; \frac{1}{n}\,\mathbf{1}\,\mathbf{1}^T\;, \tag{6.24}$$

where $\mathbf{1} = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$. The doubly centered distance matrix $\tau(\boldsymbol{D})$ is not ensured to be positive semidefinite but can be made positive by a constant-shifting method.

The objective of Isomap is

$$\mathrm{E} \;=\; \|\tau(\boldsymbol{D}_X) \;-\; \tau(\boldsymbol{D}_Y)\|_{L^2} \;, \tag{6.25}$$

where $\boldsymbol{D}_Y$ is the matrix of Euclidean distances in the projected space, $\boldsymbol{D}_X$ is the matrix of geodesic distances, and τ converts distances to inner products as defined in Eq. (6.23). This objective can be minimized by setting the coordinates $\boldsymbol{y}_i$ to the top l eigenvectors of the matrix $\tau(\boldsymbol{D}_X)$. Alg. 6.2 shows a pseudo-code for the Isomap algorithm.

Algorithm 6.2 Isomap

Given: distances $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between pairs from n data points in an m-dimensional space X, parameter k or parameter e

Construct neighborhood graph

Define the graph G over all data points by connecting points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if they are closer than e (e-Isomap), or if i is one of the k nearest neighbors of j (k-Isomap). Closeness and neighborhood is measured by $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Set edge lengths equal to $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Compute shortest paths by Floyd's algorithm

for $(i = 1 \,;\; i \;\geq\; n \,;\; i++)$ **do**
  for $(j = 1 \,;\; j \;\geq\; n \,;\; j++)$ **do**
    Initialize $d_G(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ if i, j are linked by an edge; $d_G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \infty$, otherwise.
  end for
end for
for $(k = 1 \,;\; k \;\geq\; n \,;\; k++)$ **do**
  for $(i = 1 \,;\; i \;\geq\; n \,;\; i++)$ **do**
    for $(j = 1 \,;\; j \;\geq\; n \,;\; j++)$ **do**
      $d_G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \min\{d_G(\boldsymbol{x}_i, \boldsymbol{x}_j), d_G(\boldsymbol{x}_i, \boldsymbol{x}_k) + d_G(\boldsymbol{x}_k, \boldsymbol{x}_j)\}$.
    end for
  end for
end for
define shortest path matrix $\boldsymbol{D}_X$ by $[\boldsymbol{D}_X]_{ij} = d_G(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Construct l-dimensional embedding

Compute λ_p as the p-th eigenvalue (in decreasing order) of the matrix $\tau(\boldsymbol{D}_X)$, and v_{pi} as the i-th component of the p-th eigenvector.

set $y_{ij} = \sqrt{\lambda_i} v_{ji}$.

Result $\boldsymbol{Y}$: coordinate vectors $\boldsymbol{y}_i$ in a l-dimensional ($l < m$) Euclidean space Y

6.5.2 Examples

Fig. 6.28 show an example of Isomap ($k = 6$) applied to $n = 2000$ images (64 pixels by 64 pixels) of a hand in different configurations. The images were generated by making a series of opening and closing movements of the hand at different wrist orientations, designed to give rise to a two-dimensional manifold. The images were treated as 4096-dimensional vectors, with input-space

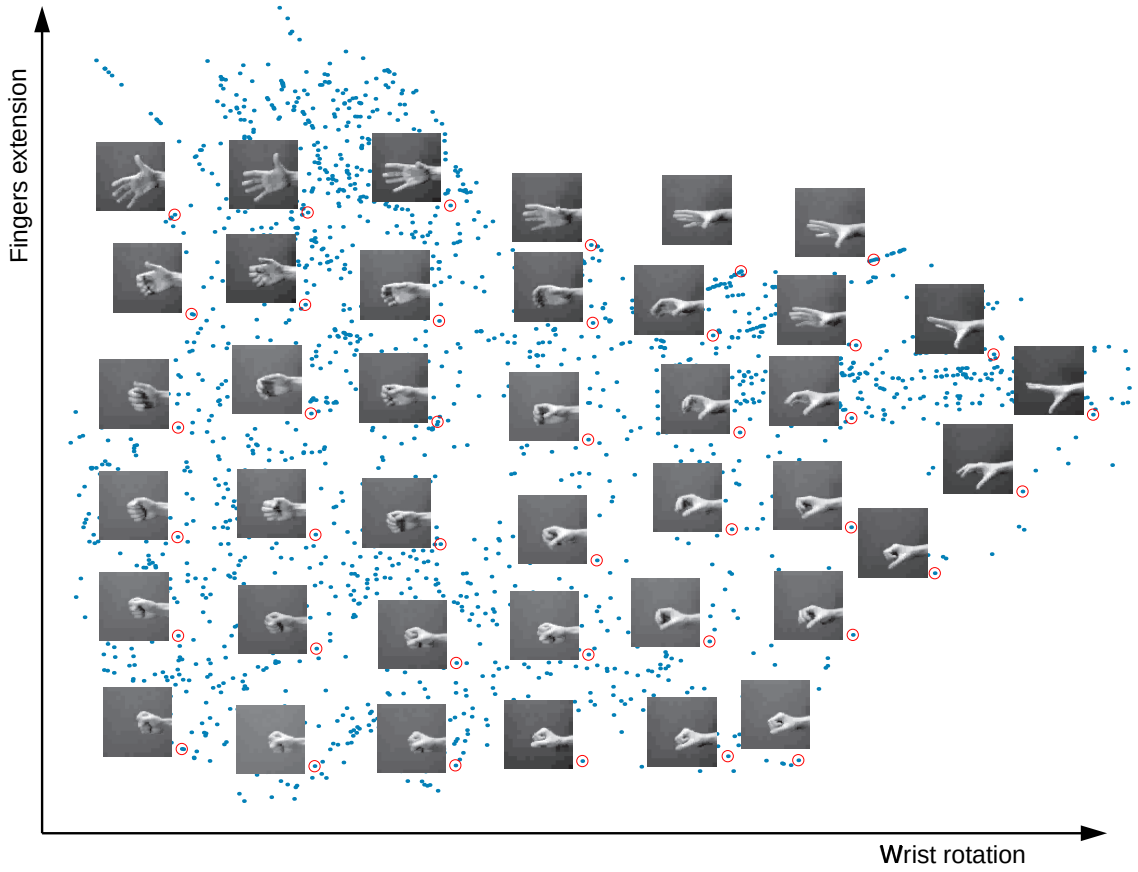


Figure 6.28: Isomap applied to images of a hand making a series of opening and closing movements at different wrist orientations. The recovered coordinate axes map approximately onto the distinct underlying degrees of freedom: wrist rotation (x-axis) and finger extension (y-axis).

distances defined in the Euclidean metric. The recovered coordinate axes map approximately onto the distinct underlying degrees of freedom: wrist rotation (x-axis) and finger extension (y-axis).

The next data set are tree counts in 1-hectare plots in the Barro Colorado Island Condit et al. [2002]. The observations are 50 plots of 1 hectare with counts of trees on each plot. There are 225 tree species for which full Latin names are used. The data give the numbers of trees that have at least 10 cm in diameter at breast height (1.3 m above the ground) in each one hectare square of forest. Within each one hectare square, all individuals of all species were tallied and are recorded. The data frame contains only the Barro Colorado Island subset of the original data. The quadrants are located in a regular grid. Fig. 6.29 shows Isomap results for this data set. For comparison also a standard multidimensional scaling result is given. The spanning tree is depicted in each graph.

Fig. 6.30 shows Isomap results for the multiple tissue data set with different parameter settings. We selected the 101 features with largest variance and then applied Isomap to these data. The results are not as good as with other methods because the observations seem not to be located on a manifold in the high-dimensional space. That was to be expected.

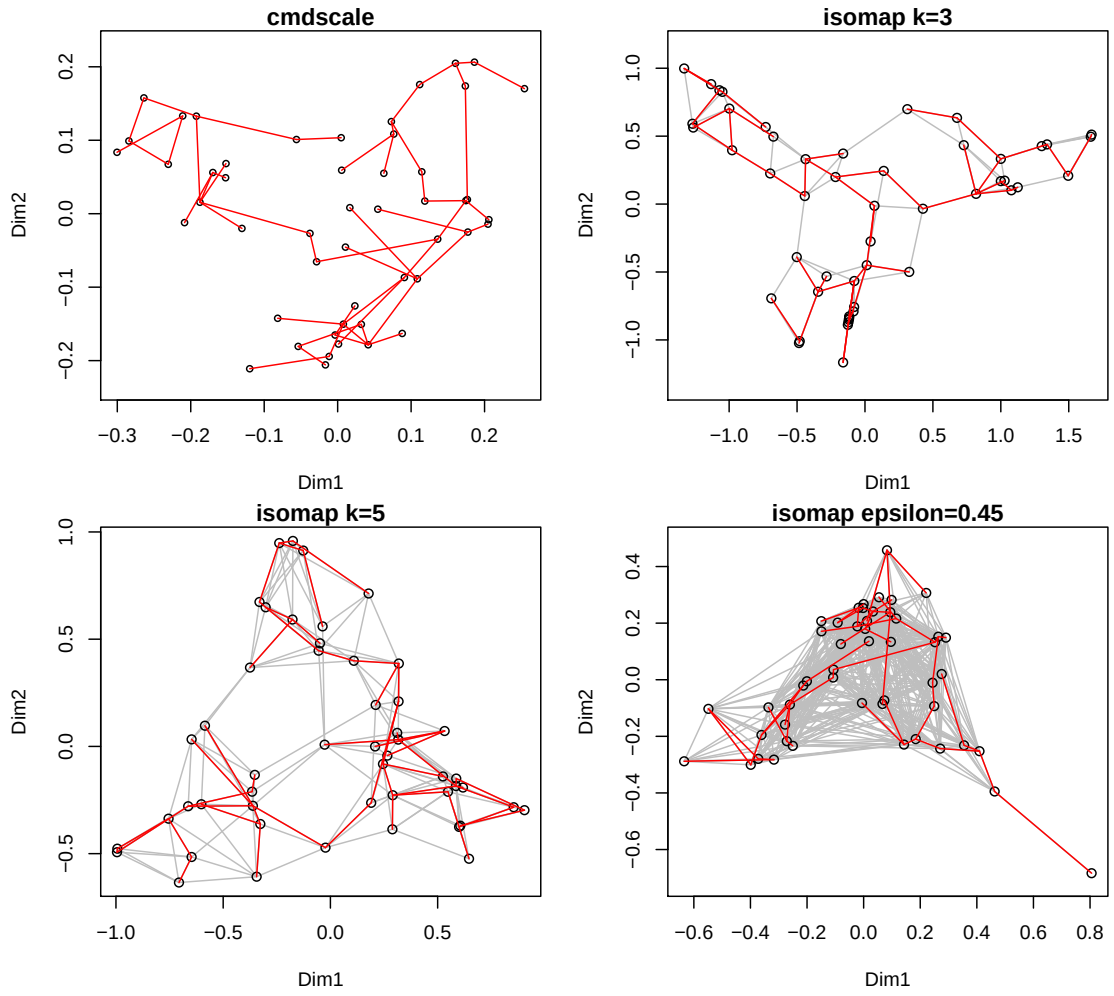


Figure 6.29: Isomap applied to Barro Colorado Island tree count. The upper left panel gives the result of standard multidimensional scaling whereas the other panels give Isomap results with different parameters. The spanning tree of the graphs is given in red.

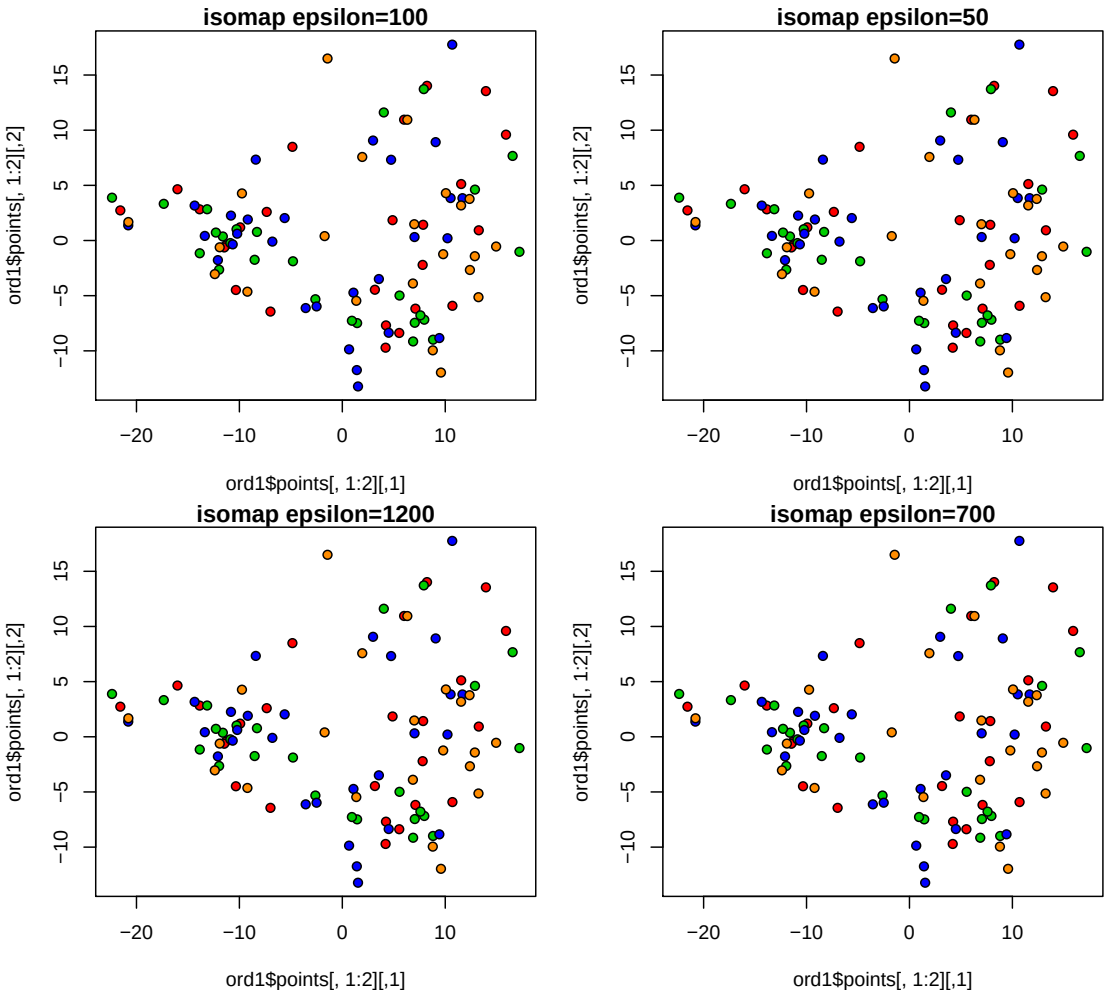


Figure 6.30: Isomap applied to multiple tissues data, where different parameter setting have been tried.

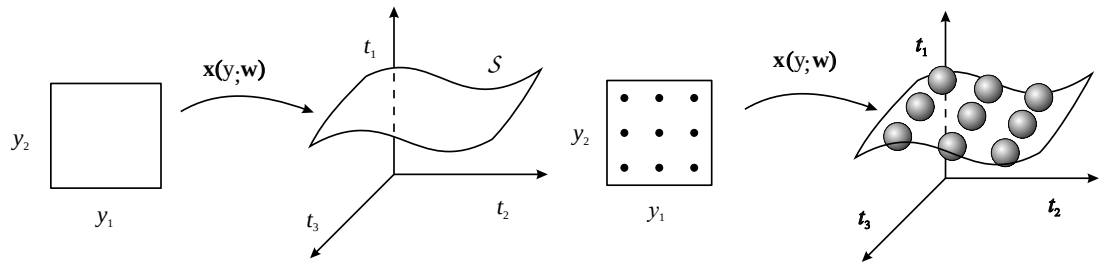


Figure 6.31: Left: The non-linear function $\boldsymbol{x}(\boldsymbol{y}; \boldsymbol{w})$ defines a manifold $\mathcal{S}$ embedded in data space given by the image of the latent-variable space under the mapping $\boldsymbol{y} \longrightarrow \boldsymbol{x}$. Right: The distribution $p(\boldsymbol{y})$ is a superposition of delta function which are located on nodes of a regular grid. The delta functions are mapped to Gaussians in the observation space. Figures from Bishop et al. [1998].

6.6 The Generative Topographic Mapping

6.6.1 The Method

The *generative topographic mapping* (GTM) Bishop et al. [1998] is a non-linear latent variable model where the parameters are found by maximizing the likelihood by the expectation maximization algorithm. GTM is an alternative to SOMs and overcomes the disadvantages of SOMs as a generative model.

Factor analysis as a generative model is a linear transformation from the latent space (the factor space) to the space of observations. GTM is similar to factor analysis as is also maps from the latent space to the observations space. In contrast to factor analysis, the mapping is nonlinear.

Latent variables $\boldsymbol{y} \in \mathbb{R}^l$ are mapped to observations $\boldsymbol{x} \in \mathbb{R}^m$ with $m > l$ (see Fig. 6.31). A distribution $p(\boldsymbol{y})$ is defined on the latent-variable space in order to construct a distribution $p(\boldsymbol{x} \mid \boldsymbol{w})$ in the observation space. The mapped data points $\boldsymbol{x}$ live in an l-dimensional manifold in the m-dimensional space. Thus, probability masses would vanish in the m-dimensional space. Therefore a Gaussian ball in the m-dimensional space around each mapped data point is defined:

$$p(\boldsymbol{t} \mid \boldsymbol{y}, \boldsymbol{w}, \beta) \;=\; \left(\frac{\beta}{2\pi}\right)^{m/2} \exp -\frac{\beta}{2} \left\|\boldsymbol{x}(\boldsymbol{y}, \boldsymbol{w}) \;-\; \boldsymbol{t}\right\|^2 . \tag{6.26}$$

The distribution in the m-dimensional space is obtained by integrating over all $\boldsymbol{x}$ that contribute to a density at $\boldsymbol{t}$:

$$p(\boldsymbol{t} \mid \boldsymbol{w}, \beta) \;=\; \int p(\boldsymbol{t} \mid \boldsymbol{x}, \boldsymbol{w}, \beta)\, p(\boldsymbol{x})\, d\boldsymbol{x} . \tag{6.27}$$

For data points $\{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n\}$, the log likelihood is

$$\log \mathcal{L} \;=\; \sum_{i=1}^{n} \ln p(\boldsymbol{t}_i \mid \boldsymbol{w}, \beta) . \tag{6.28}$$

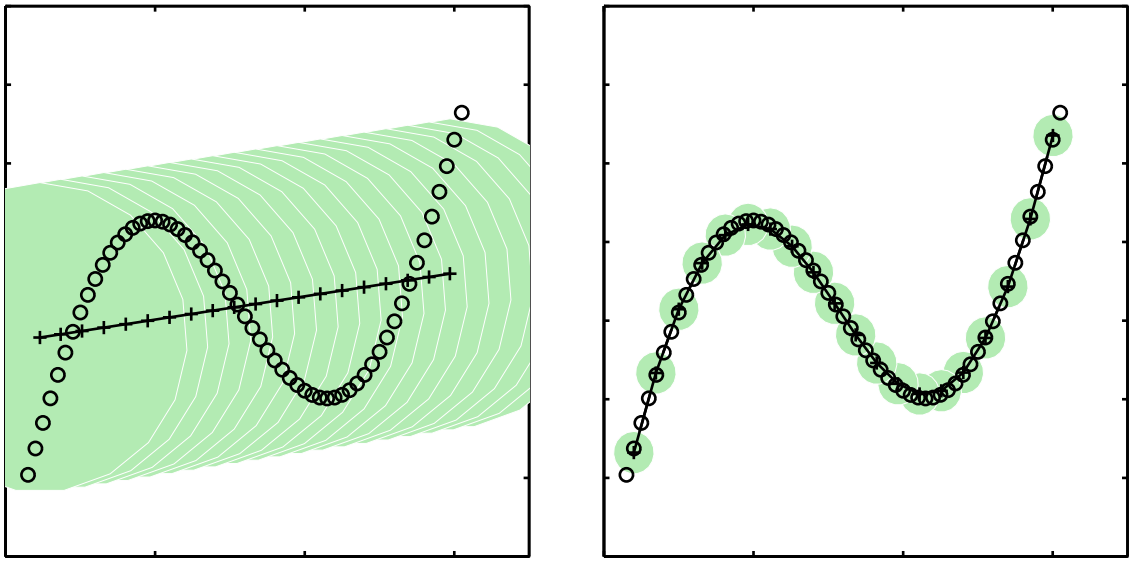


Figure 6.32: Example for GTM. Results from a toy problem involving data ("o") generated from a 1-dimensional curve embedded in 2 dimensions, together with the projected latent points ("+") and their Gaussian noise distributions (filled circles). The initial configuration, determined by principal component analysis, is shown on the left, and the converged configuration, obtained after 15 iterations of EM, is shown on the right. Figure from Bishop et al. [1998].

The distribution $p(\boldsymbol{y})$ of the latent variables is a sum of delta functions located at the nodes of a regular grid in latent space:

$$p(\boldsymbol{y}) \;=\; \frac{1}{L}\sum_{j=1}^{L}\delta(\boldsymbol{y} \;-\; \boldsymbol{y}_j)\;. \tag{6.29}$$

It follows that

$$p(\boldsymbol{t} \mid \boldsymbol{w}, \beta) \;=\; \frac{1}{L}\sum_{j=1}^{L} p(\boldsymbol{t} \mid \boldsymbol{y}_j, \boldsymbol{w}, \beta)\;, \tag{6.30}$$

which is a kernel density estimate or constraint Gaussian mixture model in the m-dimensional space but with the centers mapped from an l-dimensional space.

6.6.2 Examples

There is an R package `gtm` but it was moved out of CRAN and is no longer supported. It is still in the CRAN archive `http://cran.at.r-project.org/contrib/main/Archive/gtm/`.

6.7 t-Distributed Stochastic Neighbor Embedding

6.7.1 The Method

t-distributed stochastic neighbor embedding (t-SNE) vanderMaaten and Hinton [2008] models each high-dimensional observations by a two- or three-dimensional representation in such a way that similar observations are represented by nearby projections and dissimilar observations are represented by distant representations. Again the neighborhood relation should be preserved.

For *stochastic neighbor embedding* (SNE) the similarity of data point $\boldsymbol{x}_j$ to data point $\boldsymbol{x}_i$ is the conditional probability, $p_{j|i}$, that $\boldsymbol{x}_i$ would pick $\boldsymbol{x}_j$ as its neighbor. Neighbors are picked in proportion to their probability density under a Gaussian centered at $\boldsymbol{x}_i$. For observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, we obtain

$$p_{j|i} \;=\; \frac{\exp(-\|\boldsymbol{x}_i \;-\; \boldsymbol{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\boldsymbol{x}_i \;-\; \boldsymbol{x}_k\|^2/2\sigma_i^2)} \,. \tag{6.31}$$

For the low-dimensional projections $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ of the observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, a similar conditional probability is computed and denoted by $q_{j|i}$:

$$q_{j|i} \;=\; \frac{\exp(-\|\boldsymbol{y}_i \;-\; \boldsymbol{y}_j\|^2)}{\sum_{k\neq i}\exp(-\|\boldsymbol{y}_i \;-\; \boldsymbol{y}_k\|^2)} \,, \tag{6.32}$$

where the variance is $1/\sqrt{2}$ and $q_{i|i} = 0$.

The objective is the Kullback-Leibler divergence between the distribution P and the distribution Q:

$$KL(P||Q) \;=\; \sum_{i\neq j} p_{ij}\log\frac{p_{ij}}{q_{ij}} \,. \tag{6.33}$$

The objective is minimized by gradient descent.

The objective for the SNE approach is difficult to optimize and a problem called "crowding problem" appears. To illustrate the "crowding problem", in ten dimensions, it is possible to have 11 data points that are mutually equidistant but there is no way to model this faithfully in a two-dimensional map. The "crowding problem" is that the area of the two-dimensional map that is available to accommodate moderately distant data points will not be nearly large enough compared with the area available to accommodate nearby data points in the observation space. Hence, if we want to model the small distances accurately in the map, most of the points that are at a moderate distance from data point i will have to be placed much too far away in the two-dimensional map.

These problems motivated the development of t-distributed stochastic neighbor embedding, which has two differences to SNE:

- the objective of the SNE is symmetrized which results in simpler gradients,
- the objective uses Student's t-distribution which is a heavy-tailed distribution. The heavy-tails reduce the crowding problem and simplify the optimization problem.

The symmetry is achieved by setting

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{6.34}$$

and using these values in the objective.

Using the Student's t-distribution, we obtain

$$q_{ij} = \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l))^{-1}} . \tag{6.35}$$

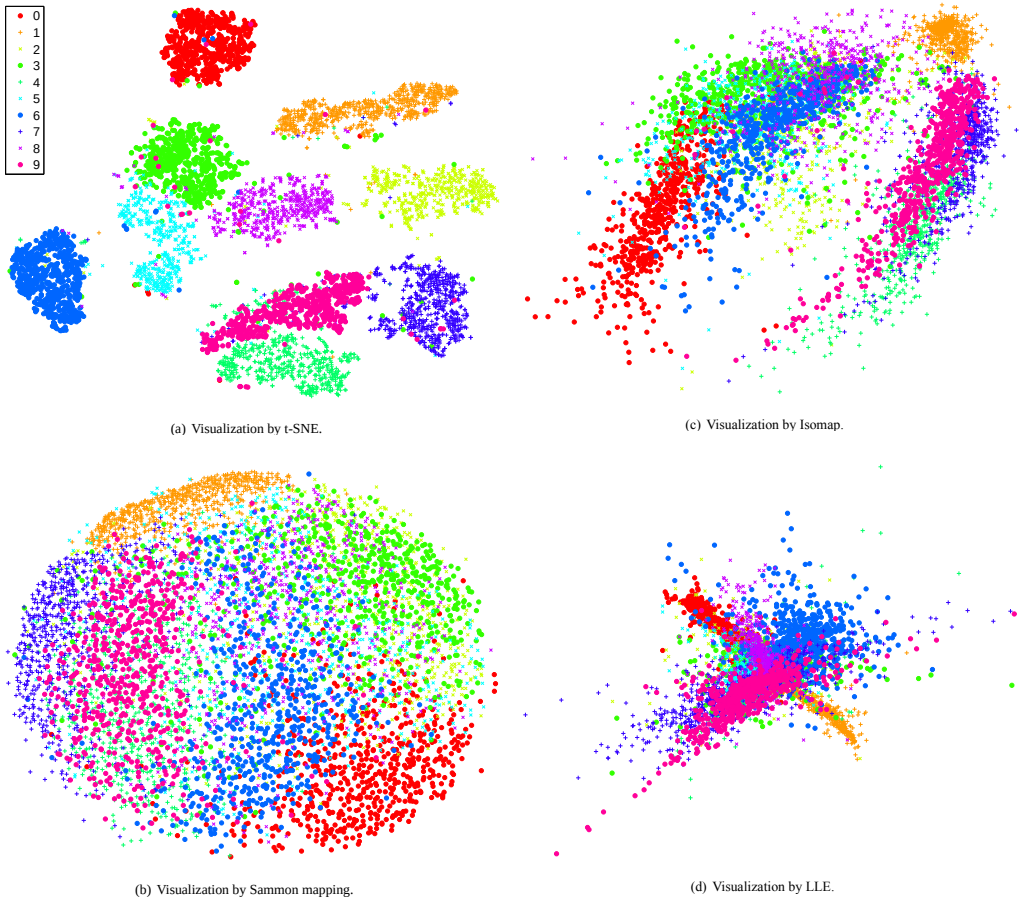
Optimization is still performed via gradient descent.

6.7.2 Examples

Fig. 6.33 shows visualizations of 6,000 handwritten digits from the MNIST data set. t-SNE is compared to Sammon's mapping, Isomap, and LLE. Fig. 6.34 shows visualizations of faces obtained from the Olivetti data base. Again, t-SNE is compared to Sammon's mapping, Isomap, and LLE. Fig. 6.35 shows visualizations of the COIL-20 data set by t-SNE, Sammon's mapping, Isomap, and LLE.

We revisit the iris data set. Fig. 6.36 shows visualizations of the iris data set by t-SNE. As comparison the down-projection onto two dimensions by PCA is again given in Fig. 6.37.

Finally we apply t-SNE to the multiple tissue data set. We selected the 101 features with largest variance and then called t-SNE and plotted the result. Fig. 6.38 shows the t-SNE down-projection with `perplexity=50` and Fig. 6.39 with `perplexity=30`. The results are not as good as with other methods because the observations are not located on a 2-dimensional manifold.



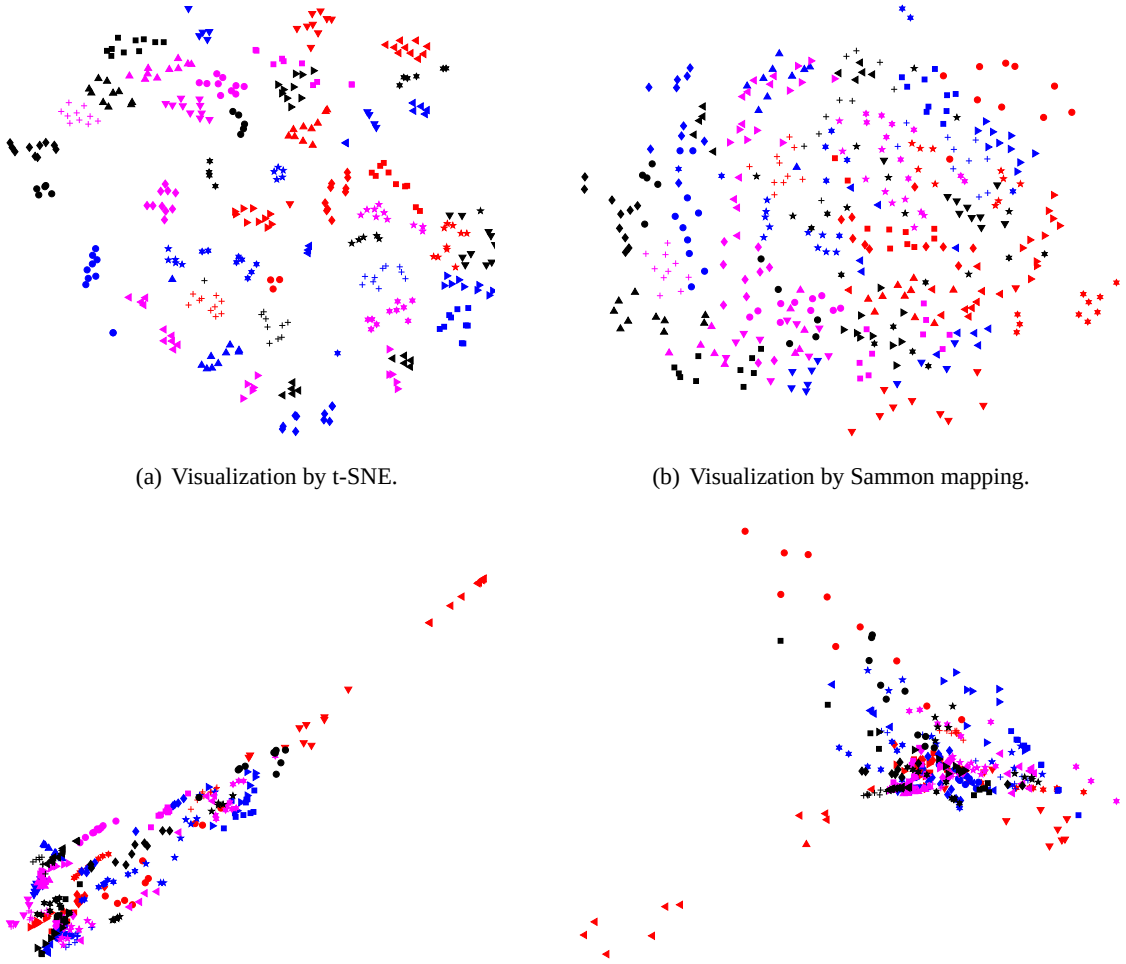
(a) Visualization by t-SNE.

(c) Visualization by Isomap.

(b) Visualization by Sammon mapping.

(d) Visualization by LLE.

Figure 6.33: Visualizations of 6,000 handwritten digits from the MNIST data set by t-SNE. Figure from vanderMaaten and Hinton [2008].



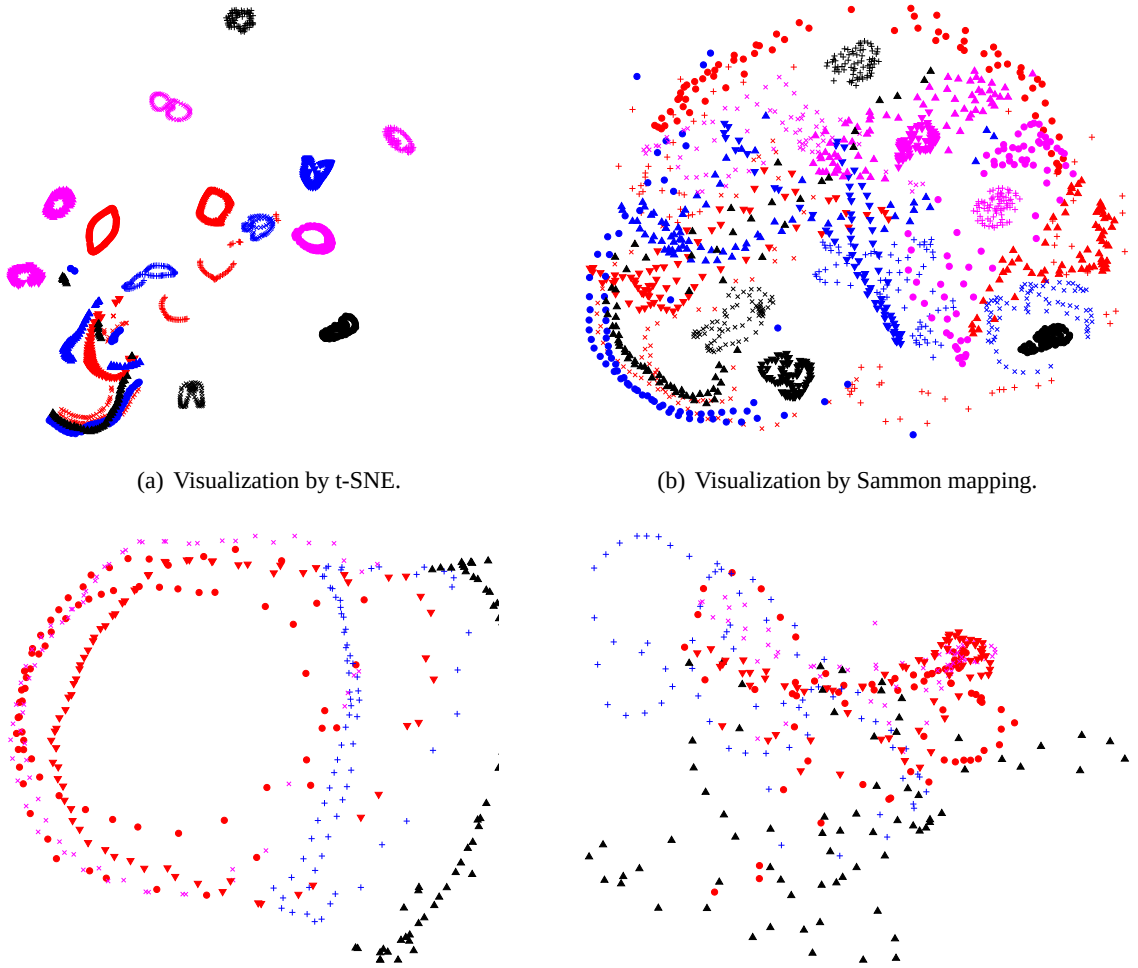
(a) Visualization by t-SNE.

(b) Visualization by Sammon mapping.

(c) Visualization by Isomap.

(d) Visualization by LLE.

Figure 6.34: Visualizations of the Olivetti faces data set by t-SNE. Figure from vanderMaaten and Hinton [2008].



(a) Visualization by t-SNE.

(b) Visualization by Sammon mapping.

(c) Visualization by Isomap.

(d) Visualization by LLE.

Figure 6.35: Visualizations of the COIL-20 data set. Figure from vanderMaaten and Hinton [2008].

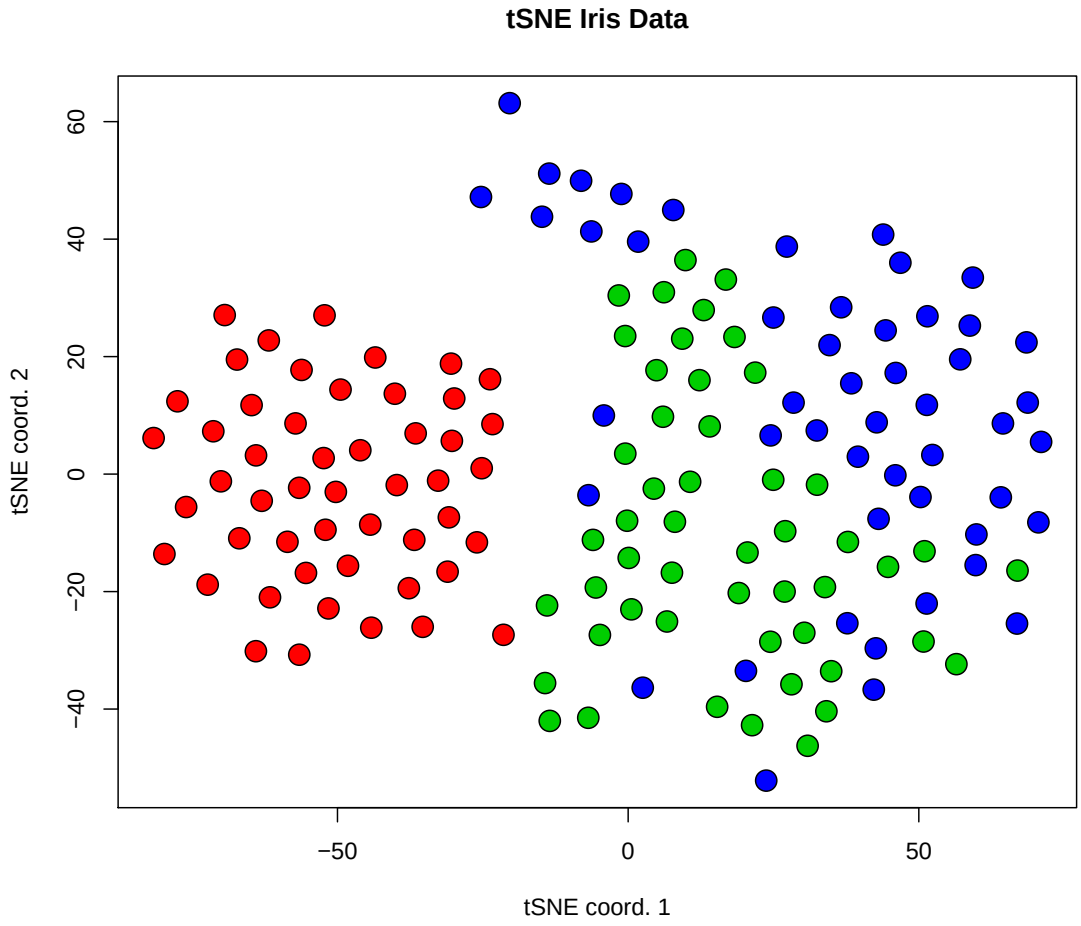


Figure 6.36: Down-projection of the iris data set to two dimensions by t-SNE.

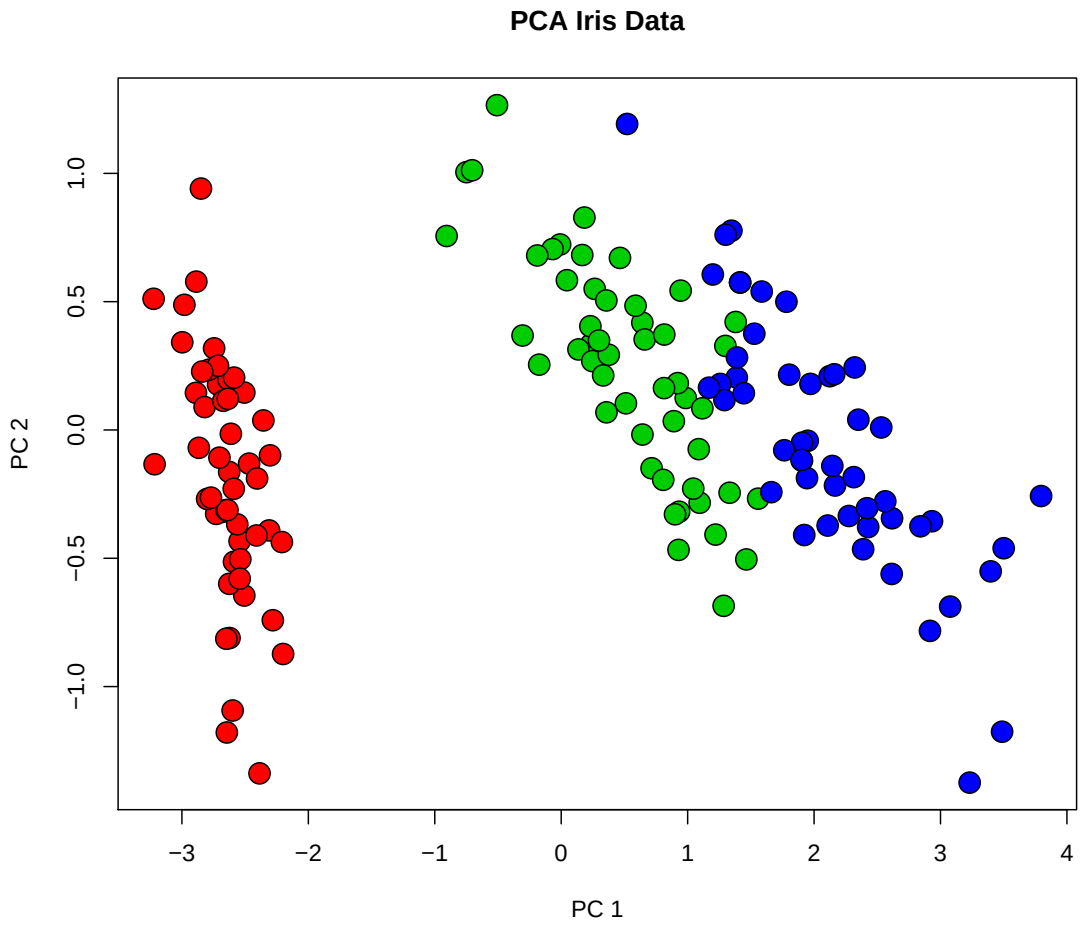


Figure 6.37: Down-projection of the iris data set to two dimensions by PCA.

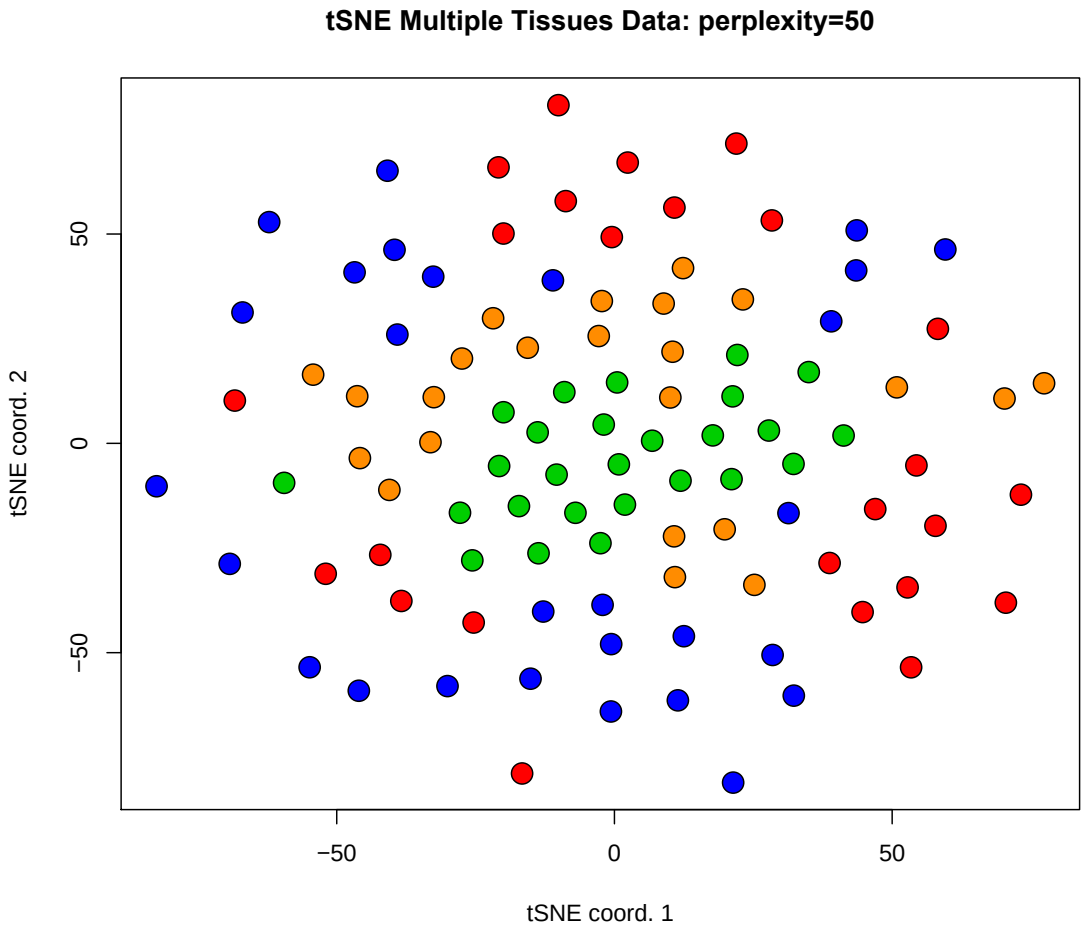


Figure 6.38: Down-projection of the multiple tissue data onto a two-dimensional space by t-SNE. We selected the 101 features with largest variance and set perplexity to 50.

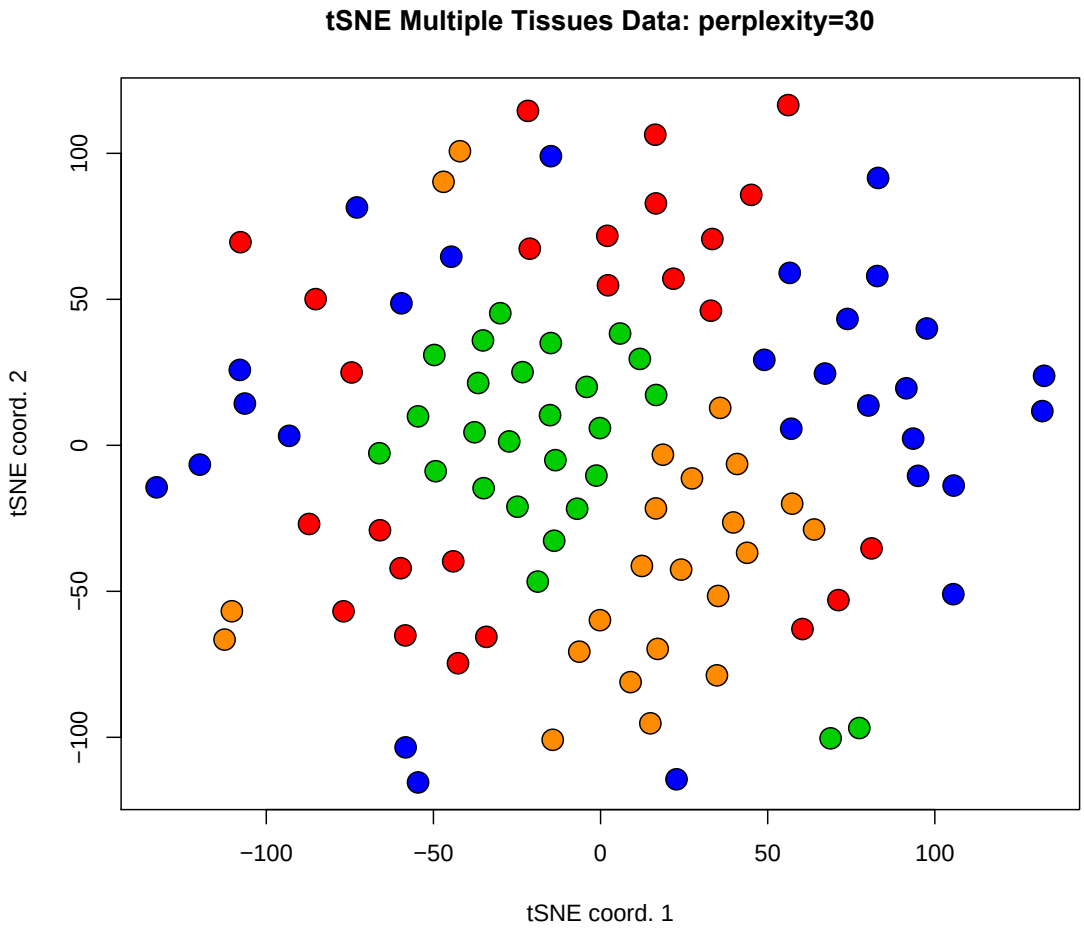


Figure 6.39: Down-projection of the multiple tissue data onto a two-dimensional space by t-SNE. We selected the 101 features with largest variance and set perplexity to 30.

6.8 Self-Organizing Maps

6.8.1 The Method

A method which is similar to multidimensional scaling is the *Self-Organizing Map* (SOM) also called *Kohonen map* Kohonen [1982, 1988, 1990, 1995b], Ritter et al. [1992, 1991], Obermayer et al. [1992], Erwin et al. [1992]. SOMs comprise two objectives: clustering (see next subsection) and down-projecting. Data points $\boldsymbol{x} \in \mathbb{R}^m$ are clustered and down-projected to points $\boldsymbol{y} \in \mathbb{R}^l$ with $m > l$. The $\boldsymbol{y}$ are clustered onto finite many $\boldsymbol{y}_k$.

For SOMs the objective function **cannot** always be expressed as a single scalar function like an energy or an error function. Scalar objectives are important to derive learning algorithms based on optimizing this function and to compare solutions. The objective of SOMs is a scalar function for discrete input spaces and for discrete neighborhood functions otherwise the objective function must be expressed as a vector valued potential function Kohonen [1995b], Cottrell et al. [1995], Ritter et al. [1992, 1991], Erwin et al. [1992]. The lack of a scalar objective function is one of the major drawbacks of SOMs, because models cannot be compared, overfitting not detected, and stopping of training is difficult to determine, and the quality of the solution is hard to assess.

In most applications, the $\boldsymbol{y}_k$ equidistantly fill a hypercube in $\mathbb{R}^l$. For each $\boldsymbol{y}_k$ there exists an associated $\boldsymbol{w}_k \in \mathbb{R}^m$ representing the cluster center in the data space. These $\boldsymbol{w}_k$ are the parameters of the SOM.

The goal now is to find cluster centers $\boldsymbol{w}_k$ (parameters) such that for data points $\boldsymbol{x}$, which are neighbors in $\mathbb{R}^m$, their projections $\boldsymbol{y}$ are also neighbors in $\mathbb{R}^l$. The goal is to down-project but preserve the neighborhood relation which gave these methods also the name "topologically ordered maps" (TOMs).

The learning can be done on-line, that is each new data point $\boldsymbol{x}$ leads to an update of the $\boldsymbol{w}_k$. The update rule is

$$k \;=\; \arg\max_s \; \boldsymbol{x}^T \boldsymbol{w}_s \tag{6.36}$$

$$(\boldsymbol{w}_t)^{\text{new}} \;=\; \boldsymbol{w}_t \;+\; \eta\, \delta\left(\|\boldsymbol{y}_t \;-\; \boldsymbol{y}_k|\|\right)\; (\boldsymbol{x} \;-\; \boldsymbol{w}_t)\;, \tag{6.37}$$

where η is the learning rate which also depends on the iteration number and is annealed and δ is the "window function" which is largest for $\boldsymbol{y}_t \;=\; \boldsymbol{y}_k$ and is decreasing with the distance to $\boldsymbol{y}_k$.

SOMs have serious disadvantages Kohonen [1995b], Bishop et al. [1998]:

- the absence of a cost function
- the lack of a theoretical basis for choosing learning rate parameter schedules and neighborhood parameters to ensure topographic ordering
- the absence of any general proofs of convergence
- the fact that the model does not define a probability density.

These problems can all be traced back to the heuristic origins of the SOM algorithm.

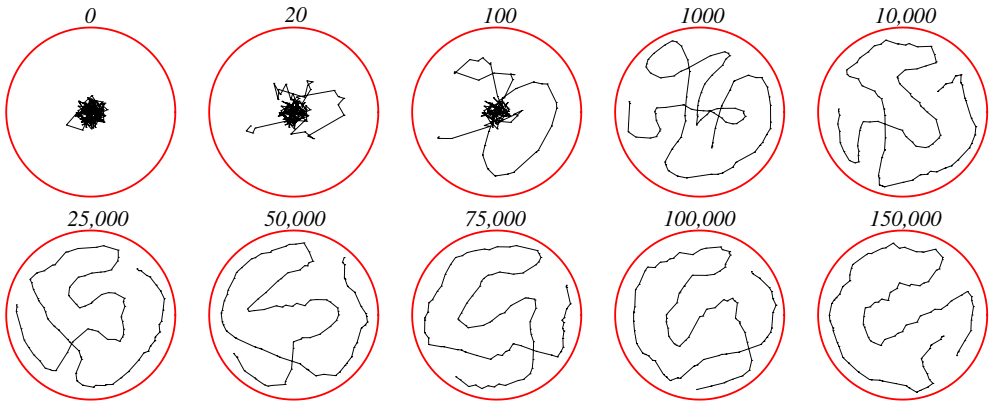


Figure 6.40: Self-organizing map example of a one-dimensional representation of a two-dimensional space. Copyright © 2001 John Wiley & Sons, Inc.

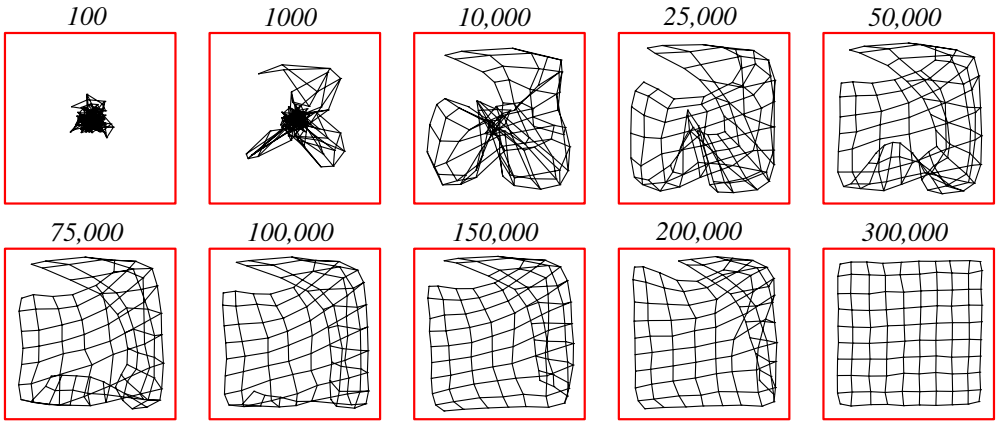


Figure 6.41: Self-organizing map example for mapping from a square data space to a square (grid) representation space. Copyright © 2001 John Wiley & Sons, Inc.

6.8.2 Examples

We show some examples of self-organizing maps. Fig. 6.40 shows a self-organizing map example of a one-dimensional representation of a two-dimensional space. A self-organizing map example for mapping from a square data space to a square (grid) representation space is given in Fig. 6.41. Fig. 6.42 shows the example of Fig. 6.41 but with different initialization. Kinks in the map do not vanish even if more patterns are presented – that is a local minimum. Fig. 6.43 again shows the example of Fig. 6.41 but with a non-uniformly sampling: the density at the center was higher than at the border.

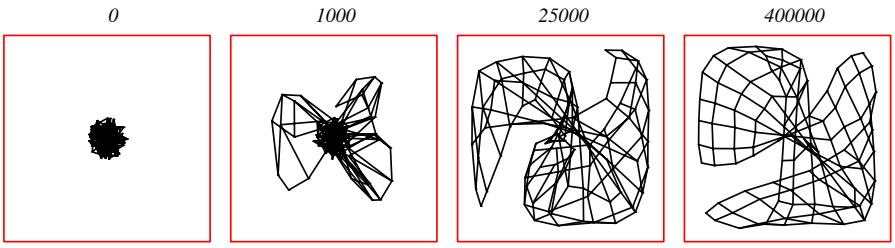


Figure 6.42: Self-organizing map example from Fig. 6.41 but with different initialization. Kinks in the map do not vanish even if more patterns are presented – that is a local minimum. Copyright © 2001 John Wiley & Sons, Inc.

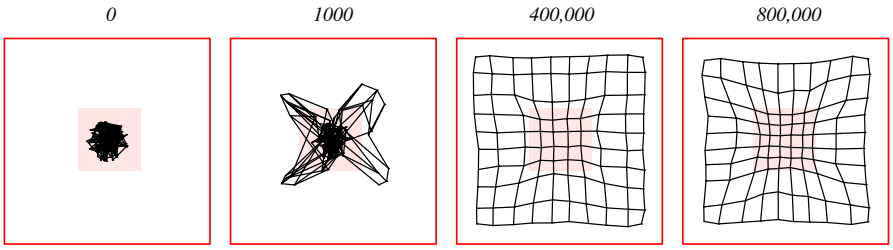


Figure 6.43: Self-organizing map example from Fig. 6.41 but with a non-uniformly sampling: the density at the center was higher than at the border. Copyright © 2001 John Wiley & Sons, Inc.

Chapter 7

Clustering

One of the best known and most popular unsupervised learning techniques is clustering. "Clusters" in the data are regions where observations group together or, in other words, regions of high data density. Often these clusters are observations which stem from one "prototype" via noise perturbations. The prototype may represent a certain situation in the real world which is repeated but has slightly different environments or is measured with noise, so that, the feature values for this situation differ for each occurrence.

Clustering extracts structures in the data and can identify new data classes which were unknown so far. An important application of clustering is data visualization, where in some cases both down-projection and clustering are combined, e.g. as for self-organizing maps which were previously considered. If observations are represented by their prototypes then clustering is a data compression method called "vector quantization".

7.1 Mixture Models

7.1.1 The Method

Since clusters are regions of high data density, density estimators which locally assign a component can be used for clustering. A component represents a cluster. Component j out of l components has parameters like location $\boldsymbol{\mu}_j$ and width or shape $\boldsymbol{\Sigma}_j$. For every mixture model the component j has a weight w_j that gives the local probability mass. A well known example is the mixtures of Gaussians (MoG) model Pearson [1894], Hasselblad [1966], Duda and Hart [1973].

In a generative model framework w_j is the probability $p(j)$ of choosing component j, the value $\boldsymbol{\theta}_j$ gives the parameters of the local component j, which has density $p(\boldsymbol{x} \mid j, \boldsymbol{\theta}_j)$. If we summarize all parameters $\boldsymbol{\theta}_j$ and $\boldsymbol{w}$ in the parameter vector $\boldsymbol{\theta}$, then we obtain the generative model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) \;=\; \sum_{j=1}^{l} p(j)\; p(\boldsymbol{x} \mid j, \boldsymbol{\theta}_j)\;. \tag{7.1}$$

For clustering, Bayes' formula can be used:

$$p(j \mid \boldsymbol{x}, \boldsymbol{\theta}) \;=\; \frac{p(\boldsymbol{x} \mid j, \boldsymbol{\theta}_j)\; p(j)}{p(\boldsymbol{x} \mid \boldsymbol{\theta})}\;. \tag{7.2}$$

Observation $\boldsymbol{x}$ is assigned to the component j with largest posterior $p(j \mid \boldsymbol{x}, \boldsymbol{\theta})$.

Before an observation was seen, each component or cluster has the prior probability $p(j) = w_j$ that a data point is drawn from it. After observing data $\boldsymbol{x}$ some clusters may be more or less probable of having produced $\boldsymbol{x}$, therefore the prior probability $p(j)$ changes to the posterior $p(j \mid \boldsymbol{x})$. The posterior tells how likely $\boldsymbol{x}$ was produced by cluster j.

Mixture models can contain other components than Gaussian distributions. For example, Poisson components were successfully used to estimate copy numbers in next generation sequencing data Klambauer et al. [2012]. Another application of mixture models was a mixture of negative binomials in order to estimate differential expressed transcripts in next generation sequencing without knowing the conditions and without having replicates Klambauer et al. [2013].

The log-likelihood is

$$\ln \mathcal{L} \;=\; \sum_{i=1}^{n} \ln p(\boldsymbol{x}_i \mid \boldsymbol{\theta}) \;. \tag{7.3}$$

The derivative of the log-likelihood is with respect to $\boldsymbol{\theta}_j$, the parameters of component j, is:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}_j} \ln \mathcal{L} \;=\; \sum_{i=1}^{n} \frac{1}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} \sum_{k=1}^{l} p(k) \; \frac{\partial}{\partial \boldsymbol{\theta}_j} p(\boldsymbol{x}_i \mid k, \boldsymbol{\theta}_k) \;= \\ \sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\theta}_j) \frac{\partial}{\partial \boldsymbol{\theta}_j} \ln p(\boldsymbol{x}_i \mid j, \boldsymbol{\theta}_j) \;, \end{aligned} \tag{7.4}$$

where we used Bayes' formula

$$p(j \mid \boldsymbol{x}_i, \boldsymbol{\theta}_j) \;=\; \frac{p(\boldsymbol{x}_i \mid j, \boldsymbol{\theta}_j) \; p(j)}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta})} \;. \tag{7.5}$$

The derivative of the log-likelihood of the model with respect to the parameters of the j-th component is the posterior expectation of component j of the derivative of the log-likelihood of component j.

7.1.2 Mixture of Gaussians

We will now consider mixture of Gaussian (MoG), where $\boldsymbol{\theta}_j = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ and

$$p(\boldsymbol{x}_i \mid j, \boldsymbol{\theta}_j) \;=\; p(\boldsymbol{x}_i \mid j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \;\sim\; \mathcal{N}\left(\boldsymbol{\mu}_j \;,\; \boldsymbol{\Sigma}_j\right) \;. \tag{7.6}$$

The model is

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) \;=\; \sum_{j=1}^{l} w_j \, \mathcal{N}\left(\boldsymbol{\mu}_j \;,\; \boldsymbol{\Sigma}_j\right) \tag{7.7}$$

$$\sum_{j=1}^{l} w_j \;=\; 1 \tag{7.8}$$

$$w_j \;\geq\; 0 \tag{7.9}$$

$$\mathcal{N}\left(\boldsymbol{\mu}_j \;,\; \boldsymbol{\Sigma}_j\right)(\boldsymbol{x}) \;=\; (2\pi)^{-m/2} \left|\boldsymbol{\Sigma}_j\right|^{-1/2} \exp\left(-\frac{1}{2} \; (\boldsymbol{x} \;-\; \boldsymbol{\mu}_j)^T \; \boldsymbol{\Sigma}_j^{-1} \, (\boldsymbol{x} \;-\; \boldsymbol{\mu}_j)\right) \;. \tag{7.10}$$

Exponential distributions like Gaussians are convenient because in Eq. (7.4) the logarithm inverts the exponential function:

$$\ln p(\boldsymbol{x} \mid j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \; = \tag{7.11}$$
$$-\; \frac{m}{2}\, \ln(2\pi) \;-\; \frac{1}{2}\, \ln|\boldsymbol{\Sigma}_j| \;-\; \frac{1}{2}\, (\boldsymbol{x} \;-\; \boldsymbol{\mu}_j)^T \; \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x} \;-\; \boldsymbol{\mu}_j)$$

which gives for the derivatives

$$\frac{\partial}{\partial \boldsymbol{\mu}_j} \ln p(\boldsymbol{x} \mid j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \; = \; \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x} \;-\; \boldsymbol{\mu}_j) \tag{7.12}$$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_j} \ln p(\boldsymbol{x} \mid j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \; = \tag{7.13}$$
$$\frac{1}{2}\, \left(\boldsymbol{\Sigma}_j^T\right)^{-1} \;+\; \frac{1}{2}\, \boldsymbol{\Sigma}_j^{-T} (\boldsymbol{x} \;-\; \boldsymbol{\mu}_j)\; (\boldsymbol{x} \;-\; \boldsymbol{\mu}_j)^T\, \boldsymbol{\Sigma}_j^{-T}\,.$$

Here also the EM-algorithm can be used where the hidden parameters $p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ must be estimated to evaluate Eq. (7.4).

The EM-algorithm is

E-step: (7.14)

$$p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \; = \; \frac{w_j\, \mathcal{N}\,(\boldsymbol{\mu}_j \;,\; \boldsymbol{\Sigma}_j)\,(\boldsymbol{x}_i)}{\sum_{t=1}^{l} w_t\, \mathcal{N}\,(\boldsymbol{\mu}_t \;,\; \boldsymbol{\Sigma}_t)\,(\boldsymbol{x}_i)}$$

M-step: (7.15)

$$\boldsymbol{w}_j^{\text{new}} \; = \; \frac{1}{n} \sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

$$\boldsymbol{\mu}_j^{\text{new}} \; = \; \frac{\sum_{i=1}^{n}\; p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\; \boldsymbol{x}_i}{\sum_{i=1}^{n}\; p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{7.16}$$

$$\boldsymbol{\Sigma}_j^{\text{new}} \; = \; \frac{\sum_{i=1}^{n}\; p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\; (\boldsymbol{x}_i \;-\; \boldsymbol{\mu}_j)\; (\boldsymbol{x}_i \;-\; \boldsymbol{\mu}_j)^T}{\sum_{i=1}^{n}\; p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}\,. \tag{7.17}$$

In order to avoid too small variances and near zero eigenvalues of $\boldsymbol{\Sigma}_j$ the mixture of Gaussian can be optimized by a maximum a posterior approach.

A proper prior for the covariance $\boldsymbol{\Sigma}$ is the Wishart density $\mathcal{W}(\boldsymbol{\Sigma}^{-1} \mid \alpha, \boldsymbol{\Psi})$, a proper prior for the weighting factors w_j is a Dirichlet density $\mathcal{D}(\boldsymbol{w} \mid \gamma)$, and a proper prior for the mean values $\boldsymbol{\mu}$ is a Gaussian $\mathcal{N}\left(\boldsymbol{\mu} \mid \boldsymbol{\nu}, \eta^{-1}\boldsymbol{\Sigma}\right)$:

$$\mathcal{W}(\boldsymbol{\Sigma}^{-1} \mid \alpha, \boldsymbol{\Psi}) \; = \; c(\alpha, \boldsymbol{\Psi})\; \left|\boldsymbol{\Sigma}^{-1}\right|^{\alpha-(m+1)/2}\; \exp\left(-\text{tr}\left(\boldsymbol{\Psi}\; \boldsymbol{\Sigma}^{-1}\right)\right) \tag{7.18}$$

$$\mathcal{D}(\boldsymbol{w} \mid \gamma) \; = \; c(\gamma) \prod_{j=1}^{l} w_j^{\gamma-1} \tag{7.19}$$

$$\mathcal{N}\left(\boldsymbol{\mu} \mid \boldsymbol{\nu}, \eta^{-1}\boldsymbol{\Sigma}\right) \; = \tag{7.20}$$
$$(2\pi)^{-m/2}\, \left|\eta^{-1}\, \boldsymbol{\Sigma}_j\right|^{-1/2} \exp\left(-\frac{\eta}{2}\; (\boldsymbol{\mu} \;-\; \boldsymbol{\nu})^T\; \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{\mu} \;-\; \boldsymbol{\nu})\right)\,,$$

where $\alpha > (m-1)/2$ and $c(\gamma)$ as well as $c(\alpha, \boldsymbol{\Psi})$ are normalizing constants. The operator "tr" is the trace operator.

The expectation-maximization algorithm is now

E-step: (7.21)

$$p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \;=\; \frac{w_j \, \mathcal{N}\left(\boldsymbol{\mu}_j \;,\; \boldsymbol{\Sigma}_j\right)(\boldsymbol{x}_i)}{\sum_{t=1}^{l} w_t \, \mathcal{N}\left(\boldsymbol{\mu}_t \;,\; \boldsymbol{\Sigma}_t\right)(\boldsymbol{x}_i)}$$

M-step: (7.22)

$$w_j^{\text{new}} \;=\; \frac{\sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \;+\; \gamma \;-1}{n \;+\; l \;(\gamma \;-\; 1)}$$

$$\boldsymbol{\mu}_j^{\text{new}} \;=\; \frac{\sum_{i=1}^{n} \; p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \; \boldsymbol{x}_i \;+\; \eta \; \boldsymbol{\nu}_j}{\sum_{i=1}^{n} \; p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \;+\; \eta} \tag{7.23}$$

$$\begin{aligned} \boldsymbol{\Sigma}_j^{\text{new}} \;=\; & \left(\sum_{i=1}^{n} \; p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \; (\boldsymbol{x}_i \;-\; \boldsymbol{\mu}_j) \; (\boldsymbol{x}_i \;-\; \boldsymbol{\mu}_j)^T \;+ \right. \\ & \left. \eta \; (\boldsymbol{\nu}_j \;-\; \boldsymbol{\mu}_j) \; (\boldsymbol{\nu}_j \;-\; \boldsymbol{\mu}_j)^T \;+\; 2 \; \boldsymbol{\Psi} \right) \\ & \left(\sum_{i=1}^{n} \; p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \;+\; 2 \, \alpha \;- m \right)^{-1} . \end{aligned} \tag{7.24}$$

Above formulas are obtained as follows: Since

$$\sum_{j=1}^{l} w_j^{\text{new}} \;=\; 1 \tag{7.25}$$

we obtain as Lagrangian for the constrained optimization problem for the w_j

$$L \;=\; \sum_{i=1}^{n} \sum_{j=1}^{l} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \; \ln w_j^{\text{new}} \;+\; \ln \mathcal{D}(\boldsymbol{w} \mid \gamma) \;-\; \lambda \left(\sum_{j=1}^{l} w_j^{\text{new}} \;-\; 1 \right) . \tag{7.26}$$

Setting the derivative to zero:

$$\begin{aligned} & \frac{\partial L}{\partial w_j} \;=\; \sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \; \left(w_j^{\text{new}}\right)^{-1} \;+\; (\gamma \;-\; 1) \; \left(w_j^{\text{new}}\right)^{-1} \;-\; \lambda \;=\; 0 \\ & \sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \;+\; (\gamma \;-\; 1) \;=\; \lambda \, w_j^{\text{new}} \end{aligned} \tag{7.27}$$

Summing over j gives

$$\sum_{i=1}^{n} \sum_{j=1}^{l} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \;+\; l \, (\gamma \;-\; 1) \;=\; \lambda \sum_{j=1}^{l} w_j^{\text{new}} \tag{7.28}$$

$$n \;+\; l \, (\gamma \;-\; 1) \;=\; \lambda \tag{7.29}$$

We obtain

$$w_j^{\text{new}} \;=\; \frac{\sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \;+\; (\gamma \;-\; 1)}{n \;+\; l\,(\gamma \;-\; 1)} \tag{7.30}$$

For the other parameters we do not have constraints. The gradient of the log-posterior L with respect to $\boldsymbol{\mu}_j$ contains the log-likelihood and the log-prior:

$$\begin{aligned} &\frac{\partial L}{\partial \boldsymbol{\mu}_j} \;=\; \sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\; \boldsymbol{\Sigma}_j^{-1}\; (\boldsymbol{x}_i \;-\; \boldsymbol{\mu}_j) \;+ \\ &\eta\; \boldsymbol{\Sigma}_j^{-1}\; (\boldsymbol{\nu}_j \;-\; \boldsymbol{\mu}_j) \;=\; 0\,. \end{aligned} \tag{7.31}$$

For the gradient with respect to $\boldsymbol{\Sigma}_j$ a trick is applied: the gradient is taken with respect to $\boldsymbol{\Sigma}_j^{-1}$ which also must be zero at the minimum, because

$$\frac{\partial L}{\partial \boldsymbol{\Sigma}_j} \;=\; \frac{\partial L}{\partial \boldsymbol{\Sigma}_j^{-1}}\, \frac{\partial \boldsymbol{\Sigma}_j^{-1}}{\partial \boldsymbol{\Sigma}_j} \;=\; -\,\boldsymbol{\Sigma}_j^{-2} \frac{\partial L}{\partial \boldsymbol{\Sigma}_j^{-1}} \tag{7.32}$$

and the variables $\boldsymbol{\Sigma}_j^{-1}$ fully represent the variables $\boldsymbol{\Sigma}_j$.

We obtain

$$\begin{aligned} &\frac{\partial L}{\partial \boldsymbol{\Sigma}_j^{-1}} \;= \\ &\frac{1}{2} \sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\; \left(\boldsymbol{\Sigma}_j \;-\; (\boldsymbol{x}_i \;-\; \boldsymbol{\mu}_j)\; (\boldsymbol{x}_i \;-\; \boldsymbol{\mu}_j)^T\right) \;+ \\ &\frac{1}{2} \left(\boldsymbol{\Sigma}_j \;-\; \eta\; (\boldsymbol{\nu}_j \;-\; \boldsymbol{\mu}_j)\; (\boldsymbol{\nu}_j \;-\; \boldsymbol{\mu}_j)^T\right) \;+ \\ &\boldsymbol{\Sigma}_j\; (\alpha \;-\; (m \;+\; 1)/2) \;-\; \boldsymbol{\Psi} \;=\; 0\,, \end{aligned} \tag{7.33}$$

where we used

$$\frac{\partial \ln |\boldsymbol{U}|}{\partial \boldsymbol{U}} \;=\; \boldsymbol{U}^{-1} \tag{7.34}$$

for $\boldsymbol{U} \;=\; \boldsymbol{\Sigma}_j^{-1}$.

Note that each component can have a prior so that we would obtain $\boldsymbol{\nu}_j$, η_j, α_j, $\boldsymbol{\Psi}_j$, and γ_j. The update formulas would be similar as above.

Default values for the hyperparameters are

$$\alpha \;=\; \frac{m}{2} \tag{7.35}$$

$$\boldsymbol{\Psi} \;=\; \frac{1}{2}\boldsymbol{I} \;\text{ OR }\; \boldsymbol{\Psi} \;=\; \frac{1}{2}\text{covar}(\boldsymbol{x}) \tag{7.36}$$

$$\gamma \;=\; 1 \tag{7.37}$$

$$\eta \;=\; 0 \tag{7.38}$$

$$\boldsymbol{\nu}_j \;=\; \text{mean}(\boldsymbol{x}) \tag{7.39}$$

A prior on the mean values $\boldsymbol{\mu}$ is in most cases not useful except a preferred region is known.

The posterior $p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ can be used for clustering: $\boldsymbol{x}_i$ belongs to the cluster j for which the posterior is largest.

But also soft clustering is possible: $p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ gives the graded or fuzzy membership of $\boldsymbol{x}_i$ to the cluster j.

7.1.3 Mixture of Poissons

We assume to have count data x and obtain the mixture of Poissons model with $w_j = p(j)$ as:

$$p(x) \;=\; \sum_{j=1}^{l} w_j \, \mathrm{P}(x; \lambda_j) \; . \tag{7.40}$$

In this model P is the *probability mass function* (not a density since we have discrete data) of the Poisson distribution:

$$\mathrm{P}(x; \lambda) \;=\; \frac{1}{x!} \, e^{-\lambda} \, \lambda^x \; . \tag{7.41}$$

In a Bayes framework for model selection, $\boldsymbol{w} = (w_1, \ldots, w_l)$ and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_l)$ are considered as random variables, thus, $p(x)$ in Eq. (7.40) becomes a conditional probability $p(x \mid \boldsymbol{w}, \boldsymbol{\lambda})$. The EM algorithm minimizes an upper bound on the negative log-posterior of the parameters. The parameter posterior of $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ is given by:

$$p(\boldsymbol{w}, \boldsymbol{\lambda} \mid x) \;=\; \frac{p(x \mid \boldsymbol{w}, \boldsymbol{\lambda}) \; p(\boldsymbol{w}) \; p(\boldsymbol{\lambda})}{\int p(x \mid \boldsymbol{w}, \boldsymbol{\lambda}) \; p(\boldsymbol{w}) \; p(\boldsymbol{\lambda}) \; d\boldsymbol{w} \; d\boldsymbol{\lambda}} \; , \tag{7.42}$$

where we assumed that the priors on $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ are independent of each other.

We use a Dirichlet prior with parameters $\boldsymbol{\gamma}$:

$$p(\boldsymbol{w}) \;=\; \mathrm{D}(\boldsymbol{w}^l; \boldsymbol{\gamma}) \;=\; b(\boldsymbol{\gamma}) \; \prod_{j=1}^{l} w_j^{\gamma_j - 1} \; , \tag{7.43}$$

where $\boldsymbol{w}^l$ is the n-dimensional vector $(w_2, \ldots, w_l)$ while w_1 is obtained via $w_1 = 1 - \sum_{j=2}^{l} w_j$. Each component w_j is distributed according to a beta distribution with mean

$$\mathrm{mean}(w_j) \;=\; \frac{\gamma_j}{\gamma_s} \; , \tag{7.44}$$

mode

$$\mathrm{mode}(w_j) \;=\; \frac{\gamma_j - 1}{\gamma_s - l} \; , \tag{7.45}$$

and variance

$$\mathrm{var}(w_j) \;=\; \frac{\gamma_j \; (\gamma_s - \gamma_j)}{\gamma_s^2 \; (\gamma_s + 1)} \; , \tag{7.46}$$

where we set

$$\gamma_s \;=\; \sum_{j=1}^{l} \gamma_i \;. \tag{7.47}$$

For the prior on $\boldsymbol{\lambda}$ we assume that it factorizes into priors for components λ_j. For each component we use an uniform distribution on a sufficiently large interval $(0, 1/t]$ with left endpoint 0 and right endpoint $1/t$. Thus, the density in $(0, 1/t]$ is

$$p(\lambda_j) \;=\; t \;. \tag{7.48}$$

According to Eq. (7.42), the posterior of the model parameters is

$$\begin{aligned} p(\boldsymbol{w}, \boldsymbol{\lambda} \mid x) \;&=\; \frac{p(x \mid \boldsymbol{w}, \boldsymbol{\lambda})\; p(\boldsymbol{w})\; p(\boldsymbol{\lambda})}{\int p(x \mid \boldsymbol{w}, \boldsymbol{\lambda})\; p(\boldsymbol{w})\; p(\boldsymbol{\lambda})\; d\boldsymbol{w}\; d\boldsymbol{\lambda}} \\ &=\; \frac{p(x \mid \boldsymbol{w}, \boldsymbol{\lambda})\; p(\boldsymbol{w})}{\int p(x \mid \boldsymbol{w}, \lambda)\; p(\boldsymbol{w})\; d\boldsymbol{w}\; d\boldsymbol{\lambda}} \\ &=\; \frac{1}{c(x)}\; p(x \mid \boldsymbol{w}, \boldsymbol{\lambda})\; p(\boldsymbol{w}) \;, \end{aligned} \tag{7.49}$$

where $c(x)$ is independent of the parameters $\boldsymbol{w}$ and $\boldsymbol{\lambda}$.

For deriving an upper bound on the log posterior needed by the EM algorithm, we deduce the following inequality for **one** sample x by introducing variables $\hat{w}_j$ with $\sum_{j=1}^{l} \hat{w}_j = 1$:

$$\begin{aligned} -\;\log p(\boldsymbol{w}, \boldsymbol{\lambda} \mid x) \;&=\; -\;\log\left(p(x \mid \boldsymbol{w}, \boldsymbol{\lambda})\; p(\boldsymbol{w}) \;/\; c(x)\right) \\ &=\; -\;\log \sum_{j=1}^{l} w_j\; \mathrm{P}(x; \lambda_j) \;-\; \log p(\boldsymbol{w}) \;+\; \log(c(x)) \\ &=\; -\;\log \sum_{j=1}^{l} \frac{\hat{w}_j}{\hat{w}_j}\; w_j\; \mathrm{P}(x; \lambda_j) \;-\; \log p(\boldsymbol{w}) \;+\; \log(c(x)) \\ &\leq\; -\sum_{j=1}^{l} \hat{w}_j\; \log \frac{w_j\; \mathrm{P}(x; \lambda_j)}{\hat{w}_j} \;-\; \log p(\boldsymbol{w}) \;+\; \log(c(x)) \\ &=\; -\sum_{j=1}^{l} \hat{w}_j\; \log\left(w_j\; \mathrm{P}(x; \lambda_j)\right) \;-\; \log p(\boldsymbol{w}) \\ &\quad\; +\sum_{j=1}^{l} \hat{w}_j\; \log \hat{w}_j \;+\; \log(c(x)) \;, \end{aligned} \tag{7.50}$$

where we applied Jensen's inequality. Note that $c(x)$ is independent of $\boldsymbol{w}$ and that for

$$\hat{w}_j \;=\; p(j \mid x, \boldsymbol{w}, \boldsymbol{\lambda}) \;=\; \frac{w_j\; \mathrm{P}(x; \lambda_j)}{p(x \mid \boldsymbol{w}, \boldsymbol{\lambda})} \tag{7.51}$$

we have in the fifth line of Eq. (7.50)

$$\log \frac{w_j\; \mathrm{P}(x; \lambda_j)}{\hat{w}_j} \;=\; \log p(x \mid \boldsymbol{w}, \boldsymbol{\lambda}) \;, \tag{7.52}$$

thus the inequality Eq. (7.50) becomes an equality.

We assume that the data set $\{x_1, \ldots, x_n\}$ of the counts is given, where the i-th count is denoted by x_i. The posterior that x_i is drawn from the j-th mixture component is

$$w_{ji} \;=\; p(j \mid x_i, \boldsymbol{w}, \boldsymbol{\lambda}) \;=\; \frac{p(j)\; p(x_i \mid j, \boldsymbol{w}, \boldsymbol{\lambda})}{p(x_i \mid \boldsymbol{w}, \boldsymbol{\lambda})} \;=\; \frac{w_j\; \mathrm{P}(x_i; \lambda_j)}{p(x_i \mid \boldsymbol{w}, \boldsymbol{\lambda})}\;, \tag{7.53}$$

where w_j is the prior of being drawn from the j-th mixture component.

We introduce for each x_i variables $\hat{w}_{ji}$ with $\sum_{j=1}^{l} \hat{w}_{ji} = 1$ which estimate $p(j \mid \boldsymbol{w}, x_i, \boldsymbol{\lambda})$ (see Eq. (7.51)). $\hat{w}_{ji}$ are is formally independent of the parameters $\boldsymbol{w}$ and $\boldsymbol{\lambda}$. For the E-step of the EM algorithm, we estimate the posterior w_{ji} by

$$\hat{w}_{ji} \;=\; \frac{w_j^{\mathrm{old}}\; \mathrm{P}(x_i; \lambda_j^{\mathrm{old}})}{p(x_i; \boldsymbol{w}^{\mathrm{old}}, \boldsymbol{\lambda}^{\mathrm{old}})}\;, \tag{7.54}$$

where for the estimation the actual parameters $\boldsymbol{w}^{\mathrm{old}}$ and $\boldsymbol{\lambda}^{\mathrm{old}}$ are used instead of the optimal parameters $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ in the expression for the posterior in Eq. (7.53).

Based on inequality Eq. (7.50) but with $\hat{w}_{ji}$ instead of $\hat{w}_j$, we define an upper bound B on the $\frac{1}{n}$ scaled negative log-posterior for **all** samples as

$$\begin{aligned} B \;=\; & -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{l} \hat{w}_{ji}\; \log\left(w_j\; \mathrm{P}(x; \lambda_j)\right) \;-\; \frac{1}{n}\; \log p(\boldsymbol{w}) \\ & +\; \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{l} \hat{w}_{ji}\; \log \hat{w}_{ji} \;+\; \frac{1}{n} \sum_{i=1}^{n} \log c(x_i)\;, \end{aligned} \tag{7.55}$$

where we summed over all terms depending on x_i. Note, that according to Eq. (7.51) and Eq. (7.52) an exact estimate in the E-step Eq. (7.54) (using the optimal parameters $\boldsymbol{w}$ and $\boldsymbol{\lambda}$) make inequality Eq. (7.50) to an equality, thus the upper bound B would be equal to the negative log posterior.

In the M-step, we minimize the upper bound B on the negative log posterior with respect to w under the constraint that the w_j sum to 1. Only terms depending on $\boldsymbol{w}$ are considered:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{l} \hat{w}_{ji} \log w_j \;-\; \frac{1}{n}\; \log p(\boldsymbol{w}) \\ \text{s.t.} \quad & \sum_{j=1}^{l} w_j \;=\; 1\;. \end{aligned} \tag{7.56}$$

The Lagrangian with Lagrange parameter ρ is

$$\begin{aligned} L \;=\; & -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{l}\hat{w}_{ji}\log w_j \;-\; \frac{1}{n}\log p(\boldsymbol{w}) \\ & +\;\rho\left(\sum_{j=1}^{l} w_j \;-\; 1\right) \\ =\; & -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{l}\hat{w}_{ji}\;\log w_j \;-\; \frac{1}{n}\sum_{j=1}^{l}(\gamma_j \;-\; 1)\;\log w_j \\ & +\;\rho\left(\sum_{j=1}^{l} w_j \;-\; 1\right)\;. \end{aligned} \tag{7.57}$$

The solution requires that, the derivative of L with respect to w_j is zero:

$$\frac{\partial L}{\partial w_j} \;=\; -\frac{1}{n}\sum_{i=1}^{n}\hat{w}_{ji}\frac{1}{w_j} \;-\; \frac{1}{n}\frac{1}{w_j}\,(\gamma_j \;-\; 1) \;+\; \rho \;=\; 0\,. \tag{7.58}$$

Multiplying this equation by w_j gives

$$-\frac{1}{n}\sum_{i=1}^{n}\hat{w}_{ji} \;-\; \frac{1}{n}\,(\gamma_j \;-\; 1) \;+\; \rho\, w_j \;=\; 0\,. \tag{7.59}$$

Summation over j leads to

$$1 \;+\; \frac{1}{n}\,(\gamma_s \;-\; l) \;=\; \rho\,. \tag{7.60}$$

Inserting this expression for ρ in Eq. (7.59) results in

$$-\frac{1}{n}\sum_{i=1}^{n}\hat{w}_{ji} \;-\; \frac{1}{n}\,(\gamma_j \;-\; 1) \;+\; \left(1 \;+\; \frac{1}{n}\,(\gamma_s \;-\; l)\right)\, w_j \;=\; 0\,. \tag{7.61}$$

Solving Eq. (7.61) for w_j gives the update rule for w_j:

$$w_j^{\text{new}} \;=\; \frac{\hat{w}_j \;+\; \frac{1}{n}\,(\gamma_j \;-\; 1)}{1 \;+\; \frac{1}{n}\,(\gamma_s \;-\; l)}\,, \tag{7.62}$$

where we used

$$\hat{w}_j \;=\; \frac{1}{n}\sum_{i=1}^{n}\hat{w}_{ji}\,. \tag{7.63}$$

We introduced $\hat{w}_j$ which sums up the $\hat{w}_{ji}$ and thereby approximates w_j. This approximation is justified because w_j can be decomposed into w_{ji}:

$$\begin{aligned} w_j \;&=\; p(j) \;=\; p(j \mid \boldsymbol{w}, \boldsymbol{\lambda}) \;=\; \int p(j, x \mid \boldsymbol{w}, \boldsymbol{\lambda})\, dx \\ &=\; \int p(j \mid x, \boldsymbol{w}, \boldsymbol{\lambda})\, p(x \mid \boldsymbol{w}, \boldsymbol{\lambda})\, dx \;=\; \mathrm{E}_{p(x|\boldsymbol{w},\boldsymbol{\lambda})}(p(j \mid x, \boldsymbol{w}, \boldsymbol{\lambda})) \\ &\approx\; \frac{1}{n} \sum_{i=1}^{n} p(j \mid x_i, \boldsymbol{w}, \boldsymbol{\lambda}) \;=\; \frac{1}{n} \sum_{i=1}^{n} w_{ji} \;=\; \hat{w}_j \;. \end{aligned} \tag{7.64}$$

In the M-step, B need not only be minimized with respect to $\boldsymbol{w}$ but also with respect to λ_j (only terms depending on λ_j are considered):

$$\min_{\lambda_j} \left(-\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{l} \hat{w}_{ji} \log \mathrm{P}(x; \lambda_j) \right) . \tag{7.65}$$

For the minimum, the derivative of the above objective with respect to λ_j must be zero. Using

$$\log \mathrm{P}(x_i; \lambda_j) \;=\; -\log(x_i!) \;-\; \lambda_j \;+\; x_i \log(\lambda_j) \;, \tag{7.66}$$

this derivative is

$$-\frac{1}{n} \sum_{i=1}^{n} \left(-1 \;+\; \frac{x_i}{\lambda_j} \right) \hat{w}_{ji} \;. \tag{7.67}$$

Multiplying Eq. (7.67) by λ_j and solving it for λ_j gives the update rule:

$$\lambda_j^{\mathrm{new}} \;=\; \frac{\sum_{i=1}^{n} x_i\, \hat{w}_{ji}}{\sum_{i=1}^{n} \hat{w}_{ji}} \;. \tag{7.68}$$

The update rules can be summarized as follows:

$$\hat{w}_{ji} \;=\; \frac{w_j^{\mathrm{old}}\, \mathrm{P}(x_i; \lambda_j^{\mathrm{old}})}{p(x_i \mid \boldsymbol{w}^{\mathrm{old}}, \boldsymbol{\lambda}^{\mathrm{old}})} \;, \tag{7.69}$$

$$w_j^{\mathrm{new}} \;=\; \frac{\frac{1}{n} \sum_{i=1}^{n} \hat{w}_{ji} \;+\; \frac{1}{n}(\gamma_j \;-\; 1)}{1 \;+\; \frac{1}{n}(\gamma_s \;-\; l)} \;, \tag{7.70}$$

$$\lambda_j^{\mathrm{new}} \;=\; \frac{\frac{1}{n} \sum_{i=1}^{n} \hat{w}_{ji}\, x_i}{\frac{1}{n} \sum_{i=1}^{n} \hat{w}_{ji}} \;. \tag{7.71}$$

Concerning the EM algorithm the update rule Eq. (7.69) is the E-step, the update rule Eq. (7.70) is the M-step for $\boldsymbol{w}$, and the update rule Eq. (7.70) is the M-step for $\boldsymbol{\lambda}$.

The update rule Eq. (7.70) can be obtained in an alternative way. The Dirichlet distribution is conjugate to the multinomial distribution, that is the posterior $p(\boldsymbol{w} \mid \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_i, \ldots, \boldsymbol{w}_n\})$ is a Dirichlet distribution as is the prior $p(\boldsymbol{w})$ with $\boldsymbol{w}_i = p(\boldsymbol{w} \mid x_i)$. The Dirichlet prior $p(\boldsymbol{w}) =$

$\mathrm{D}(\boldsymbol{w}^1; \boldsymbol{\gamma})$ with parameters $\boldsymbol{\gamma}$ leads to the conjugate posterior $p(\boldsymbol{w} \mid \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_i, \ldots, \boldsymbol{w}_n\})$ with parameters

$$\hat{\boldsymbol{\gamma}} \;=\; \boldsymbol{\gamma} \;+\; \sum_{i=1}^{n} \boldsymbol{w}_i \;=\; \boldsymbol{\gamma} \;+\; N\,\boldsymbol{w}\,, \tag{7.72}$$

where we used Eq. (7.64). We obtain update rule Eq. (7.70) from Eq. (7.72) component-wise by first replacing the unknown values w_{ji} by their estimates $\hat{w}_{ji}$ and then computing the posterior's mode because we search for the maximum posterior.

7.1.4 Examples

In the following we will show examples for the mixtures of Gaussians. Fig. 7.1 shows a mixture of Gaussians example on toy data. The results presented in the panels in Fig. 7.1 differ in the way the initialization of the MoG model was done. "em" is a two-step procedure: first several EM runs are performed with high tolerance, which leads to fast but not very precise solutions. Subsequently, a run is started with low tolerance, that is the run which will give a precise solution. "rnd" samples some random initializations and picks the best solution for a long EM run. "svd" utilizes singular value decomposition to find a good initialization for the EM algorithm.

For the clustering we can impose different constraints on the parameters, especially the covariance matrix. "Spherical" means that the covariance matrix is a multiple of the identity. Diagonal means that the clusters are elongated along the axis but may have different variance. Volume means the weighting factor w_j or, equivalently, the priors $p(j)$, which can be all equal or have different values. Some constraints are:

```
univariate mixture:
"E"  =  equal variance (one-dimensional)
"V"  =  variable variance (one-dimensional)
multivariate mixture:
"EII"  =  spherical, equal volume
"VII"  =  spherical, unequal volume
"EEI"  =  diagonal, equal volume and shape
"VEI"  =  diagonal, varying volume, equal shape
"EVI"  =  diagonal, equal volume, varying shape
"VVI"  =  diagonal, varying volume and shape
"EEE"  =  ellipsoidal, equal volume, shape, and orientation
"EEV"  =  ellipsoidal, equal volume and equal shape
"VEV"  =  ellipsoidal, equal shape
"VVV"  =  ellipsoidal, varying volume, shape, and orientation
single component:
"X"  =  univariate normal
"XII"  =  spherical multivariate normal
"XXI"  =  diagonal multivariate normal
"XXX"  =  elliposidal multivariate normal
```

Fig. 7.2 shows the results of mixture of Gaussians with 3 components applied to the iris data set. The upper left panel shows the true clusters (the species). The other panels show the result

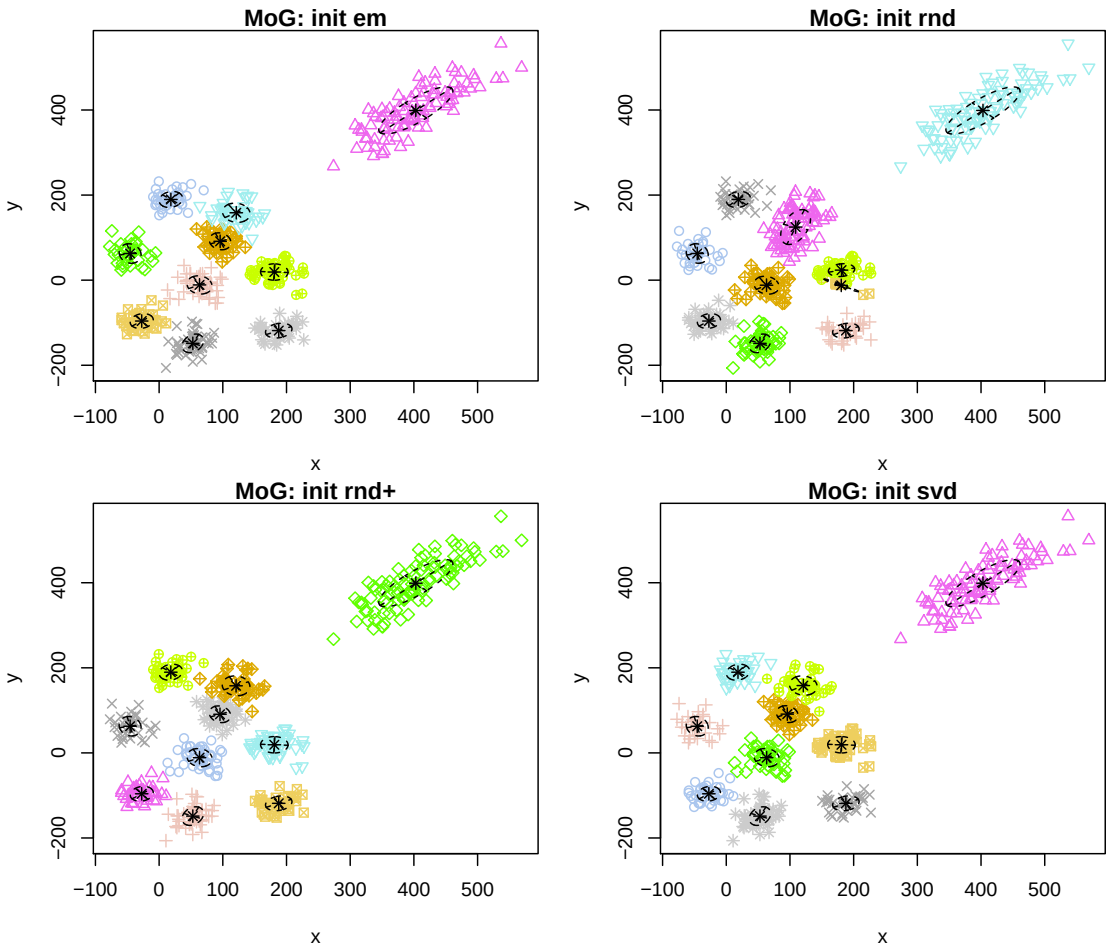


Figure 7.1: Mixture of Gaussians example on toy data from the R package `EMCluster`. The panels differ in the way the initialization of the MoG model was done. The ellipsoid Gaussians are depicted in the figures.

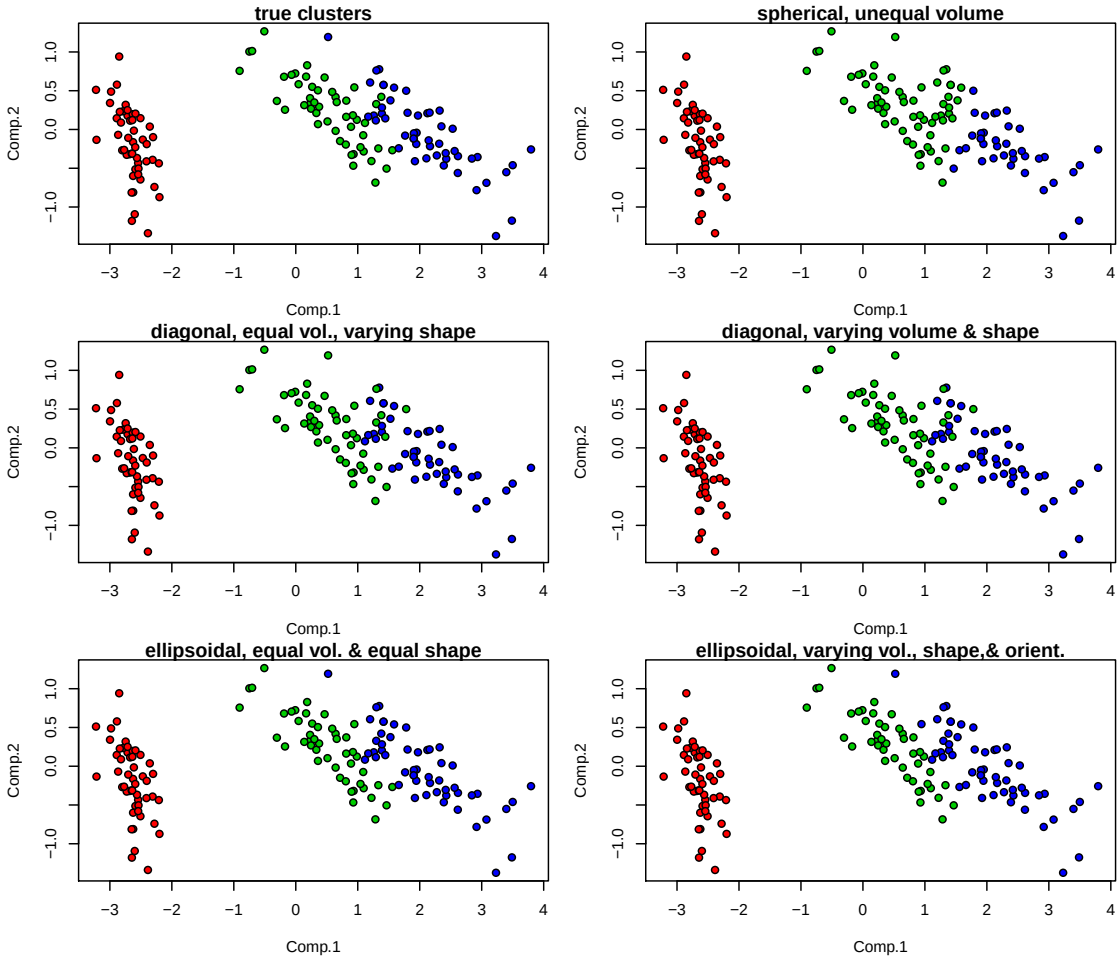


Figure 7.2: Mixture of Gaussians with 3 components applied to Iris data. The upper left panel shows the true clusters (the species). The other panels show the result of mixture of Gaussians with different constraints on the parameters.

of mixture of Gaussians with different constraints on the parameters. The constraints enforce "spherical, unequal volume", "diagonal, equal volume, varying shape", "diagonal, varying volume and shape", "ellipsoidal, equal volume and equal shape", and "ellipsoidal, varying volume, shape, and orientation". The latter does not impose constraints on the mixture of Gaussians model. The results differ only slightly. Fig. 7.3 shows the results of mixture of Gaussians with 6 components applied to the iris data set.

Fig. 7.4 shows the results of mixture of Gaussians with 3 and 4 components applied to the multiple tissues data set. The 76 genes with largest variance are filtered before applying the mixture of Gaussian model. The upper left panel shows the true clusters (the tissues). The other panels show the result of Mixture of Gaussians with two different constraints, "spherical, unequal volume" and "diagonal, equal volume, varying shape", and different number of components. Fig. 7.5 shows the results of mixture of Gaussians with different number of components applied to the multiple tissues data set. The upper left panel shows the true clusters (the tissues). The other panels show

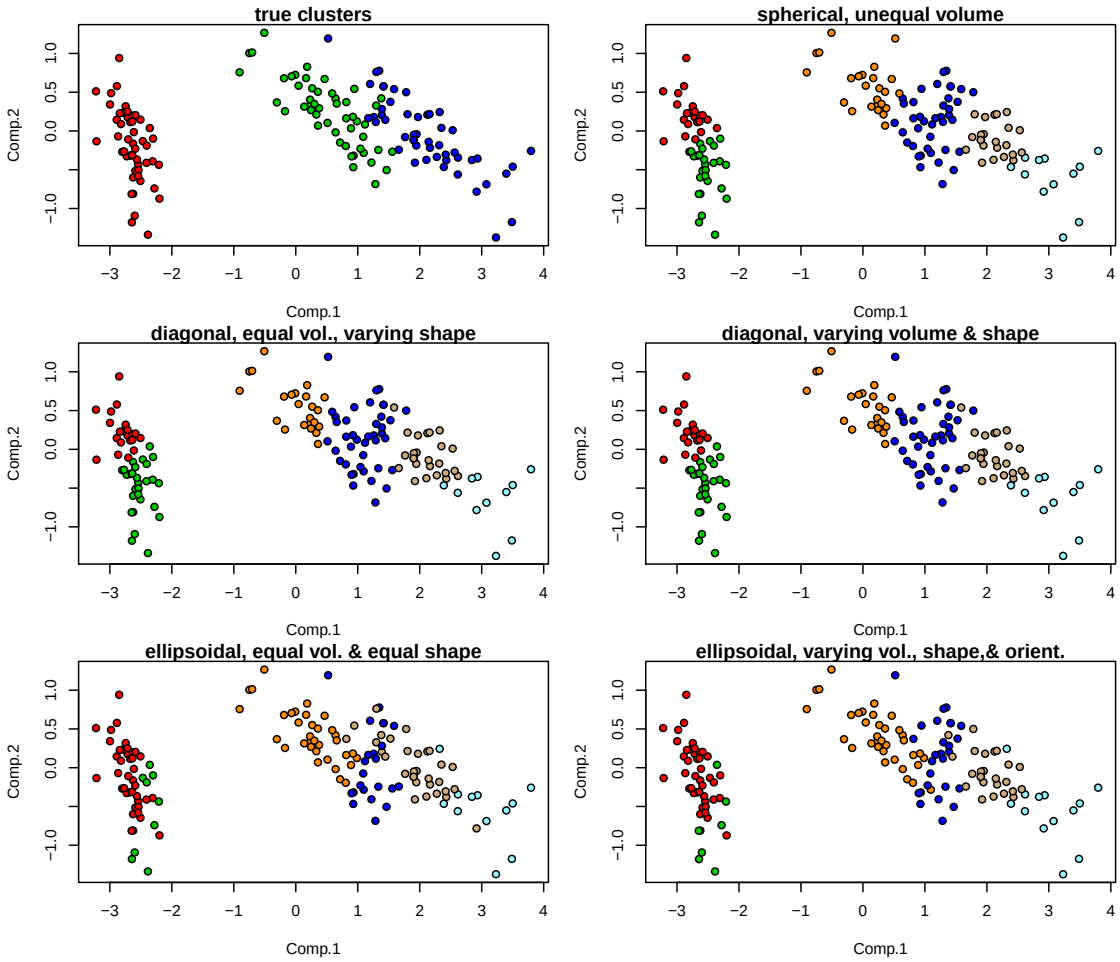


Figure 7.3: Mixture of Gaussians with 6 components applied to Iris data. The upper left panel shows the true clusters (the species). The other panels show the result of mixture of Gaussians with different constraints on the parameters.

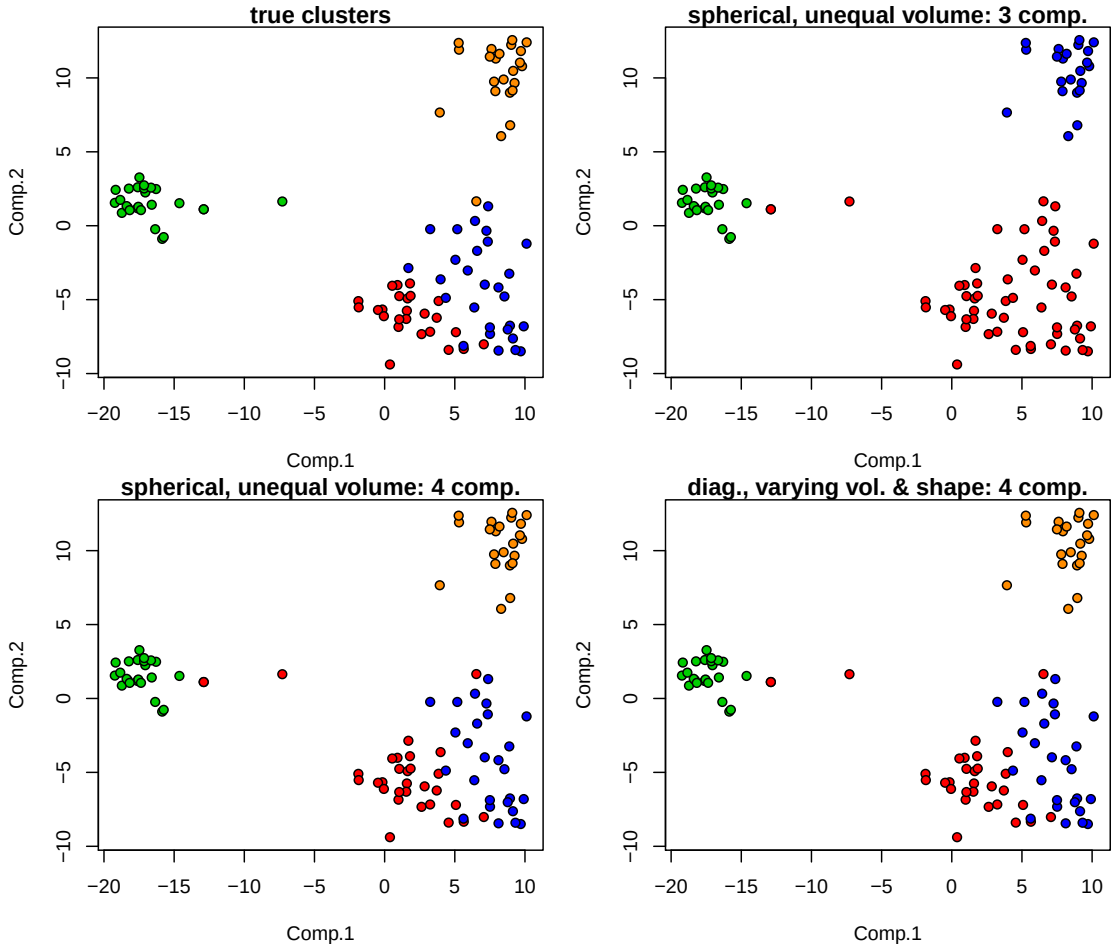


Figure 7.4: Mixture of Gaussians applied to multiple tissues with 3 and 4 components. 76 genes with largest variance are filtered.

the result of mixture of Gaussians with spherical components which may have unequal volume. We tested 3 to 7 components.

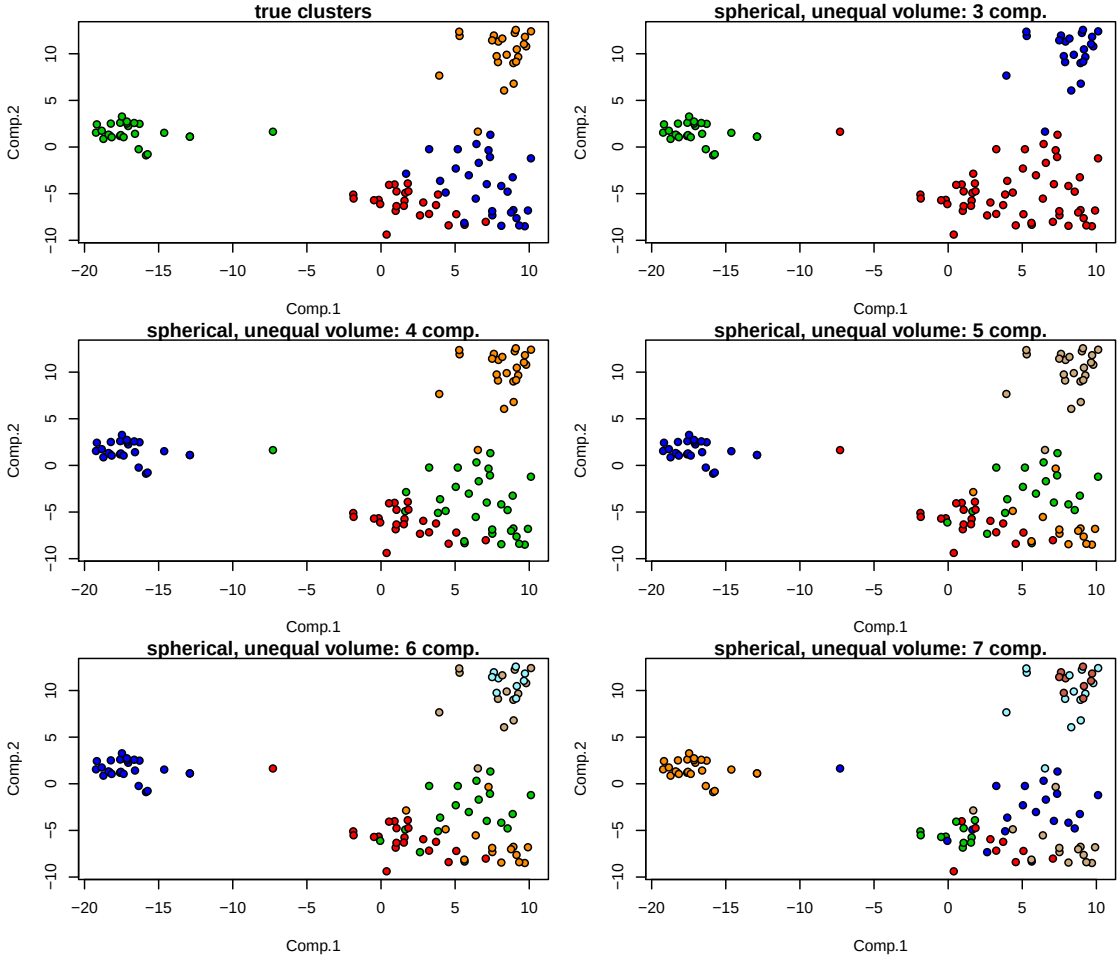


Figure 7.5: Mixture of Gaussians applied to multiple tissues with different number of components. The components are spherical but may have unequal volume.

7.2 k-Means Clustering

7.2.1 The Method

Probably the best known clustering algorithm is *k-means clustering* Forgy [1965], Hartigan [1972, 1975], Hartigan and Wong [1979]. k-means assumes k clusters but we denote the number of clusters by l to keep the notation that we used for other methods.

If we simplify the model of mixture clustering, then we obtain k-means clustering. The simplifications are:

- equal weight (equal volume) for each component: $w_j = \frac{1}{l}$,
- spherical and equal (between components) covariance $\boldsymbol{\Sigma}_j^{-1} = \boldsymbol{I}$,
- hard (discrete) cluster membership (a sample belongs to a cluster or not).

The only remaining parameters are the cluster centers.

For the cluster memberships we used $p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ in Eq. (7.2). This value is determined by the weight w_j of the j-th component and the distance $(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j)$ of $\boldsymbol{x}_i$ to the mean $\boldsymbol{\mu}_j$. The simplification sets $w_j = \frac{1}{l}$ and $\boldsymbol{\Sigma}_j^{-1} = \boldsymbol{I}$. Thus, x_i belongs to the cluster j with the closest center $\boldsymbol{\mu}_j$, i.e. the smallest Euclidean distance $\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|$ to $\boldsymbol{x}_i$. The third simplification was hard cluster membership, therefore, we obtain:

$$p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j) = \begin{cases} 1 & \text{if } j = c_{\boldsymbol{x}_i} = \arg\min_k \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\| \\ 0 & \text{otherwise} \end{cases} . \tag{7.73}$$

The M-step in Appendix 7.1.2 in Eq. (7.15) becomes (only the centers are updated):

$$\boldsymbol{\mu}_j^{\text{new}} = \frac{1}{n_j} \sum_{i=1,\ j=c_{\boldsymbol{x}_i}}^{n} \boldsymbol{x}_i \tag{7.74}$$

$$n_j = \sum_{i=1}^{n} p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \sum_{i=1,\ j=c_{\boldsymbol{x}_i}}^{n} 1 , \tag{7.75}$$

where n_j is the number of data points assigned to cluster j.

Therefore $\boldsymbol{\mu}_j^{\text{new}}$ is the mean of the data points assigned to cluster j. The *k-means clustering* algorithm is given in Alg. 7.1.

The k-means clustering:

- fast,
- robust to outliers (covariance),
- simple (can be an advantage or a disadvantage),
- prone to the initialization.

Algorithm 7.1 k-means

Given: data $\{\boldsymbol{x}\} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, number of clusters l

BEGIN initialization
initialize the cluster centers $\boldsymbol{\mu}_j$, $1 \leq j \leq l$
END initialization

BEGIN Iteration

Stop=false
while Stop=false **do**
for $(i = 1\ ;\ i\ \geq\ n\ ;\ i++)$ **do**
assign $\boldsymbol{x}_i$ to the nearest $\boldsymbol{\mu}_j$
end for
for $(j = 1\ ;\ j\ \geq\ l\ ;\ j++)$ **do**

$$\boldsymbol{\mu}_j^{\text{new}} \ = \ \frac{1}{n_j} \sum_{i=1,\ j=c_{\boldsymbol{x}_i}}^{n} \boldsymbol{x}_i$$

end for
if stop criterion fulfilled **then**
Stop=true
end if
end while
END Iteration

For example, consider an initialization which places one center near several outliers which are separated from the rest of the data points. The other data points have other cluster centers closer to them. Then this outlier cluster will remain in each iteration at the outliers even if other cluster are not modeled. This behavior can be serious in high dimensions.

The membership can be made continuous by using a softmax function. Let us again assume $w_j = \frac{1}{l}$ and $\mathbf{\Sigma}_j^{-1} = \boldsymbol{I}$. But now we use a continuous estimate of $p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j, \mathbf{\Sigma}_j) = p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j)$, where the distances do not have an exponential decay as in Eq. (7.14).

We define the softmax membership with parameter b as

$$p^b(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j) = \frac{\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^{-2/(b-1)}}{\sum_{k=1}^{l} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^{-2/(b-1)}} \,. \tag{7.76}$$

and obtain

$$\boldsymbol{\mu}_j^{\text{new}} = \frac{\sum_{i=1}^{n} p^b(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j)\, \boldsymbol{x}_i}{\sum_{i=1}^{n} p^b(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j)} \,. \tag{7.77}$$

The objective, which is minimized, is

$$\sum_{j=1}^{l} \sum_{i=1}^{n} p^b(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j)\, \boldsymbol{x}_i\, \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2 \,. \tag{7.78}$$

This algorithm is called *fuzzy k-means clustering* and described in Alg. 7.2.

7.2.2 Examples

We demonstrate k-means on an artificial data set in two dimensions with five clusters. Fig. 7.6 shows the result of k-means with $k = 5$ where an optimal solution is found. Filled circles mark the cluster centers. Local minima are shown in Fig. 7.7 and Fig. 7.8. In both cases one cluster explains two true clusters while one true cluster is divided into two model clusters. Fig. 7.9 shows a local minimum, where three model clusters share one true cluster.

We apply k-means with $k = 8$ to the five cluster data set. In this case the number of model clusters does not match the number of true clusters. Therefore the solution will always be worse than the optimal solution with the correct number of clusters. Fig. 7.10 shows a solution where three true clusters are shared by pairs of model clusters. Fig. 7.11 shows a solution where one true cluster is shared by 3 model clusters and another by 2 model clusters. This solution is very typical and another example is presented in Fig. 7.12. Fig. 7.13 shows a solution where a true cluster is shared by four model clusters. The remaining true clusters are correctly explained by one model cluster.

Algorithm 7.2 Fuzzy k-means

Given: data $\{\boldsymbol{x}\} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, number of clusters l, parameter b

BEGIN initialization
initialize the cluster centers $\boldsymbol{\mu}_j$, $1 \leq j \leq l$, and $w_j(\boldsymbol{x}_i) = p(j \mid \boldsymbol{x}_i, \boldsymbol{\mu}_j)$ so that $\sum_{j=1}^{l} w_j(\boldsymbol{x}_i) = 1$, $w_j(\boldsymbol{x}_i) \geq 0$.
END initialization

BEGIN Iteration

Stop=false
while Stop=false **do**

$$\boldsymbol{\mu}_j^{\text{new}} = \frac{\sum_{i=1}^{n} w_j(\boldsymbol{x}_i)\, \boldsymbol{x}_i}{\sum_{i=1}^{n} w_j(\boldsymbol{x}_i)}$$

$$w_j(\boldsymbol{x}_i) = \frac{\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^{-2/(b-1)}}{\sum_{k=1}^{l} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^{-2/(b-1)}}$$

if stop criterion fulfilled **then**
Stop=true
end if
end while
END Iteration

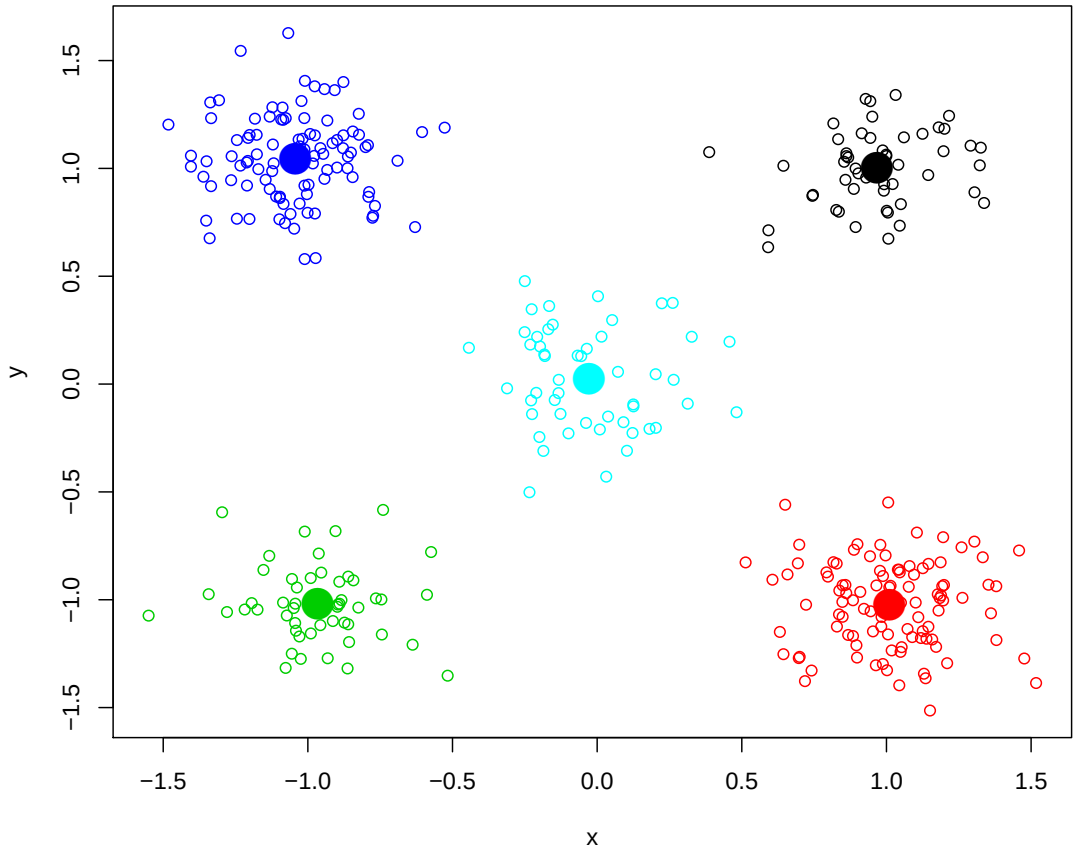


Figure 7.6: k-means clustering of the five cluster data set with $k = 5$ where filled circles mark the cluster centers. An optimal solution is found.

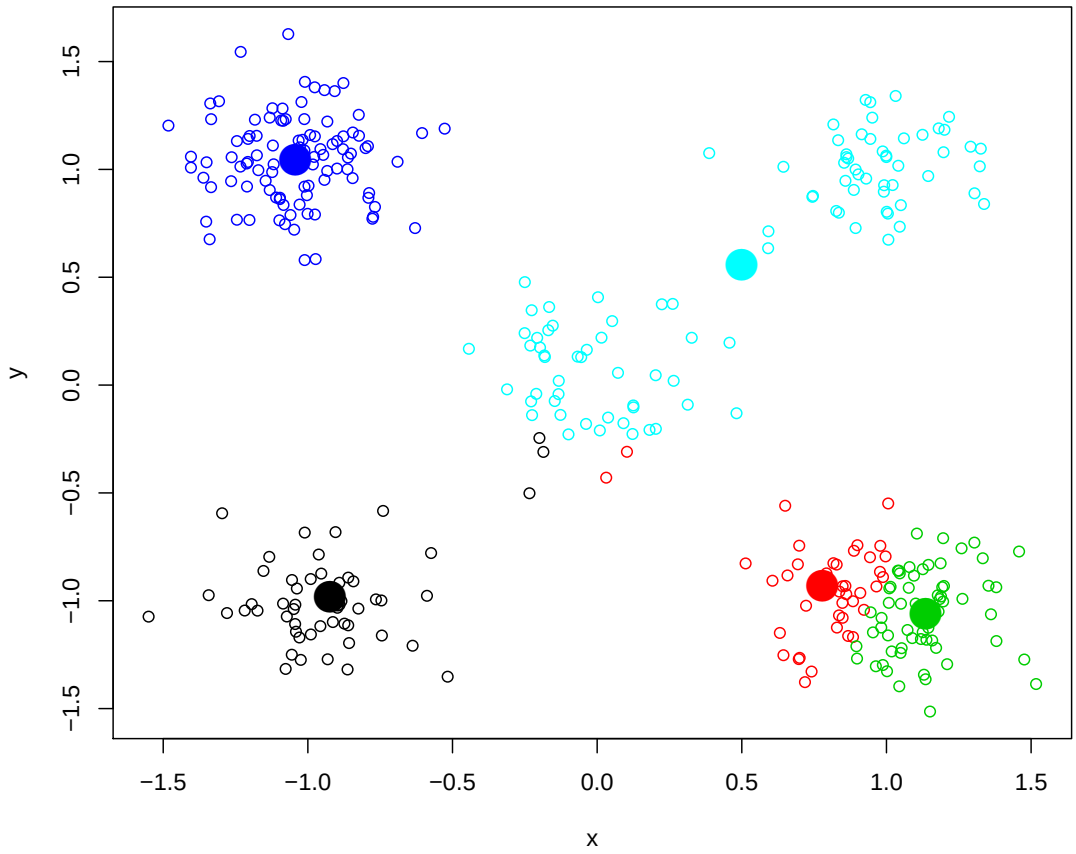


Figure 7.7: k-means clustering of the five cluster data set with $k = 5$ where filled circles mark the cluster centers. A local minimum is found.

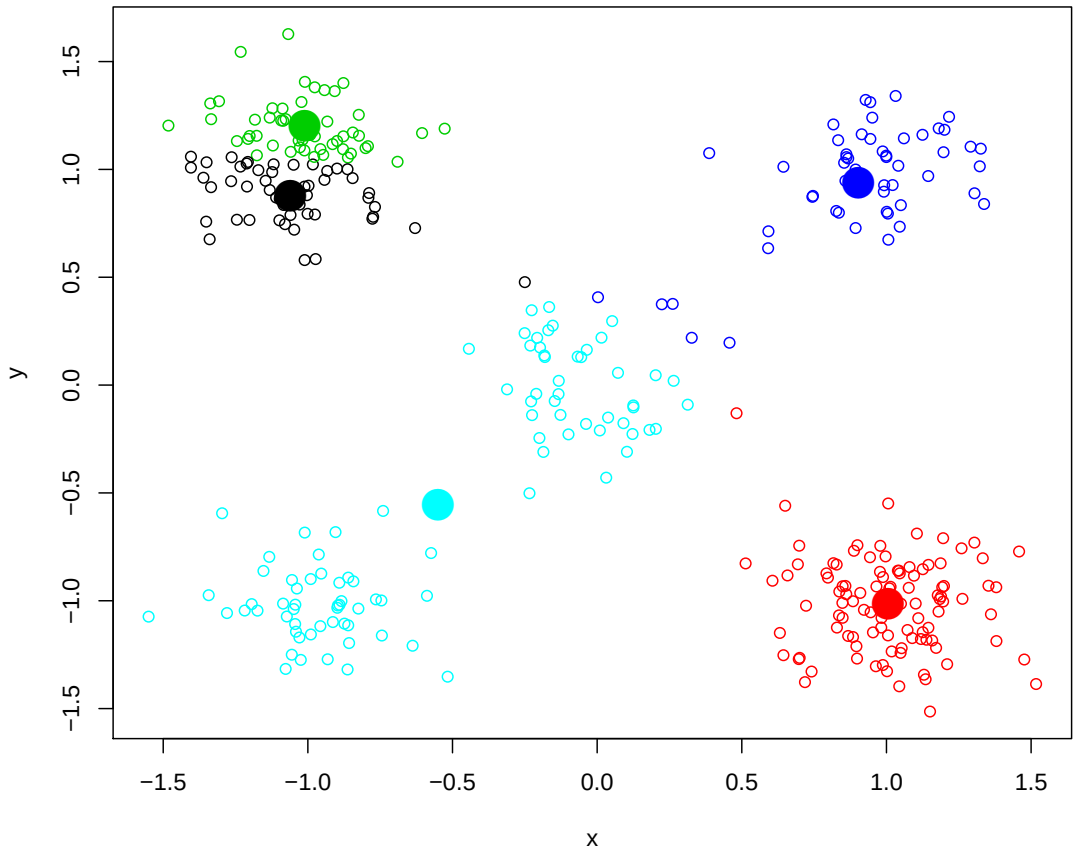


Figure 7.8: k-means clustering of the five cluster data set with $k = 5$ where filled circles mark the cluster centers. A local minimum is found.

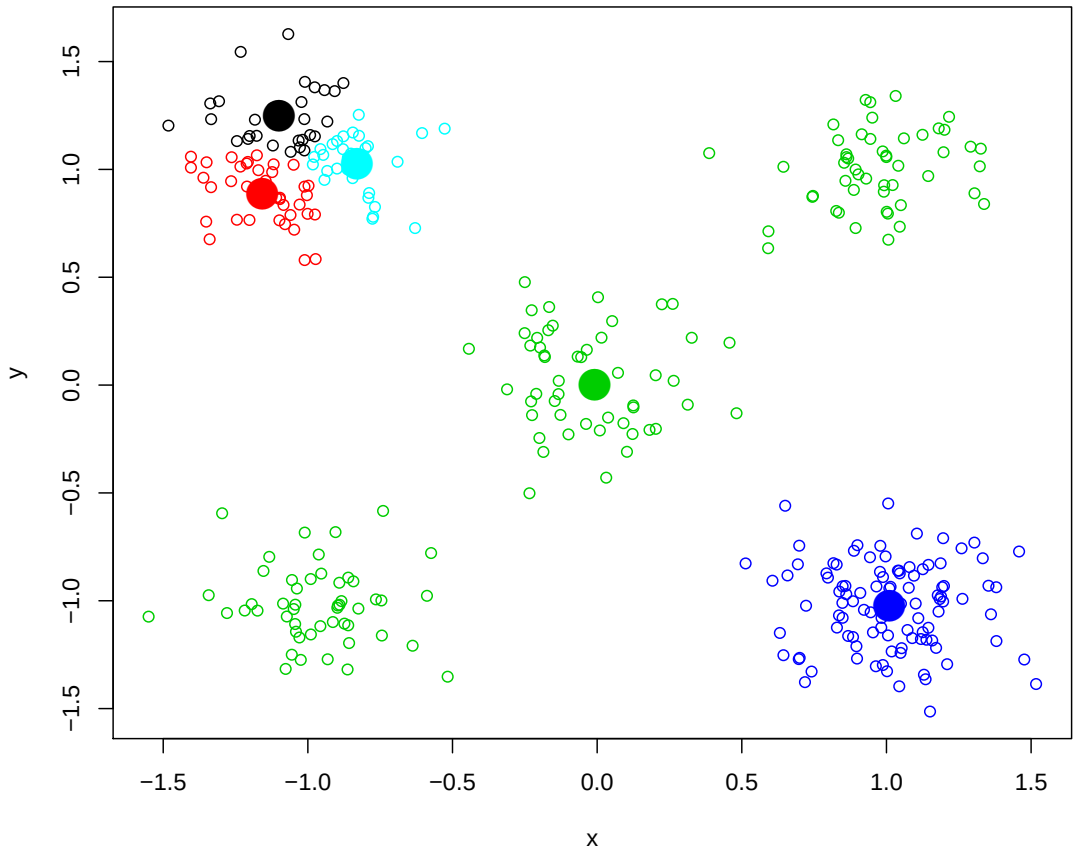


Figure 7.9: k-means clustering of the five cluster data set with $k = 5$ where filled circles mark the cluster centers. A local minimum is found where three model cluster share one true cluster.

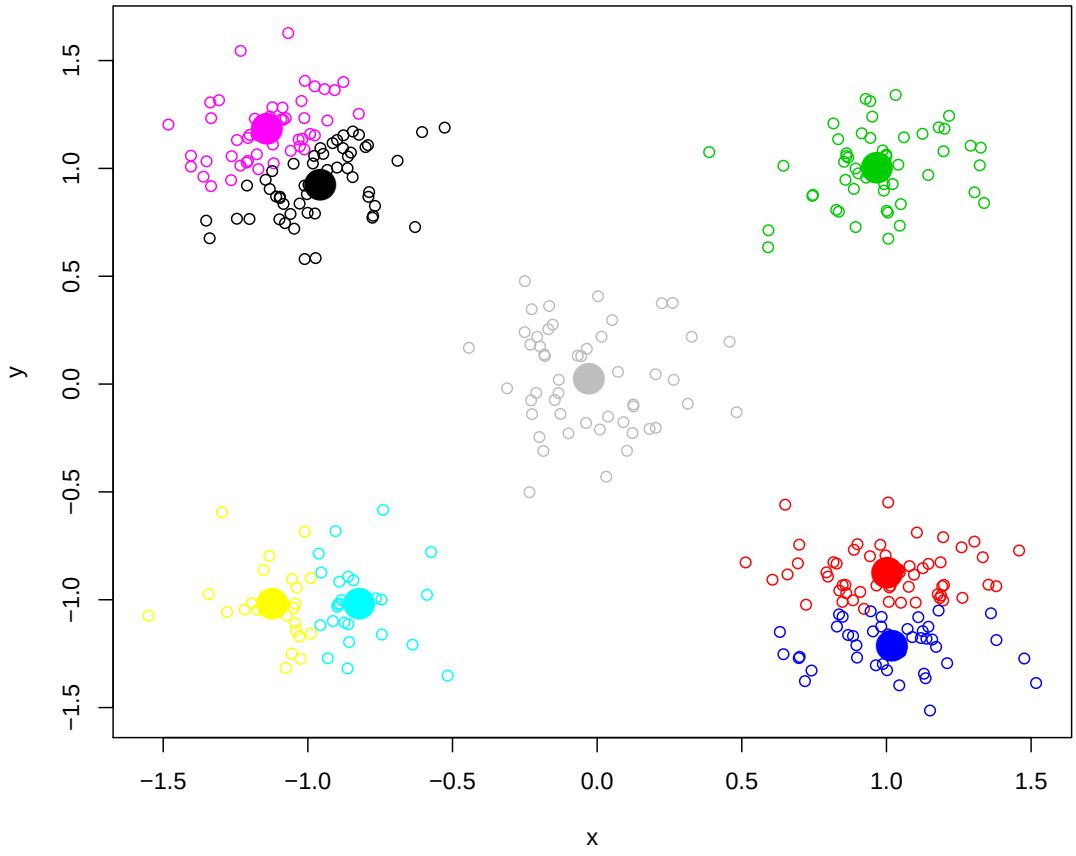


Figure 7.10: k-means clustering of the five cluster data set with $k = 8$ where filled circles mark the cluster centers. True clusters are shared by pairs of model clusters.

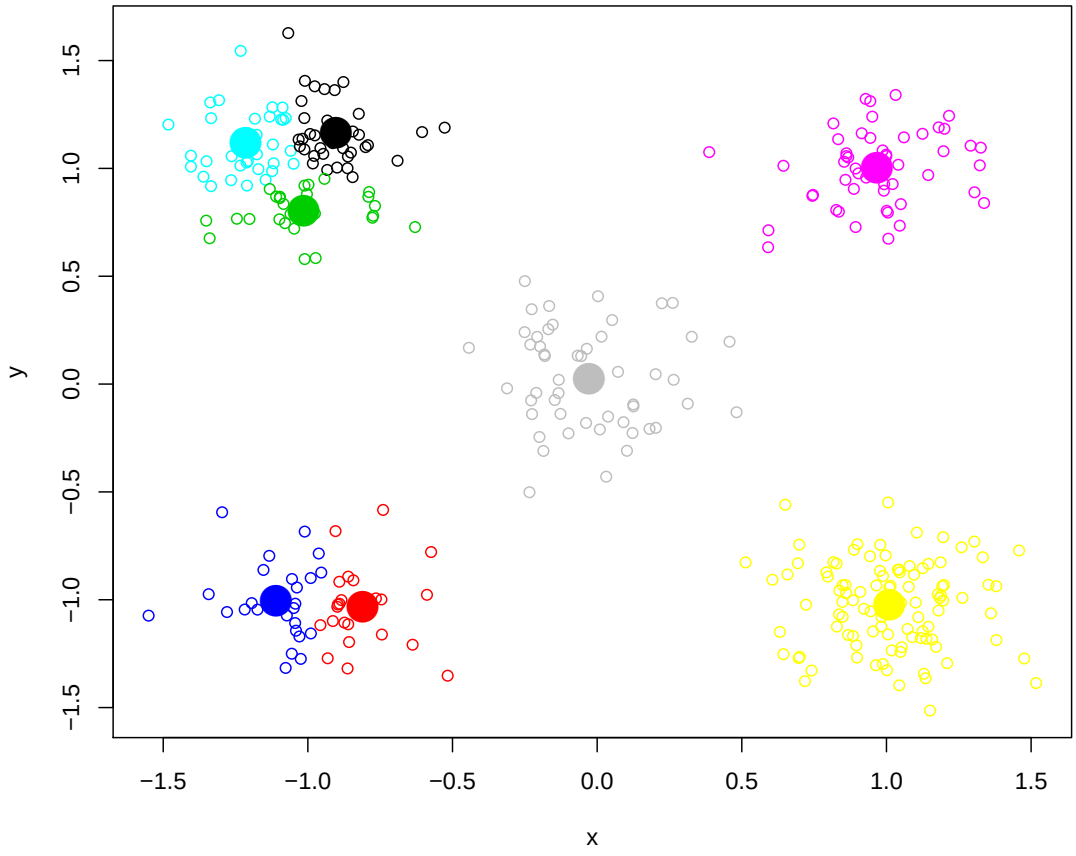


Figure 7.11: k-means clustering of the five cluster data set with $k = 8$. Typical case where one true cluster is shared by 3 model clusters and another by 2 model clusters.

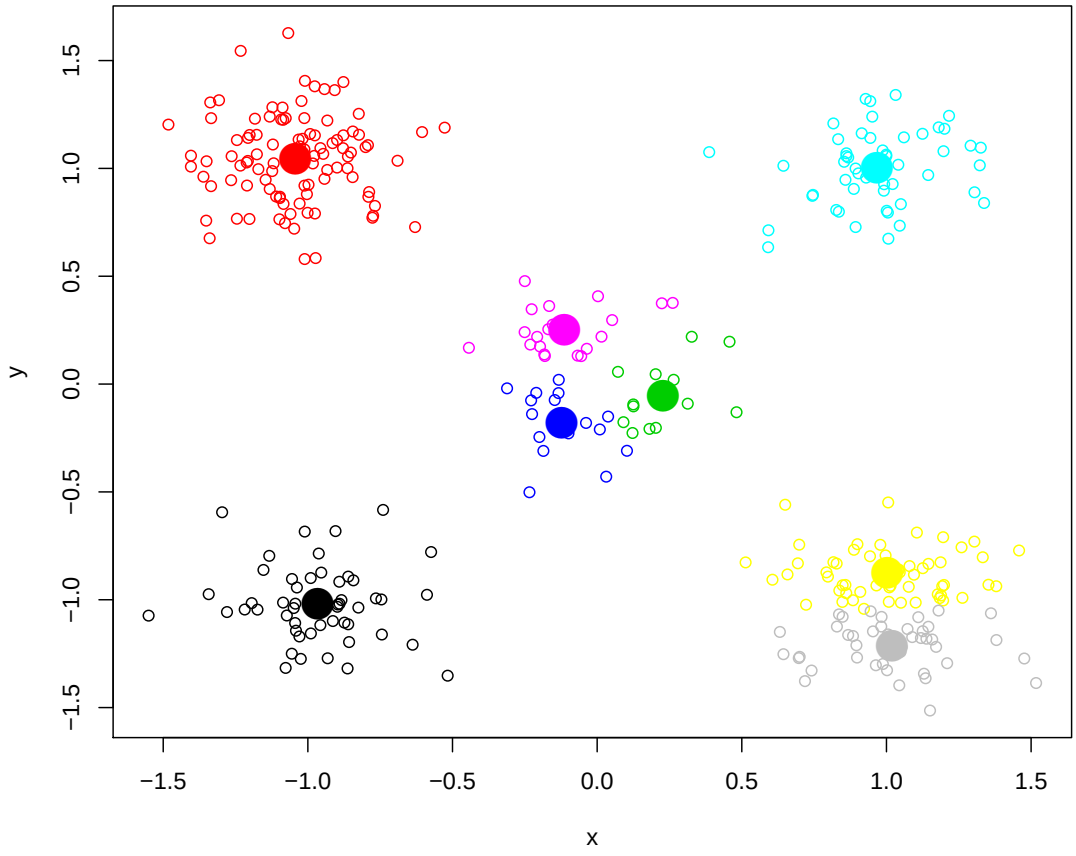


Figure 7.12: k-means clustering of the five cluster data set with $k = 8$. Another example of the situation like Fig. 7.11.

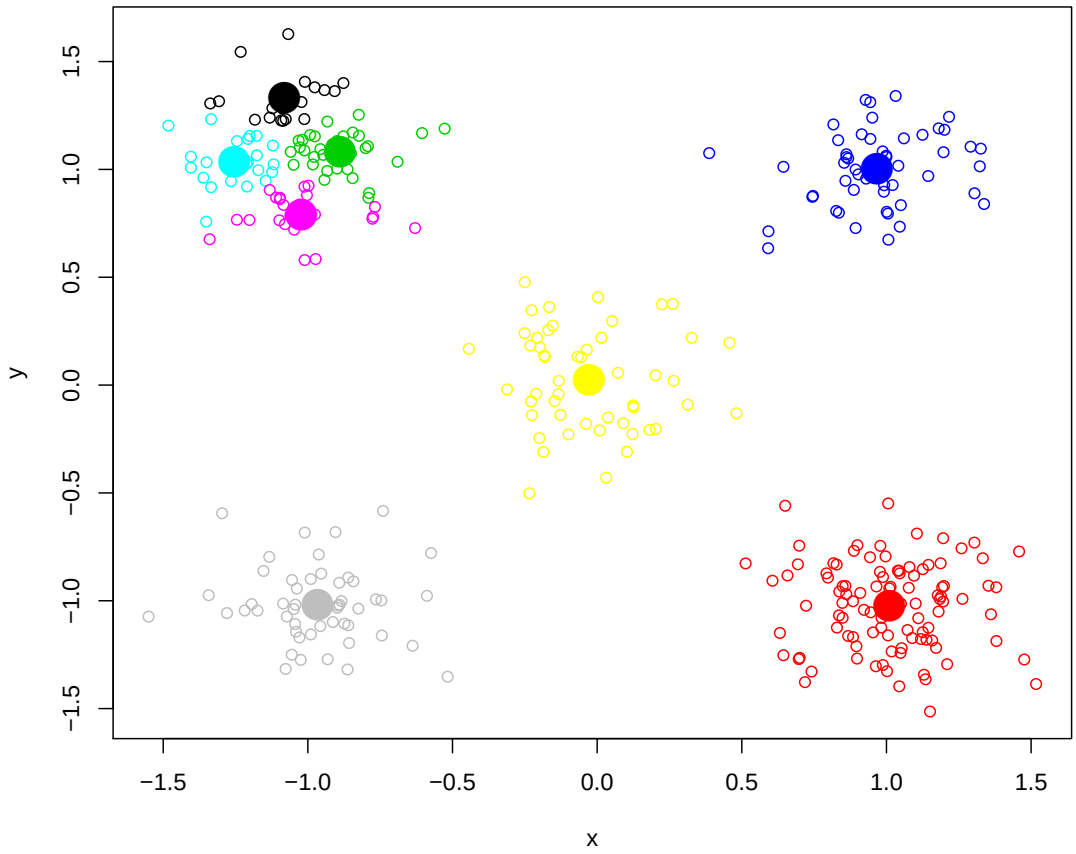


Figure 7.13: k-means clustering of the five cluster data set with $k = 8$. A true cluster is shared by four model clusters.

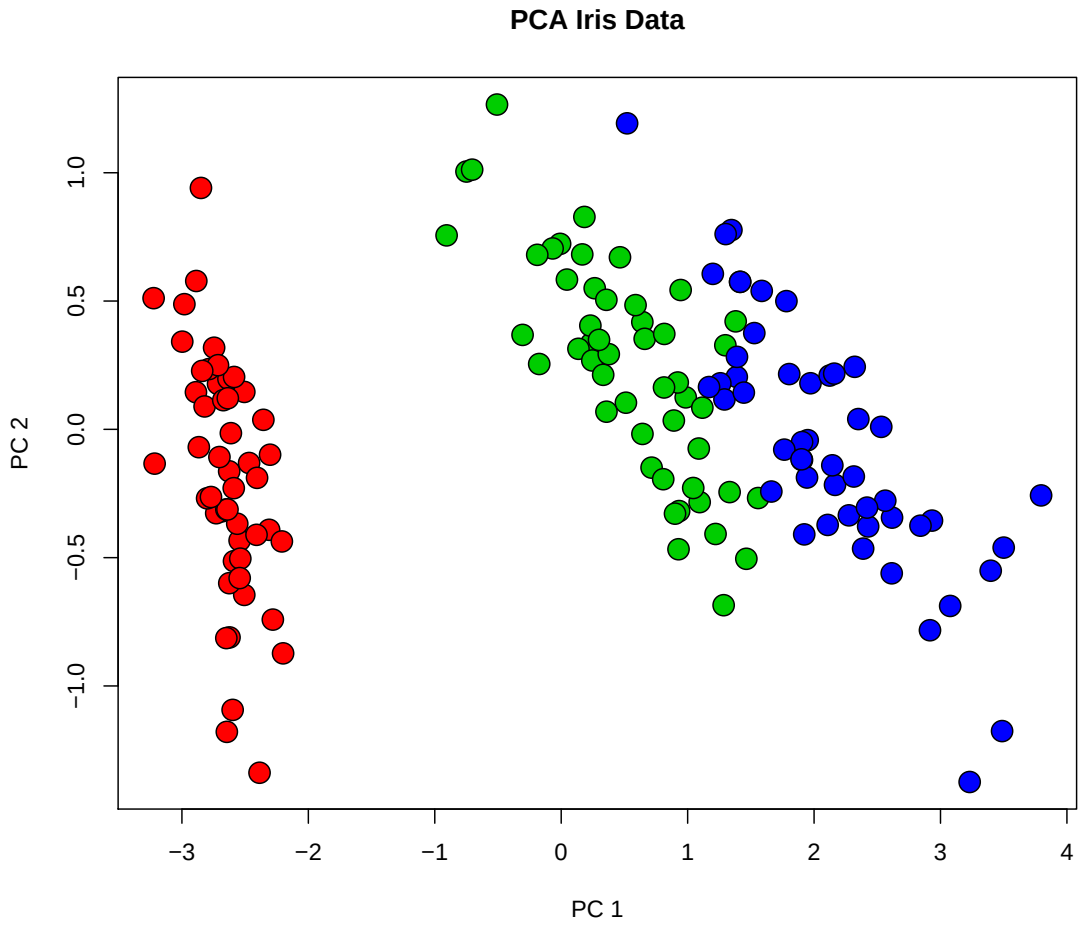


Figure 7.14: Down-projection of the iris data set to two dimensions by PCA. The true classes are marked by colors.

We apply k-means to the Iris data set. To remind the reader, the down-projection onto two dimensions by PCA is again given in Fig. 7.14, where the true classes are marked by colors. Fig. 7.15 shows a typical solution of k-means applied to the Iris data set. The solution is quite good, only at the border assignments are made wrong. Fig. 7.16 gives another typical solution of k-means for the Iris data set. This solution is not good as two components share a cluster. Important question is whether the quality of these solutions can be distinguished if the true classes are not known.

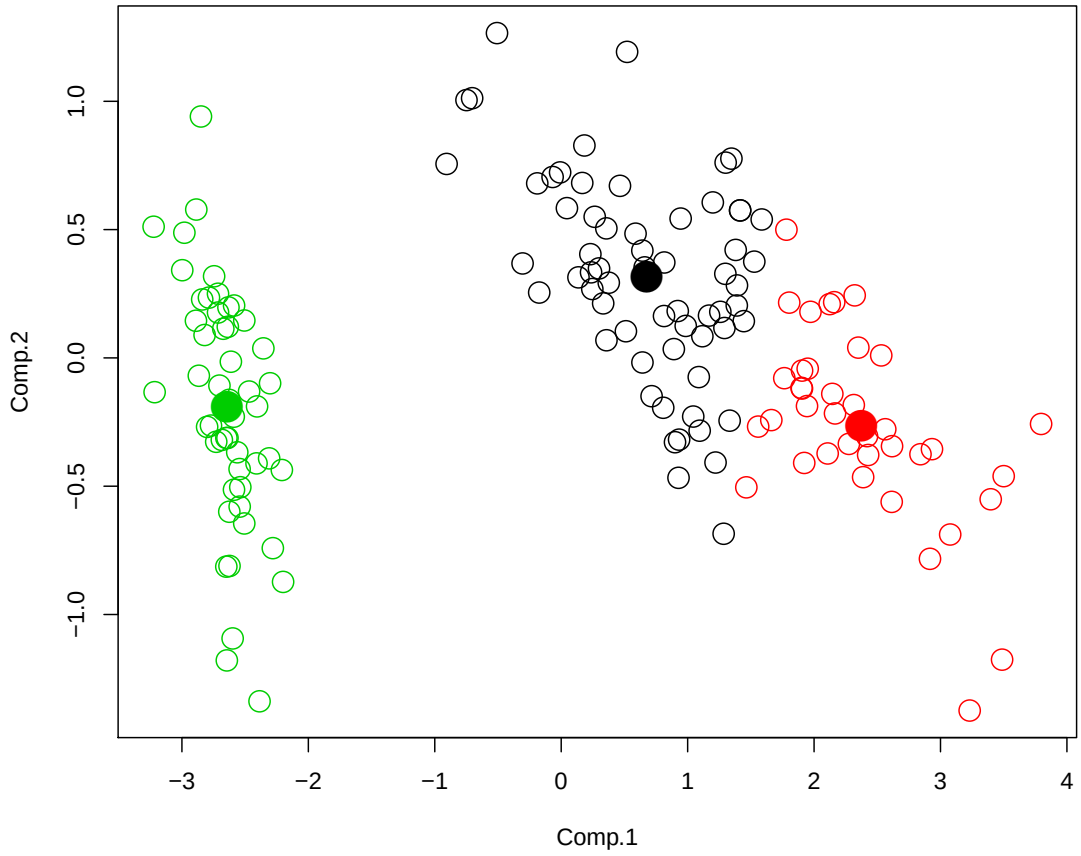


Figure 7.15: k-means clustering of the Iris data set. The **first** typical solution where filled circles are cluster centers. The solution is quite good, only at the border assignments are made wrong.

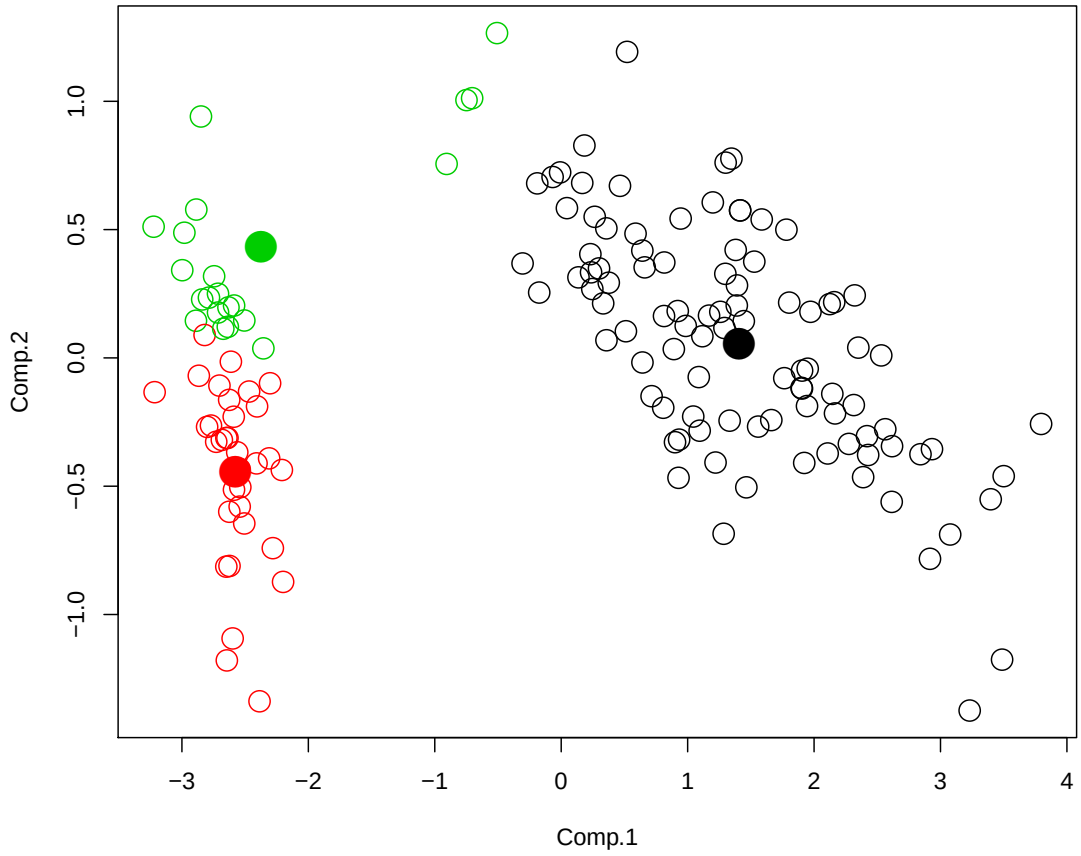


Figure 7.16: k-means clustering of the Iris data set. The **second** typical solution where filled circles are cluster centers. This solution is not good as two components share a cluster.

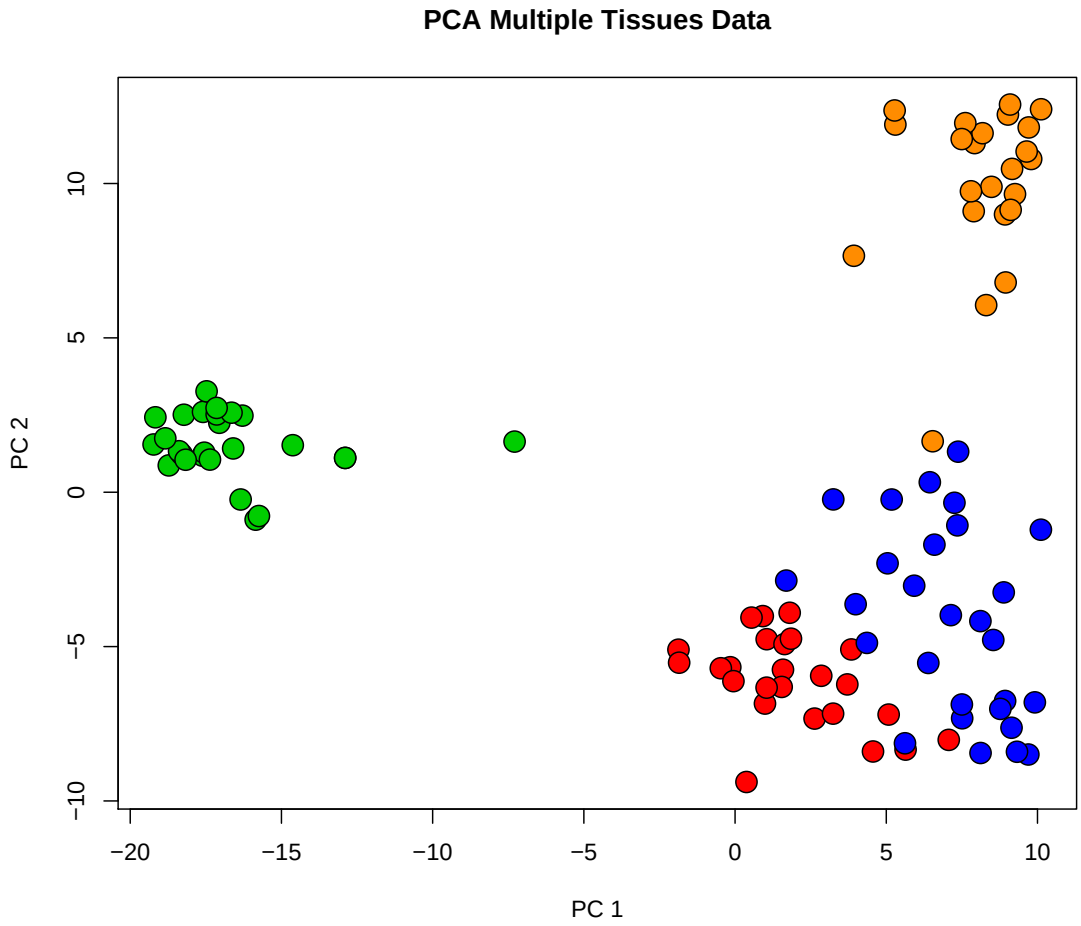


Figure 7.17: Down-projection of the multiple tissue data set to two dimensions by PCA. The true classes are marked by colors.

We apply k-means to the multiple tissues data set. To remind the reader, the down-projection onto two dimensions by PCA is again given in Fig. 7.17, where the true classes are marked by colors. For the down-projection the 101 features with the largest variance are used. k-means is applied to the full data set. Fig. 7.18 shows the typical solution of k-means applied to the multiple tissues data set. This solution appears in almost all cases. The classes are almost perfectly identified. Fig. 7.19 gives another solution of k-means for the multiple tissues data set. This solution is not as good as the solution in Fig. 7.18. Another suboptimal solution is shown in Fig. 7.20.

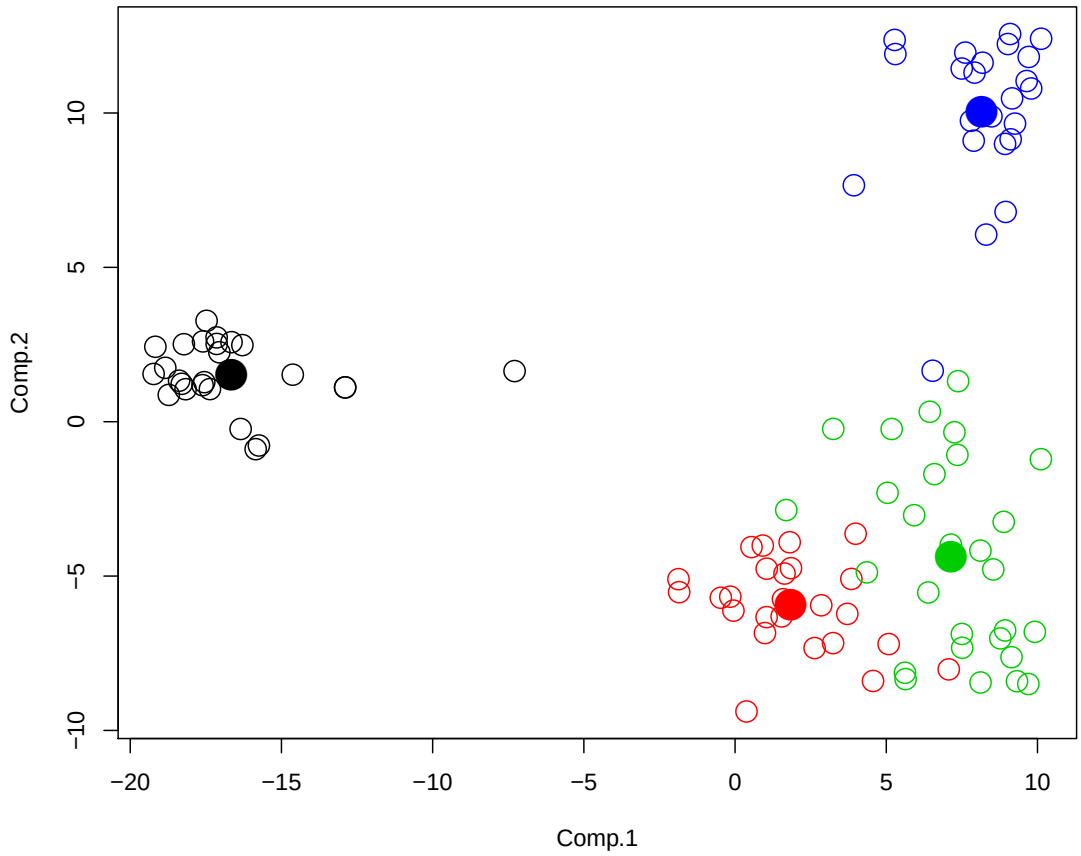


Figure 7.18: k-means clustering of the multiple tissue data set with $k = 4$. Filled circles are cluster centers. This is the solution found in almost all initializations. The classes are almost perfectly identified.

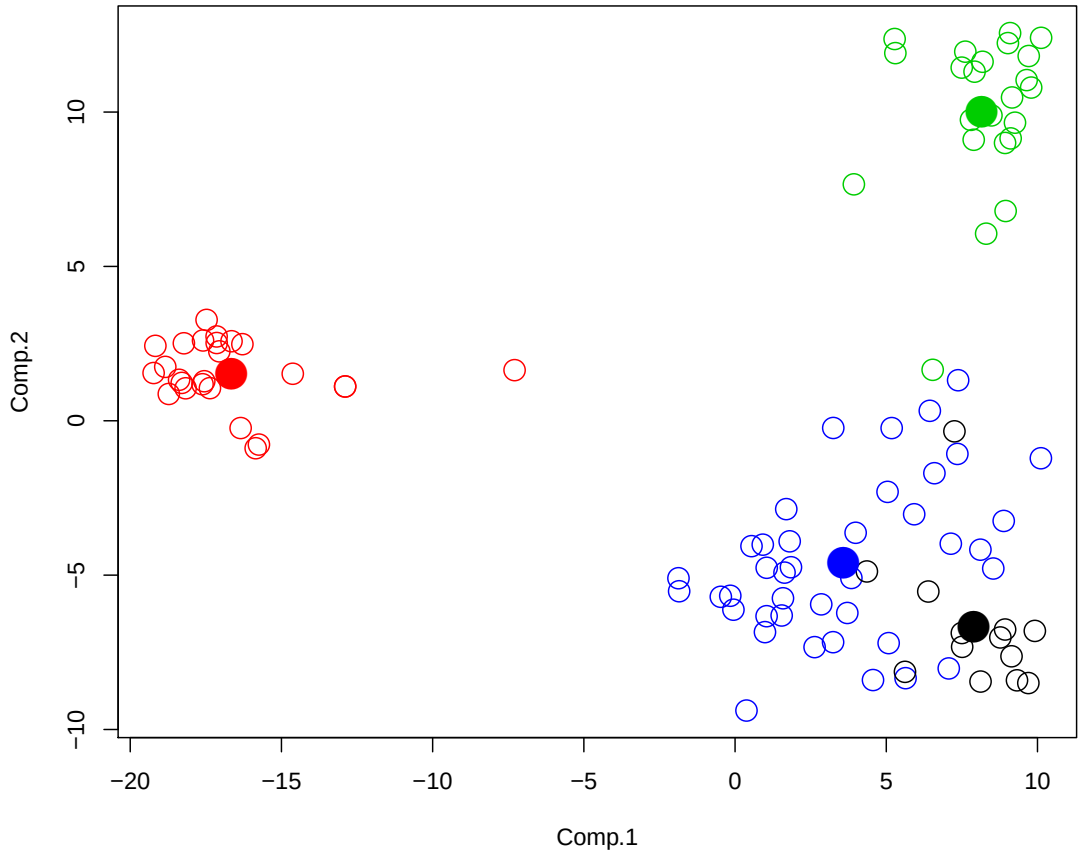


Figure 7.19: k-means clustering of the Iris data set with $k = 4$. This solution is not as good as the typical solution from Fig. 7.18.

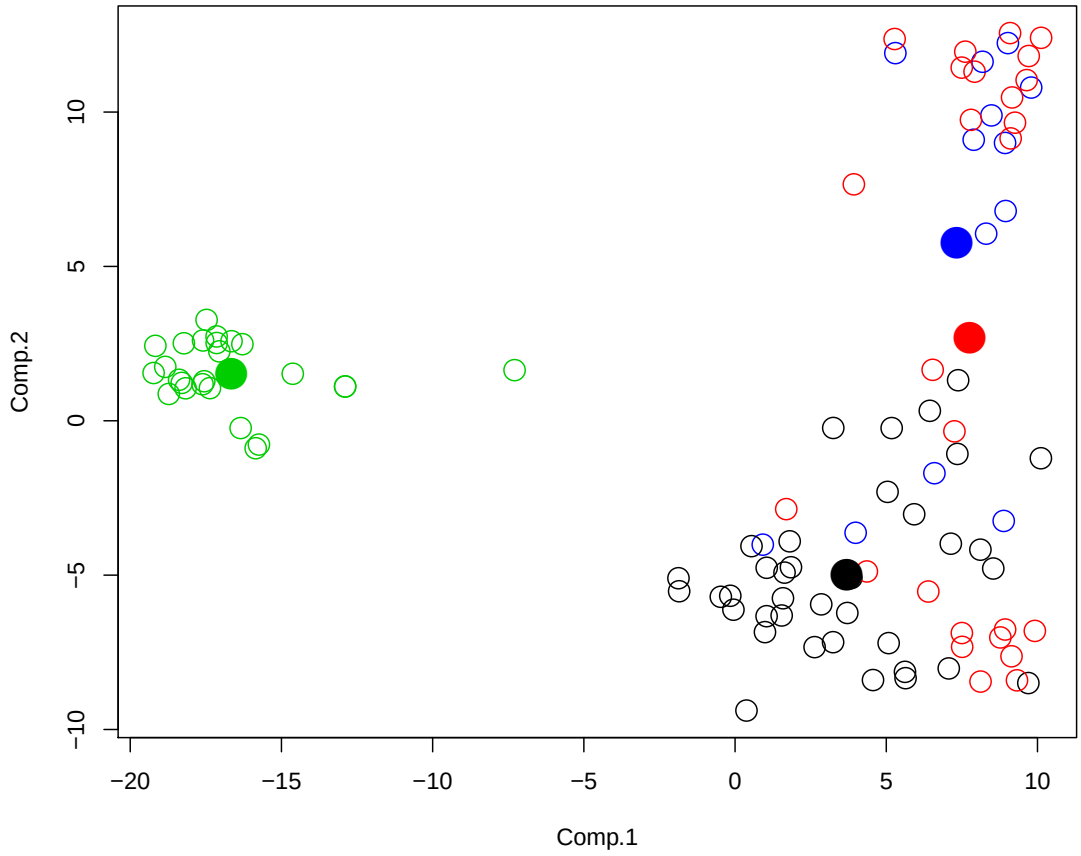


Figure 7.20: k-means clustering of the Iris data set with $k = 4$. Again, this solution is not as good as the typical solution from Fig. 7.18.

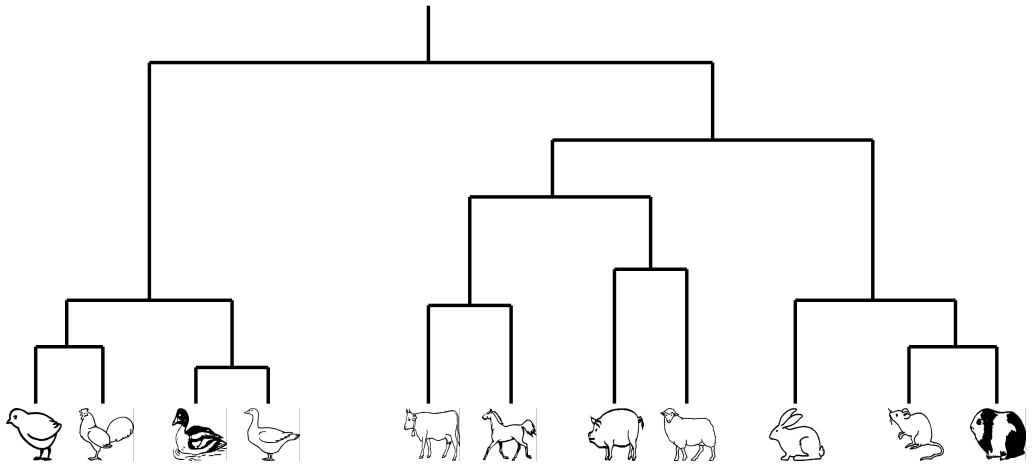


Figure 7.21: Example of hierarchical clustering of animal species where the result is given as a dendrogram (corresponding tree).

7.3 Hierarchical Clustering

7.3.1 The Method

So far we did not consider distances and structures between the clusters. Distances between clusters help to evaluate the clustering result and single clusters. In particular it would help to decide whether clusters should be merged or not. *Hierarchical clustering* supplies distances between clusters which are captured in a dendrogram. Fig. 7.21 depicts a dendrogram as the result of hierarchical clustering. Hierarchical clustering can be performed

- *agglomerative*, that is, *bottom up*, where the clustering starts with all clusters having a single observations and then clusters are merged until only one cluster remains
- *divisive*, that is, *top down*, where the clustering starts with one cluster and clusters are split until only clusters with a single observation remain.

In Bioinformatics the method "Unweighted Pair Group Method using arithmetic Averages" (UPGMA) applies hierarchical clustering in order to construct a phylogenetic tree. In machine learning the UPGMA method is called *agglomerative hierarchical clustering*, where the closest clusters are merged to give a new cluster. Agglomerative hierarchical clustering is initialized by clusters that consist of a single observation. Then clusters are iteratively merged until only one cluster remains.

Agglomerative hierarchical clustering can be used with different distance measures between clusters A and B:

$$\begin{array}{lll} d_{\min}(A,B) = \min_{\boldsymbol{a}\in A,\boldsymbol{b}\in B} \|\boldsymbol{a}-\boldsymbol{b}\| & \text{(single linkage)} \\ d_{\max}(A,B) = \max_{\boldsymbol{a}\in A,\boldsymbol{b}\in B} \|\boldsymbol{a}-\boldsymbol{b}\| & \text{(complete linkage)} \\ d_{\text{avg}}(A,B) = \frac{1}{n_A\, n_B} \sum_{\boldsymbol{a}\in A} \sum_{\boldsymbol{b}\in B} \|\boldsymbol{a}-\boldsymbol{b}\| & \text{(average linkage)} \\ d_{\text{mean}}(A,B) = \|\bar{\boldsymbol{a}}-\bar{\boldsymbol{b}}\| & \text{(average linkage)} \end{array}, \tag{7.79}$$

where n_A (n_B) is the number of elements in A (B) and $\bar{\boldsymbol{a}}$ ($\bar{\boldsymbol{b}}$) is the mean of cluster A (B). For the element distance $\|.\|$ any distance measure is possible like the Euclidean distance, the Manhattan distance, or the Mahalanobis distance.

For clusters with a single element these distance measures are equivalent, however for clusters with more than one element there is a difference.

- complete linkage $d_{\max}$ avoids that clusters are elongated in some direction, that is, the smallest distance between points may remains small. This means that the cluster may not be well separated.
- single linkage $d_{\min}$ ensures that each pair of elements, where one is from one cluster and the other is from another cluster, has a minimal distance. The result of single linkage guarantees that after a cut of the hierarchical clustering tree, the distance between clusters has a minimal value. In machine learning single linkage clustering is relevant for *leave-one-cluster-out* cross-validation. Leave-one-cluster-out cross-validation assumes that a whole new group of objects is unknown and left out. Therefore in the training set there is no object that is similar to a test set object. Leave-one-cluster-out cross-validation is known from protein structure prediction.
- average linkage d_{avg} is the "Unweighted Pair Group Method using arithmetic Averages" (UPGMA) method.

Instead of starting with clusters containing a single object *bottom up* clustering can start *top down*, that is, starting with a single cluster containing all objects. Such *divisive* or top down clustering methods are based on graph theoretic considerations. First the minimal spanning tree is built. Then the largest edge is removed which gives two clusters. Now the second largest edge can be removed and so on. It might be more appropriate to compute the average edge length within a cluster and find the edge which is considerably larger than other edges in the cluster. This means long edges are selected locally as an edge that does not fit to the cluster structure and not globally. At node level, the edge of each node can be determined which is considerably larger than other edges of this node. The inconsistent (considerably larger) edges can be removed stepwise and new clusters are produced.

7.3.2 Examples

We perform hierarchical clustering on the US Arrest data set. This data set contains statistics, in arrests per 100,000 residents, for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

A data consists of 50 observations with 4 features / variables:

- Murder: Murder arrests (per 100,000),
- Assault: Assault arrests (per 100,000),
- UrbanPop: Percent urban population,
- Rape: Rape arrests (per 100,000).

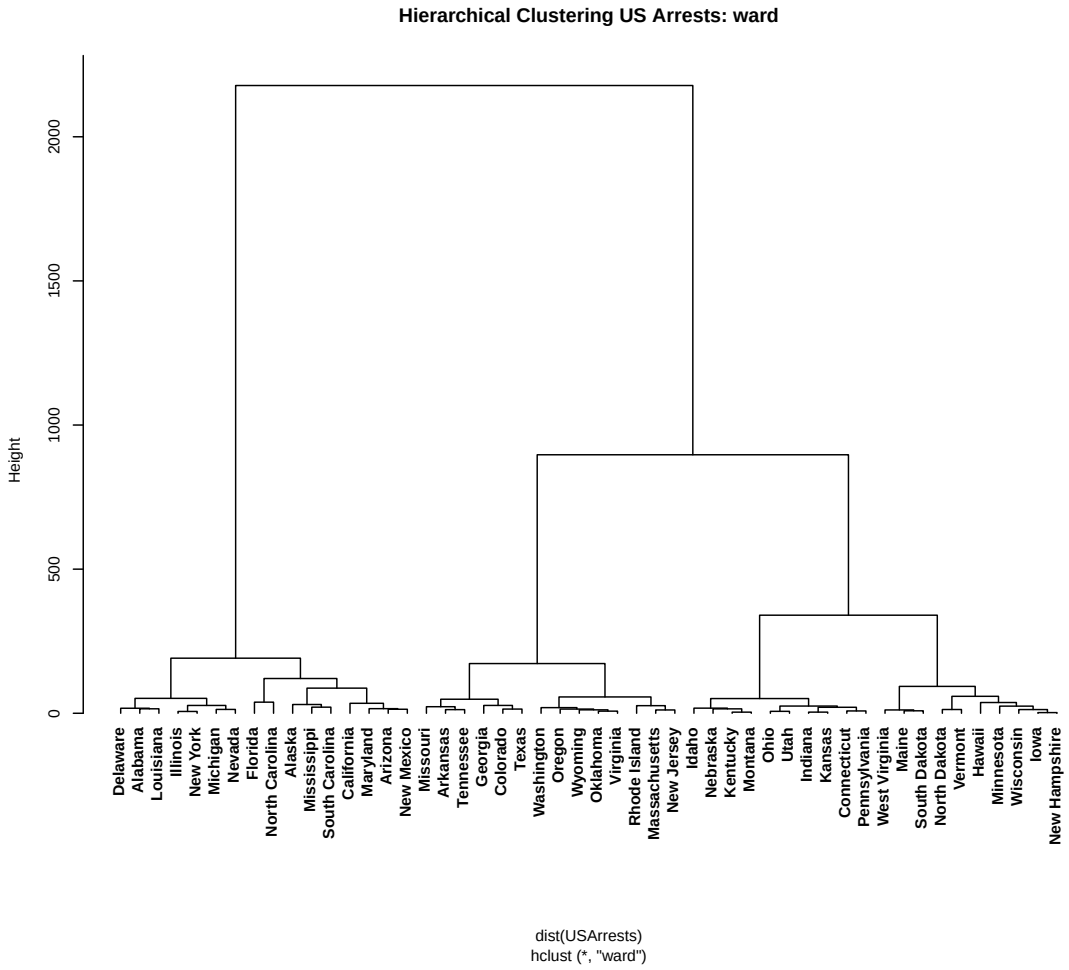


Figure 7.22: Hierarchical clustering of the US Arrest data set using Ward's minimal variance which gives compact, spherical clusters.

We test the distance measures "ward", "single", "complete", "average", "mcquitty", "median", and "centroid". Fig. 7.22 shows the results agglomerative hierarchical clustering using Ward's minimal variance as distance which gives compact, spherical clusters. Fig. 7.23 shows the results for single linkage which gives similar clusters with a minimal distance. Fig. 7.24 shows the results for complete linkage (minimal spanning tree) which is a "friends of friends" clustering. Fig. 7.25 shows the results for average linkage, which corresponds to UPGMA in bioinformatics (distance between averages of cluster elements). Fig. 7.26 shows the results for the McQuitty distance. Fig. 7.27 shows the results for median distance which is not a monotone distance measure. Fig. 7.28 shows the results for centroid distance which is also not a monotone distance measure.

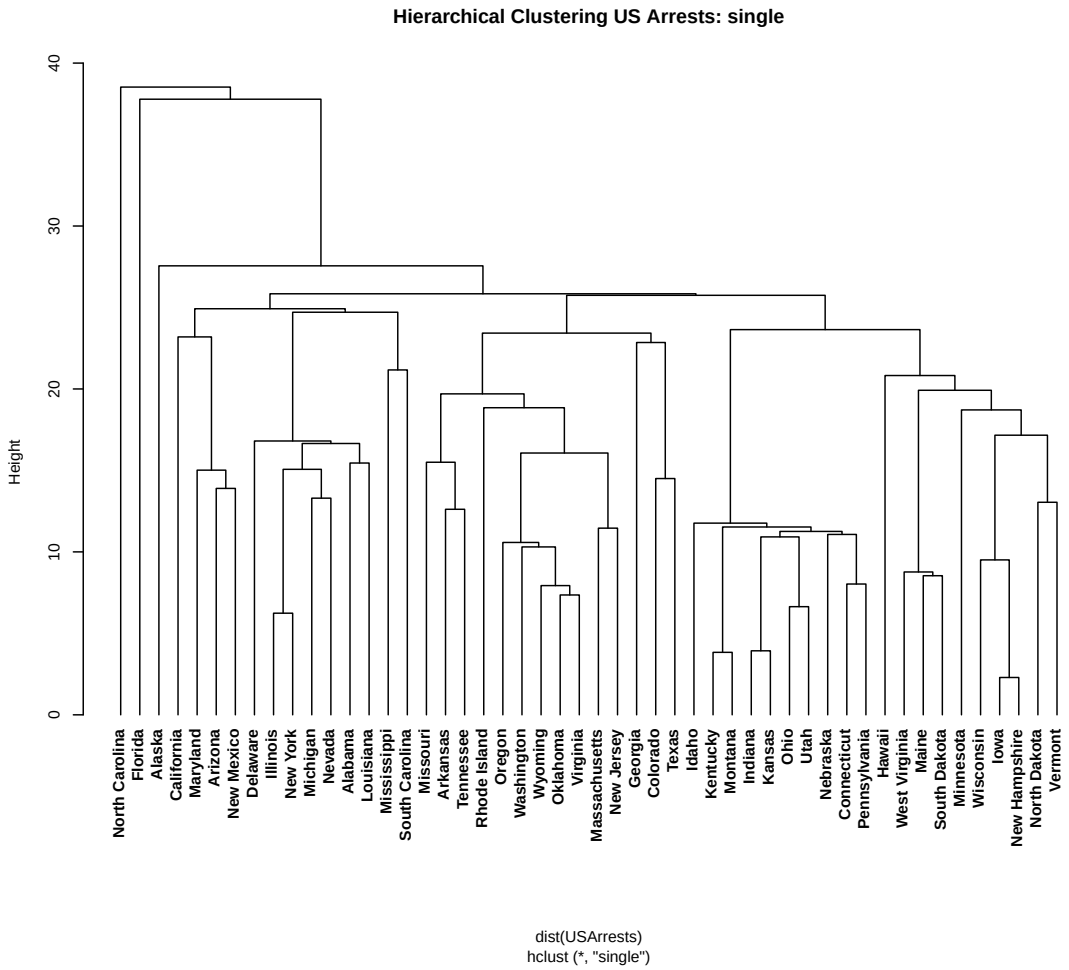


Figure 7.23: Hierarchical clustering of the US Arrest data set using single linkage which gives similar clusters with a minimal distance.

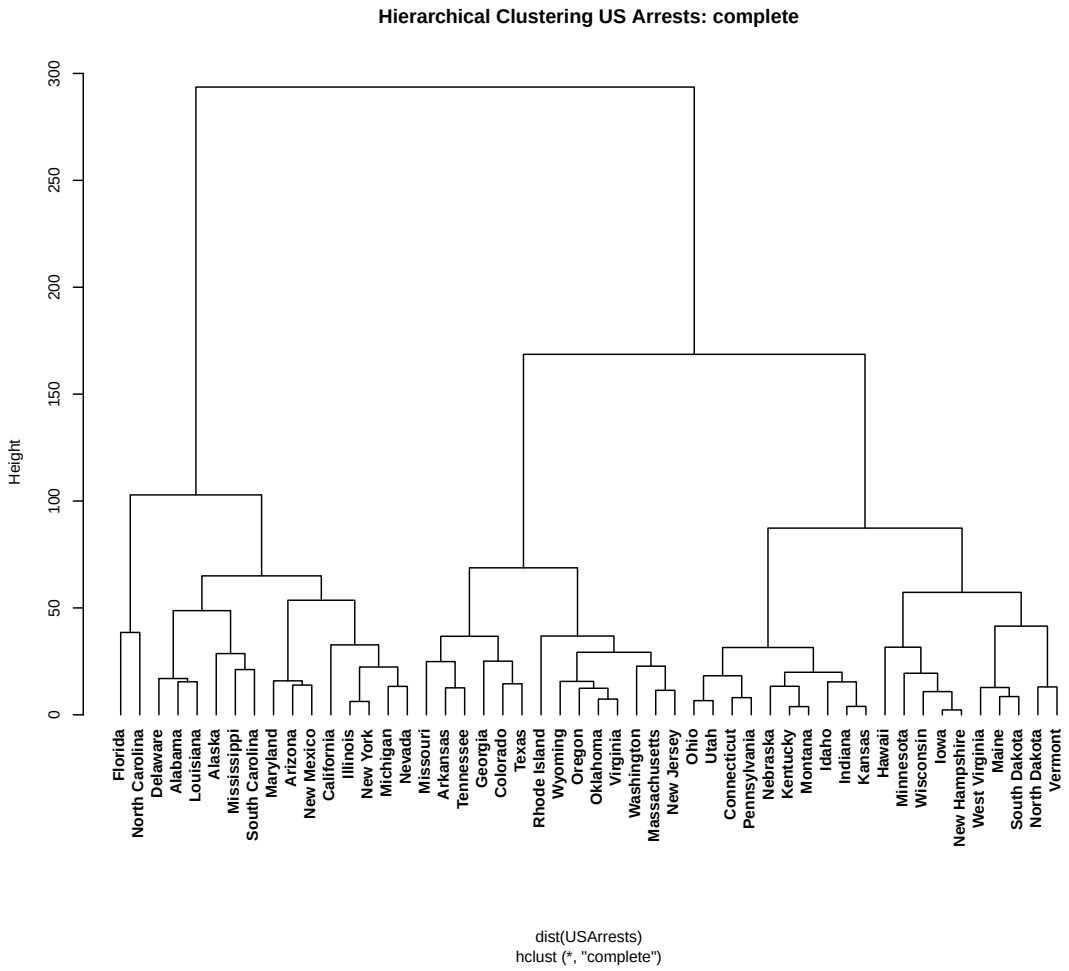


Figure 7.24: Hierarchical clustering of the US Arrest data set using complete linkage (minimal spanning tree) which is a “friends of friends” clustering.

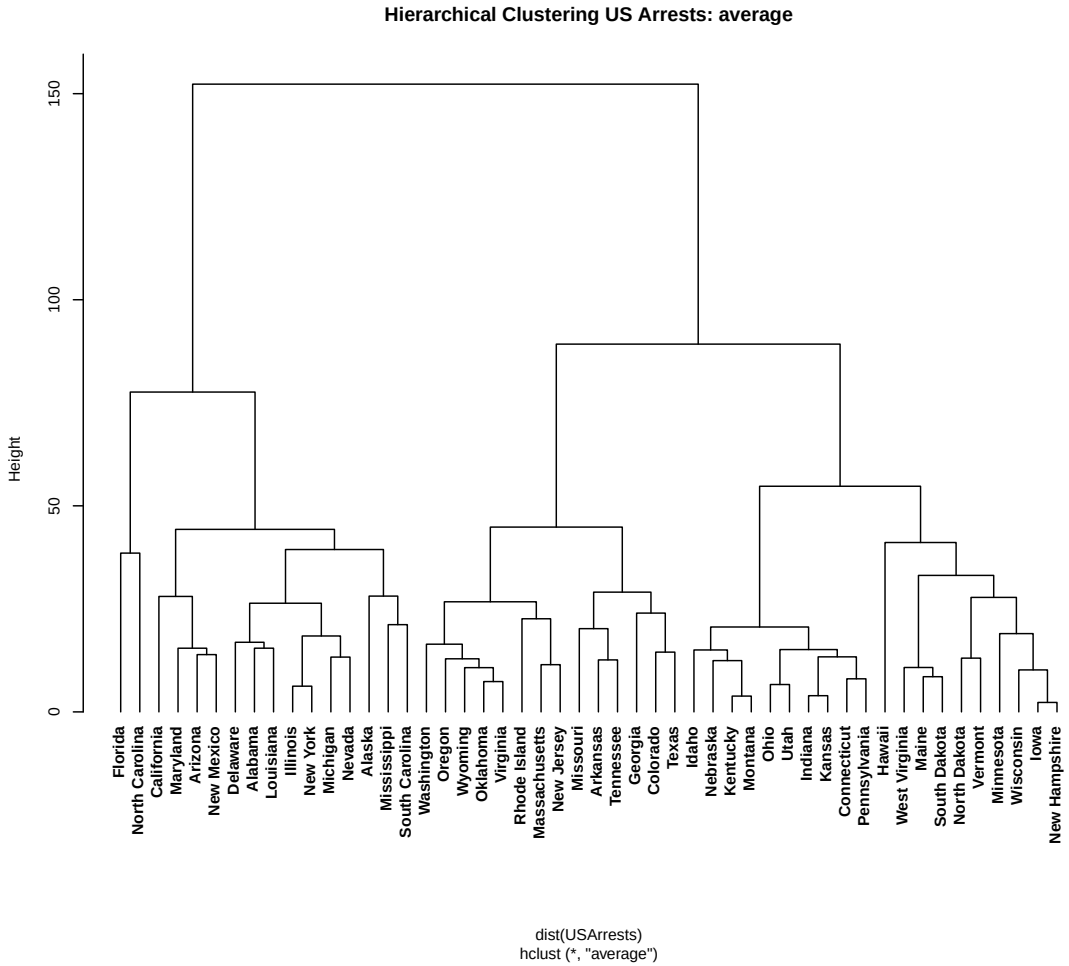


Figure 7.25: Hierarchical clustering of the US Arrest data set using average linkage.

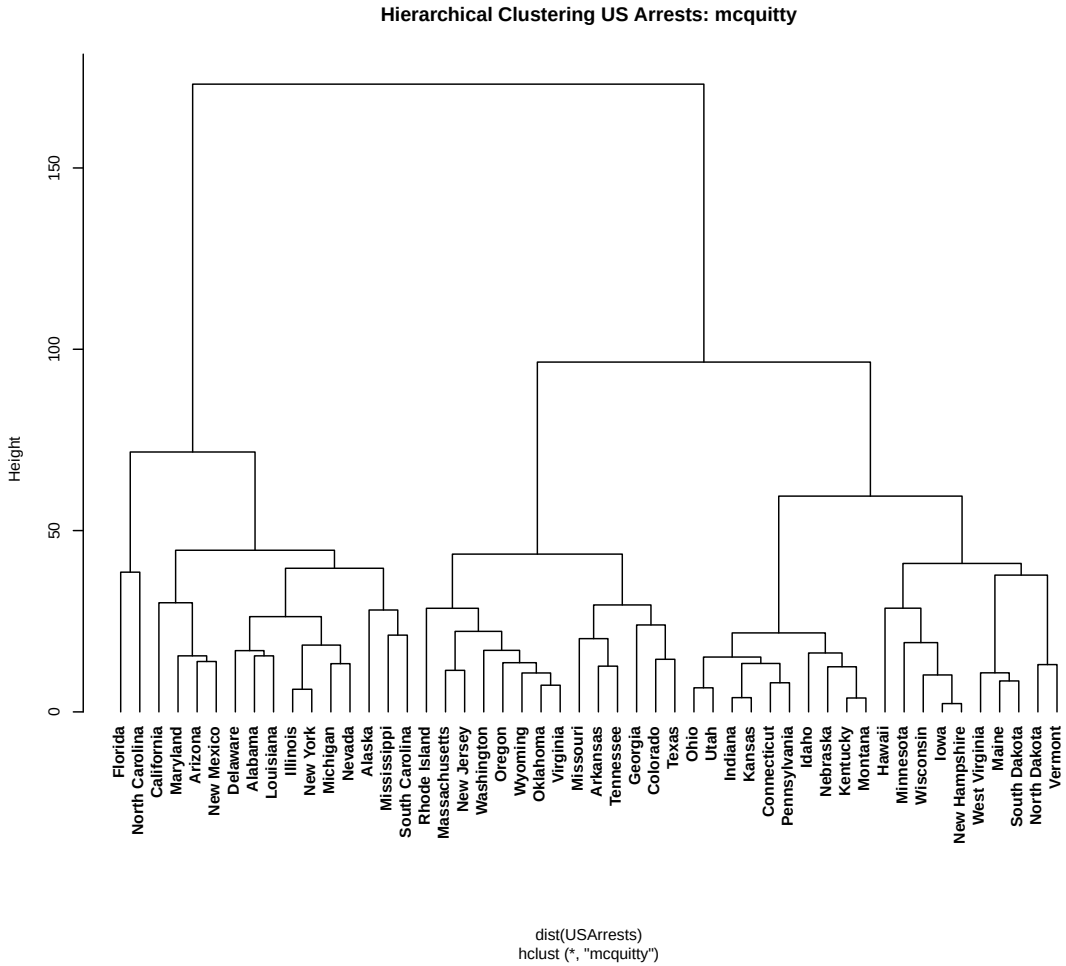


Figure 7.26: Hierarchical clustering of the US Arrest data set using McQuitty's distance.

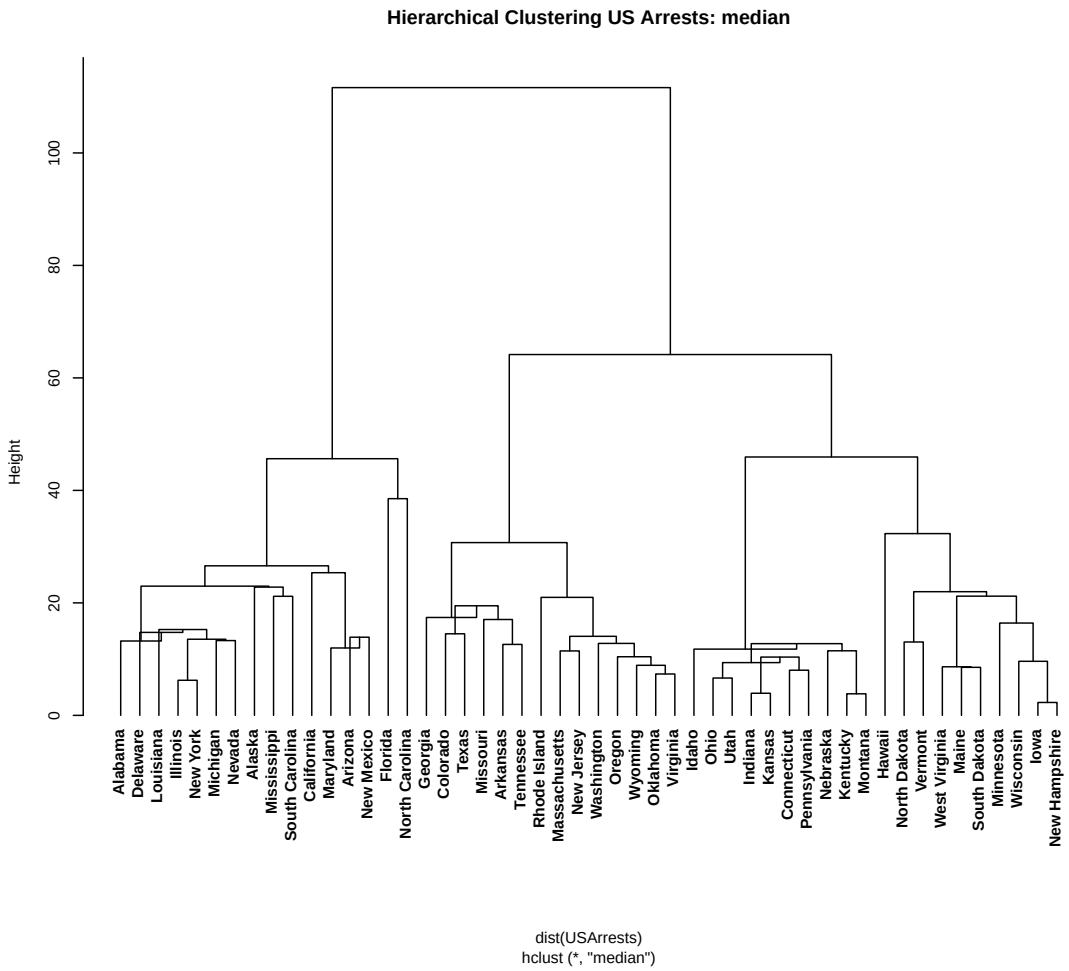


Figure 7.27: Hierarchical clustering of the US Arrest data set using median distance (not monotone).

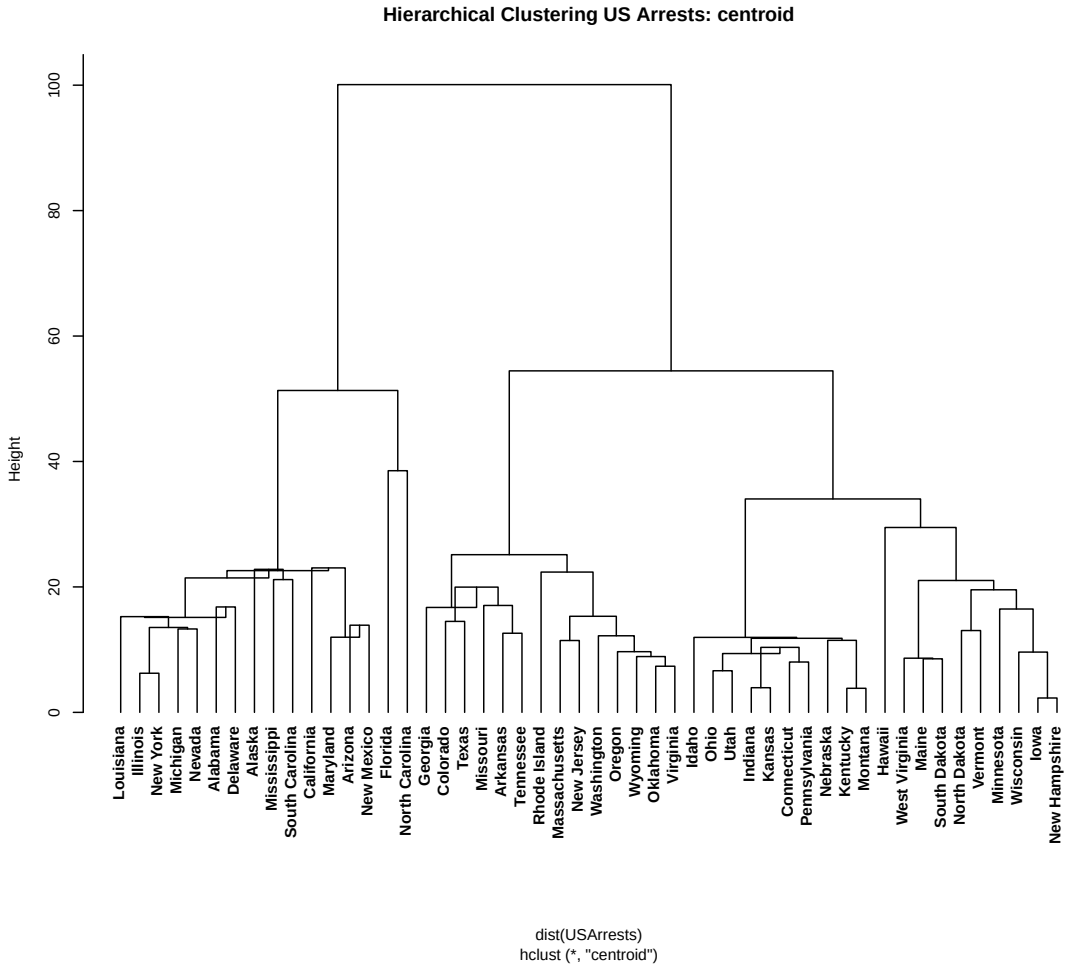


Figure 7.28: Hierarchical clustering of the US Arrest data set using centroid distance (not monotone).

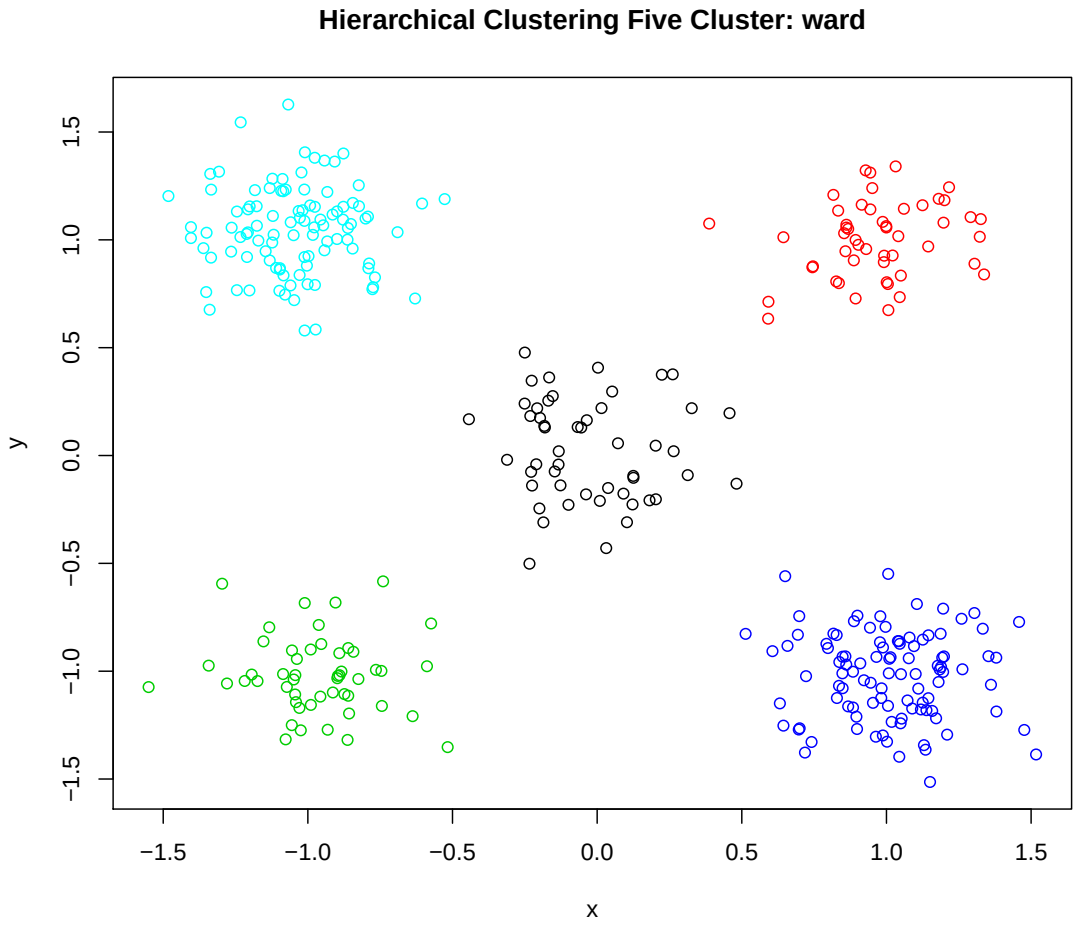


Figure 7.29: Hierarchical clustering of the five cluster data set. With all distance measures the optimal solution is found.

Next we apply hierarchical clustering to the five cluster data set. Fig. 7.29 shows the result for Ward's distance which is perfect. The results do not change if other distance measures than Ward's are used.

We apply hierarchical clustering to the iris data set. Fig. 7.30 shows the results for the distance measures Ward, average linkage, complete linkage, and single linkage for 3 and 5 components. Ward with 3 components performs well and average linkage with 3 components is worse. However, hierarchical clustering has problems to separate the close iris species. For 5 components either equal large cluster or small clusters are separated. Ward divides true clusters in equal large clusters while other methods separate small clusters. Single linkage separates out clusters with large distances to other clusters which do not reflect the true clusters.

Next we apply hierarchical clustering to the multiple tissue data set. Fig. 7.31 shows the results for the distance measures Ward, average linkage, complete linkage, and single linkage for 4 and 6 components. Ward with 4 components performs well. Again the correct number of clusters is essential to obtain good clustering results.

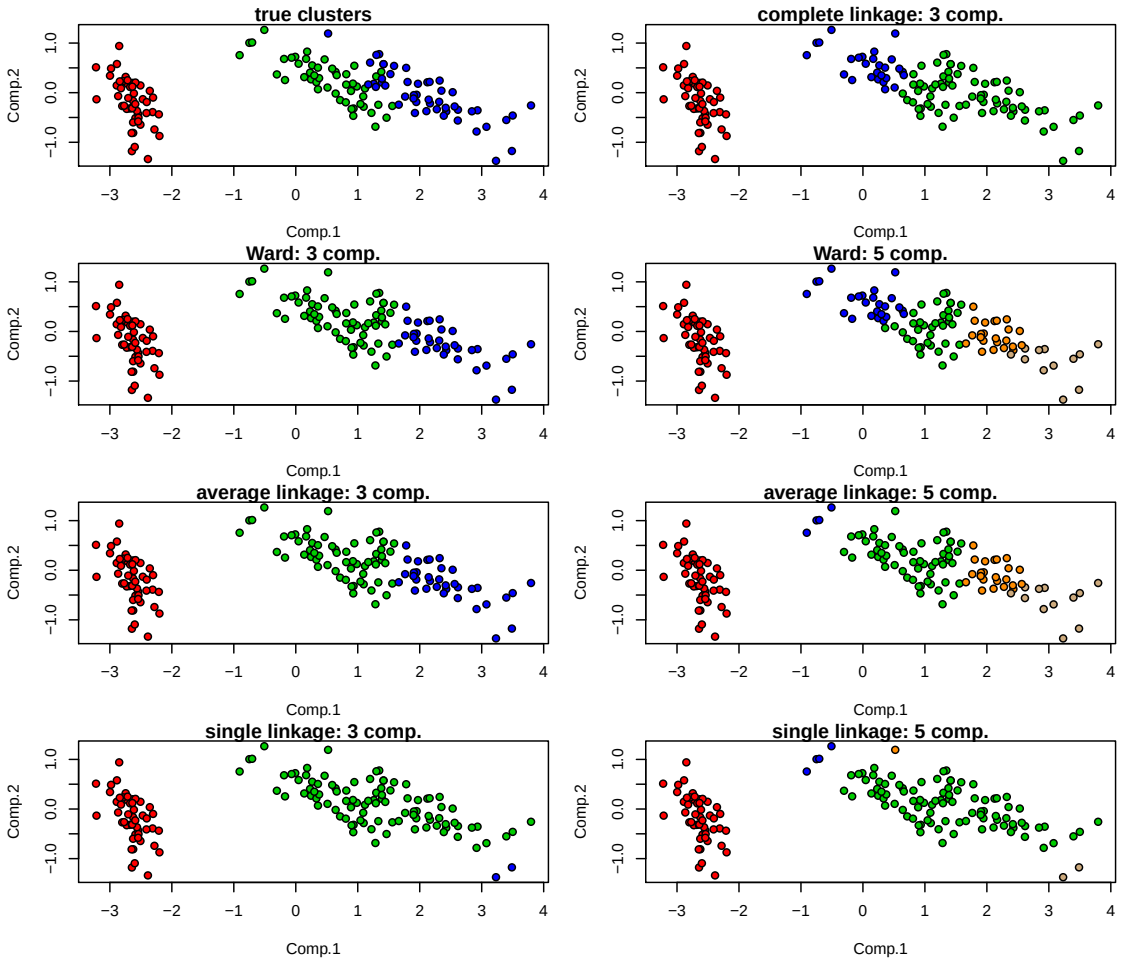


Figure 7.30: Hierarchical clustering of the iris data set. Ward with 3 components performs well and average linkage with 3 components is worse. For 5 components either equal large cluster or small clusters are separated.

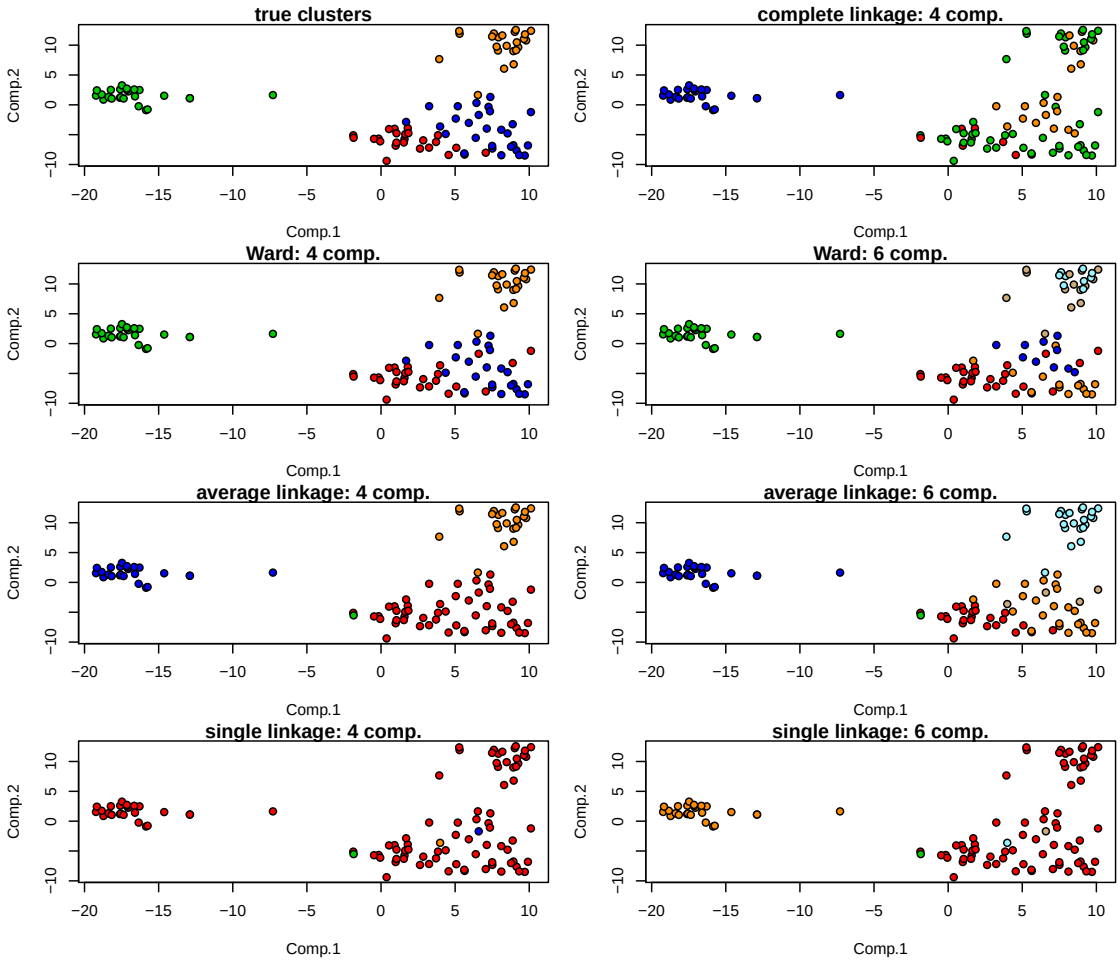


Figure 7.31: Hierarchical clustering of the multiple tissues data set. Ward with 4 components performs well. If correct number of cluster is known, the performance is better than with wrong number of clusters.

7.4 Similarity-Based Clustering

So far the distances which are required for clustering are computed from the representation of the objects by features. The feature vectors are embedded in a vector space in which distances between the vectors can be computed like the Euclidean distance, the Manhattan distance, or the Mahalanobis distance. However, many data sets are given by similarities between objects. The similarities may be links in the web domain, interactions of humans (facebook), co-occurrences of objects (co-expression of genes or co-citations), spacial distances between objects (cities on a map or atoms in a molecule), or co-processing (compressing two documents or sorting two sets). In bioinformatics a very prominent examples are the alignment of two sequences and the structural alignment of two proteins. Clustering, which only uses the similarities between objects but does not require to represent the objects via feature vectors, is called *similarity-based clustering*.

7.4.1 Aspect Model

Our first model, the *aspect model* Hofmann and Puzicha [1999], Hofmann et al. [1999], considers discrete data, where observations are pairs (x, y) taht are counted. That means the number of occurrences of x together with y are counted. Such data appear if relations like "person x buys product y" or "person x participates in y" are counted. Applications are found in information retrieval by document-word relations or in bioinformatics by sample-gene relations.

The aspect model assumes the model

$$p(x, y) \;=\; \sum_z p(z)\; p(x \mid z)\; p(y \mid z)\;, \tag{7.80}$$

where $z \in \{z_1, \ldots, z_l\}$ is the class variable and $p(x, y)$ is the probability of the observation (x, y). The underlying assumption is that x and y are independent conditioned on z:

$$p(x, y) \;=\; \sum_z p(x, y, z) \;=\; \sum_z p(z)\; p(x, y \mid z) \;=\; \sum_z p(z)\; p(x \mid z)\; p(y \mid z)\;. \tag{7.81}$$

$p(x \mid z)$ and $p(y \mid z)$ are the class conditional probabilities and $p(z)$ is the class prior probability. Therefore, observation (x, y) is observed more or less often than expected by random because a hidden factor z has an effect on the occurrence of both x and y. We already mentioned John Paulos' example in ABCNews.com:

> "Consumption of hot chocolate is correlated with low crime rate, but both are responses to cold weather."

Here x is "Consumption of hot chocolate", y is "crime rate", and z is "cold weather".

The maximum likelihood model parameters $p(x \mid z)$ and $p(y \mid z)$ can be estimated by an EM algorithm. The E-step is

$$p(z \mid x, y) \;=\; \frac{p(z)\; p(x \mid z)\; p(y \mid z)}{\sum_{z'} p(z')\; p(x \mid z')\; p(y \mid z')} \tag{7.82}$$

and the M-step is

$$p(z) = \sum_{x,y} n(x,y)\, p(z \mid x,y) \tag{7.83}$$

$$p(y \mid z) = \frac{\sum_x n(x,y)\, p(z \mid x,y)}{p(z)} \tag{7.84}$$

$$p(x \mid z) = \frac{\sum_y n(x,y)\, p(z \mid x,y)}{p(z)}\,, \tag{7.85}$$

where $n(x,y)$ is the number of observations (x,y). $n(x,y)$ is the entry in the data matrix located at the row of x and the column of y. For the updates only the counts $n(x,y)$ are required, therefore, we view the aspect model as similarity-based clustering.

Clustering of the x can be based on

$$p(z \mid x) = \frac{p(x \mid z)\, p(z)}{p(x)}\,, \tag{7.86}$$

$$p(x) = \sum_{z,y} p(z)\, p(x \mid z)\, p(y \mid z)\,. \tag{7.87}$$

z indicates the cluster, that is, each z represents one cluster. Analog formulas are obtained for clustering y or pairs (x,y).

7.4.2 Affinity Propagation

7.4.2.1 The Method

Mixture clustering and k-means cluster continuous data such that cluster members are similar to the cluster center. If the cluster centers are actual data points, they are called "prototypes" or "exemplars". Clustering that enforces cluster centers to be data points is called *exemplar-based clustering*. A popular technique of exemplar-based clustering is the k-centers clustering MacQueen [1967]. It starts with an initial set of randomly selected exemplars and iteratively refines this set so as to decrease the sum of squared errors. However k-centers is only applicable for a small number of clusters where the chances are high that a good initialization leads to a good solution.

A method that overcomes the problems of k-centers and showed good performance in different applications is *Affinity propagation* Frey and Dueck [2006, 2007], Givoni and Frey [2009]. Affinity propagation is both a similarity-based and an exemplar-based clustering method. The similarities between object i and object k are given by $s(i,k)$. The values $s(k,k)$ are called "preferences" and used to determine how likely object k becomes an exemplar. The larger $s(k,k)$, the more likely object k becomes an exemplar.

The values $r(i,k)$ are called "responsibilities" which are messages sent from object i to candidate exemplar k. The responsibility reflects the evidence that k serves as an exemplar for i. In other words, how well can k represent the object i. The responsibility takes into account how well other candidate exemplars can represent object i (competition between candidate exemplars). Here i summarizes the environment for better exemplars.

The values $a(i,k)$ are called "availabilities" which are messages sent from candidate exemplar k to object i. The availability reflects the evidence that k is indeed an exemplar. The availability $a(i,k)$ takes into account how many objects are already represented by candidate exemplar k and what is the input preference of k.

The affinity propagation algorithm starts with initializing:

$$a(i,k) \;=\; 0\;. \tag{7.88}$$

The responsibilities are updated using

$$r(i,k) \;=\; s(i,k) \;-\; \max_{k',k'\neq k}\{a(i,k') \;+\; s(i,k')\}\;. \tag{7.89}$$

The availabilities are updated using

$$a(i,k) \;=\; \min\left\{0\;,\; r(k,k) \;+\; \sum_{i',i'\notin\{i,k\}} \max\{0\;,\; r(i',k)\}\right\} \tag{7.90}$$

$$a(k,k) \;=\; \sum_{i',i'\neq k} \max\{0\;,\; r(i',k)\}\;. \tag{7.91}$$

$a(k,k)$ is the evidence that k is an exemplar based on the positive responsibilities sent from objects i.

Fig. 7.32 shows how affinity propagation sends messages. Fig. 7.33 shows the specific messages which are passed on in the algorithm of affinity propagation.

For more detail on affinity propagation, see the homepage `http://www.psi.toronto.edu/index.php?q=affinity%20propagation`. To apply affinity propagation to your own data, it is convenient to use the R package `APCluster` which contains useful visualizations and extensions Bodenhofer et al. [2011].

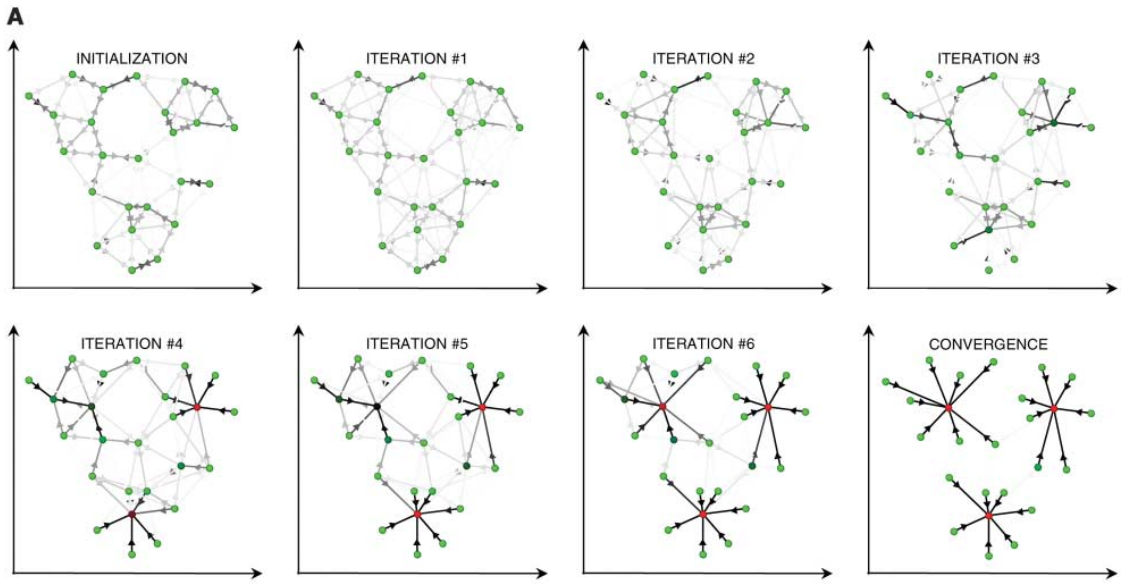


Figure 7.32: Affinity propagation is illustrated for two-dimensional data points, where negative Euclidean distance (squared error) was used to measure similarity. Each point is colored according to the current evidence that it is a cluster center (exemplar). The darkness of the arrow directed from point i to point k corresponds to the strength of the transmitted message that point i belongs to exemplar point k. Figure from Frey and Dueck [2007].

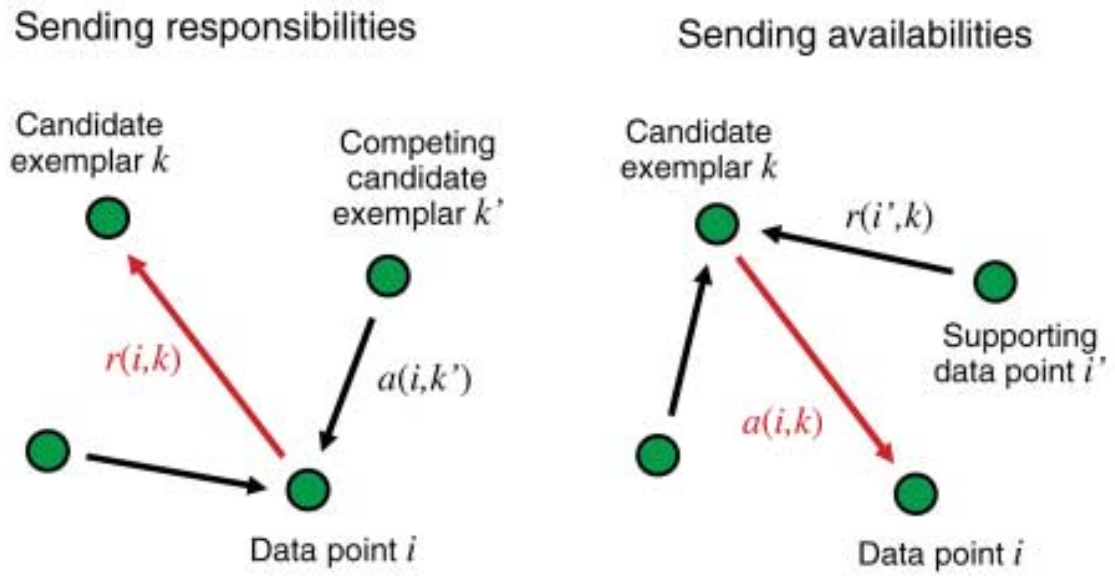


Figure 7.33: Left: "Responsibilities" $r(i,k)$ are sent from data points to candidate exemplars and indicate how strongly each data point favors the candidate exemplar over other candidate exemplars. Right: "Availabilities" $a(i,k)$ are sent from candidate exemplars to data points and indicate to what degree each candidate exemplar is available as a cluster center for the data point. Figure from Frey and Dueck [2007].

Figure 7.34: Affinity propagation: faces. The 15 images with highest squared error under either affinity propagation or k-centers clustering are shown in the top row. The middle and bottom rows show the exemplars assigned by the two methods, and the boxes show which of the two methods performed better for that image, in terms of squared error. Affinity propagation found higher-quality exemplars. Figure from Frey and Dueck [2007].

7.4.2.2 Examples

Fig. 7.34 shows an example of affinity propagation with faces. Fig. 7.35 shows results of affinity propagation at identifying key sentences and and at air-travel routing. Fig. 7.36 shows another example of affinity propagation applied to face images.

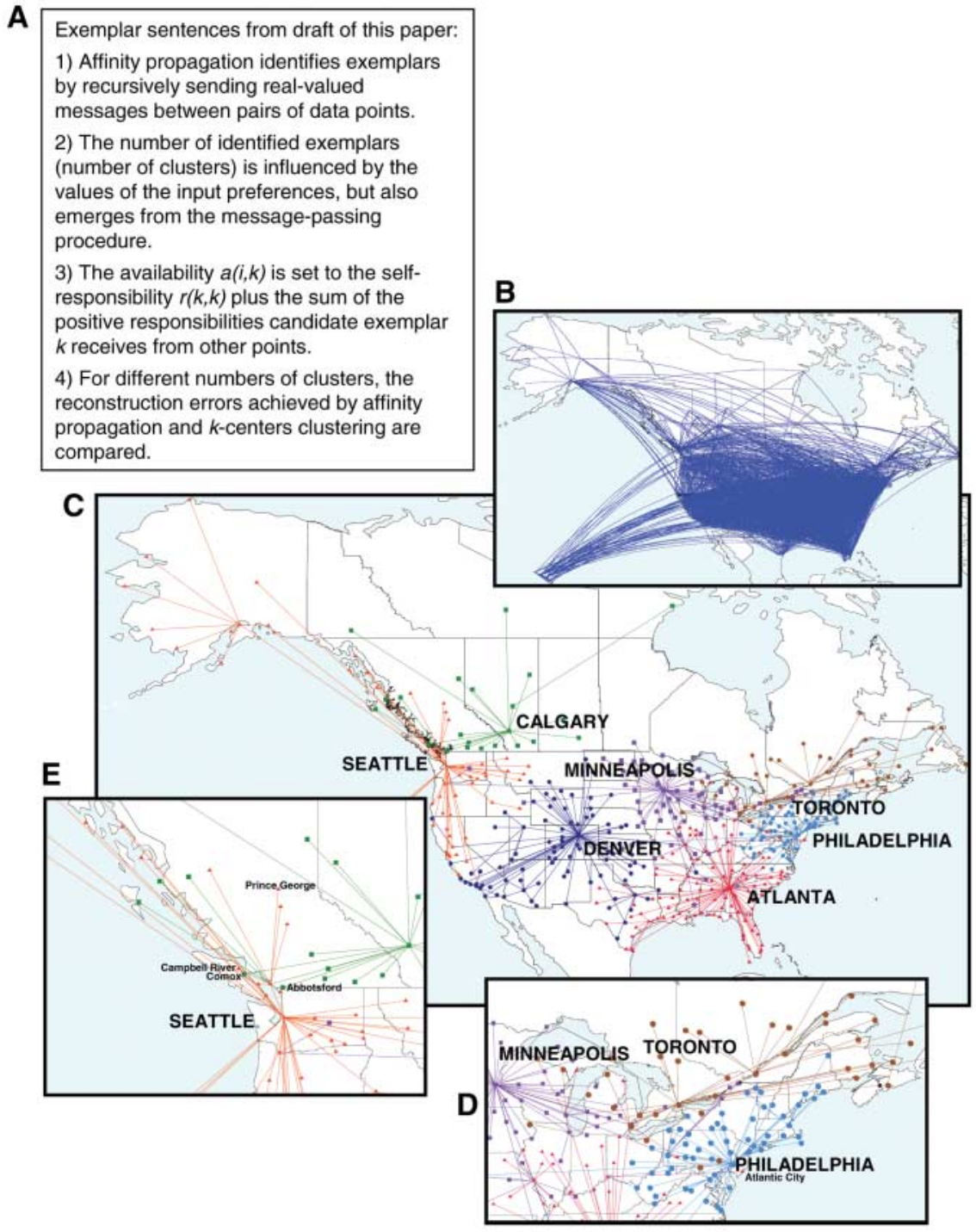


Figure 7.35: Affinity propagation for identifying key sentences and air-travel routing. **(A)** Similarities between pairs of sentences in a draft of this manuscript were constructed by matching words. Four identified exemplar sentences are shown. **(B)** Affinity propagation was applied to similarities derived from air-travel efficiency (estimated travel time) between the 456 busiest commercial airports in Canada and the United States. **(C)** Seven exemplars identified by affinity propagation are color-coded, and the assignments of other cities to these exemplars is shown. Cities located quite near to exemplar cities may be members of other more distant exemplars due to the lack of direct flights between them. **(D)** The inset shows that the Canada-USA border roughly divides the Toronto and Philadelphia clusters, due to a larger availability of domestic flights vs. international flights. The west coast is shown in **(E)**, where extraordinarily frequent airline service between Vancouver and Seattle connects Canadian cities in the northwest to Seattle. Figure from Frey and Dueck [2007].

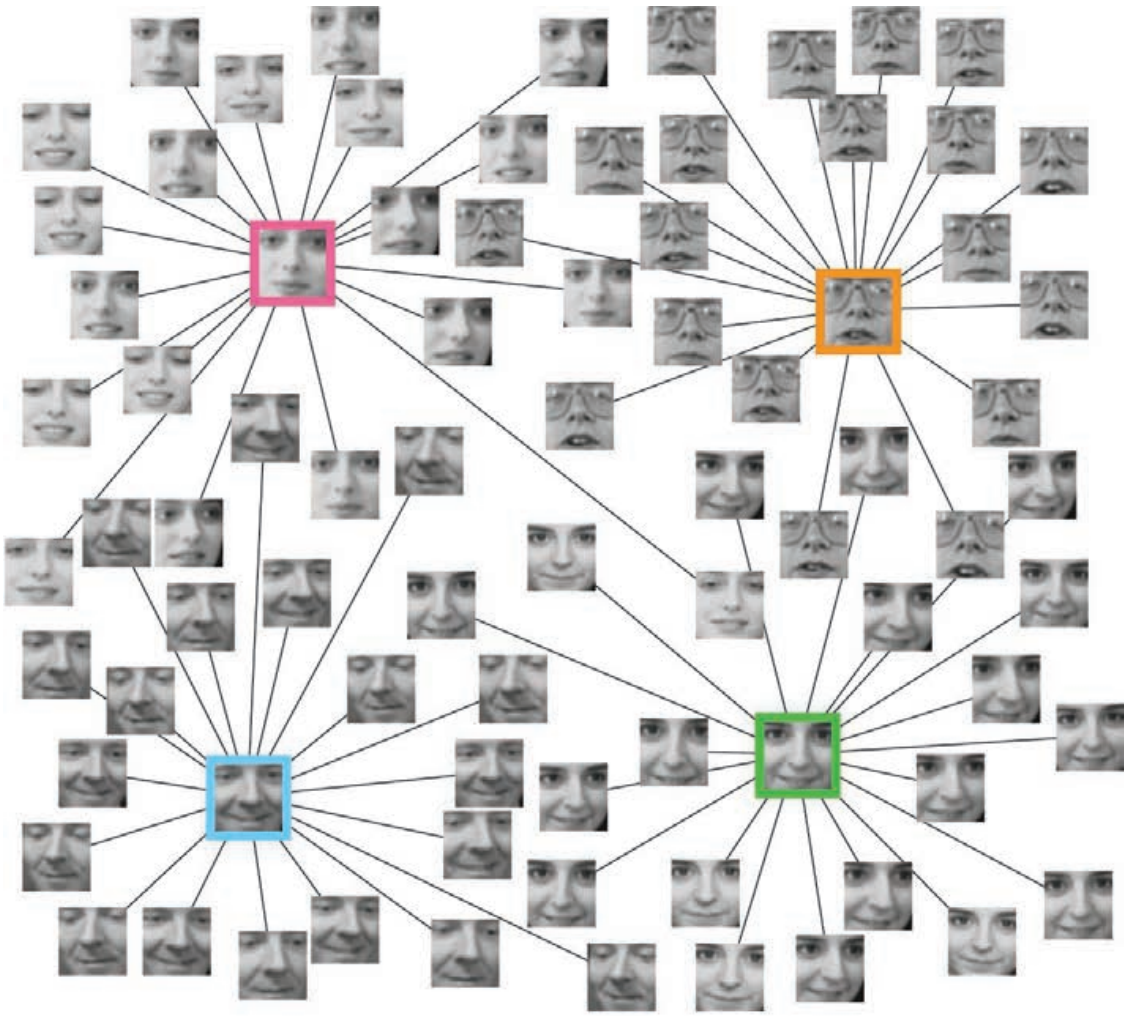


Figure 7.36: Affinity propagation: faces (2). Exemplars (highlighted by colored boxes) have been detected from a group of faces by affinity propagation. Figure from Frey and Dueck [2007].

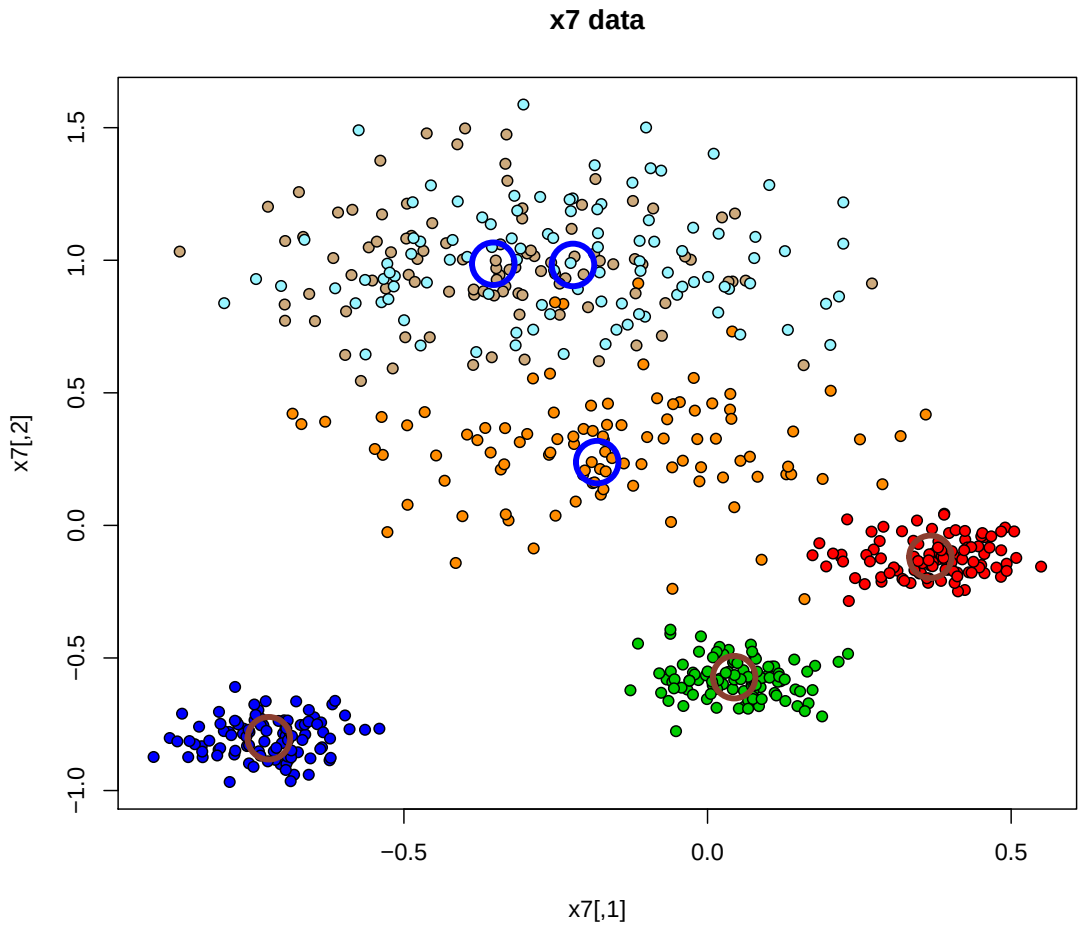


Figure 7.37: The x7 data set. 6 clusters are indicated by color coding of which 3 are smaller than the other 3. Circles indicate the centers of the clusters.

To demonstrate how affinity propagation clusters observations, we create a data set with 6 clusters in a two-dimensional space. Three clusters are smaller than the other three clusters, that is, they have a smaller variance. This data set is depicted in Fig. 7.37 where the cluster centers are indicated by circles.

From this data set we computed similarities by the Euclidean distance. We cluster the x7 data set by affinity propagation. The result of affinity propagation is shown in Fig. 7.38. Circles indicate the true cluster centers and black rectangles the exemplars found by affinity propagation. The result is quite good. The two overlapping large clusters are identified. Affinity propagation implicitely prefers spherical clusters, therefore the two large clusters are detected. Another version of this kind of data generation is shown in Fig. 7.39 and the result of affinity propagation in Fig. 7.40. At the upper right, a large and a small cluster are wrongly merged. AP assumes sphered equal large clusters, therefore cannot detect the small cluster within the large cluster.

The x9 data set contains also 6 clusters of which 2 are small, 2 are medium sized, and 2 are large with respect to the variance. Fig. 7.41 shows this data set. The result of affinity propagation

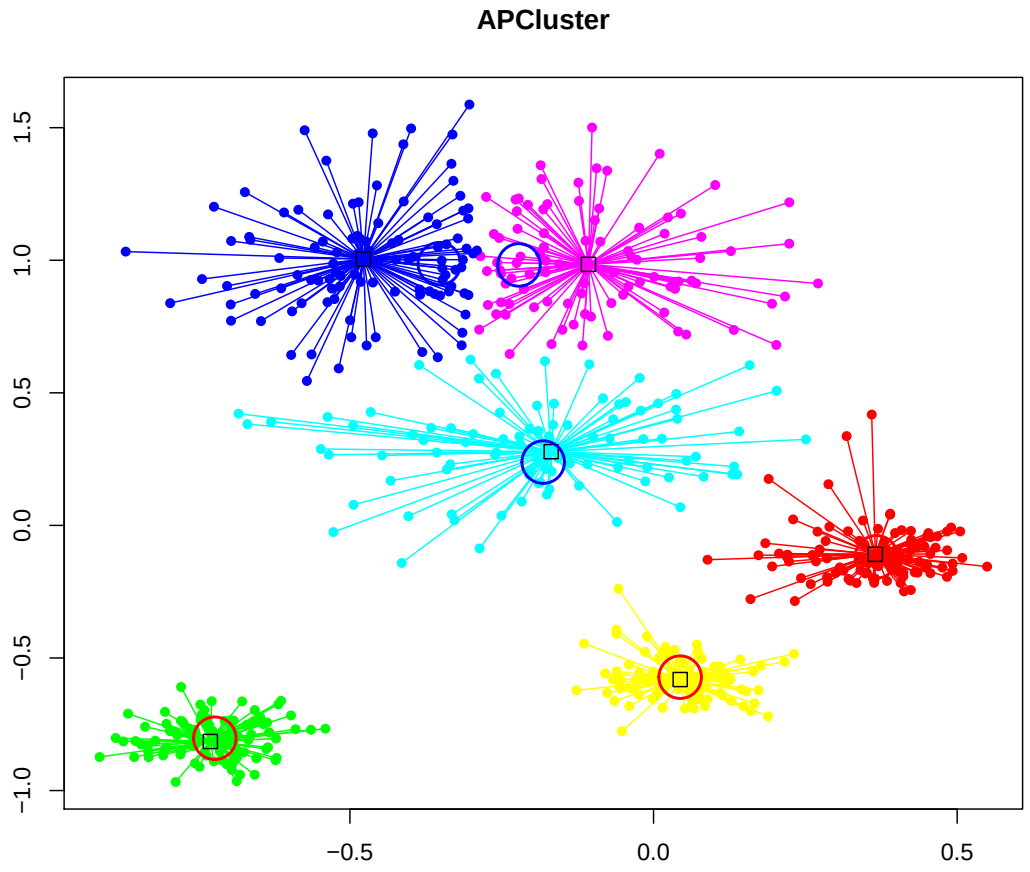


Figure 7.38: Affinity propagation applied to the x7 data set. Circles indicate the true cluster centers and black rectangles the exemplars found by affinity propagation. The result is quite good. The two overlapping large clusters are identified.

is depicted in Fig. 7.42. One cluster could not be separated into the two true clusters.

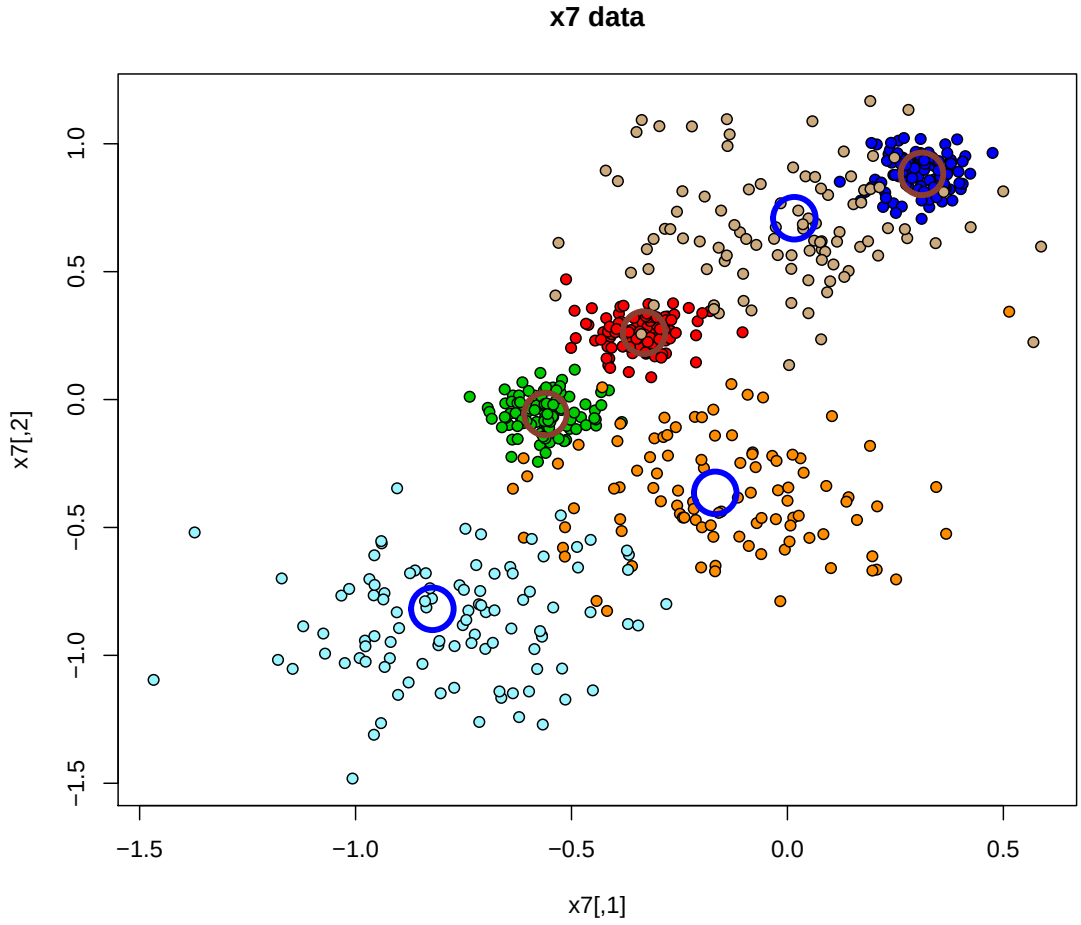


Figure 7.39: The x7A data set. 6 clusters are indicated by color coding of which 3 are smaller than the other 3. Circles indicate the centers of the clusters.

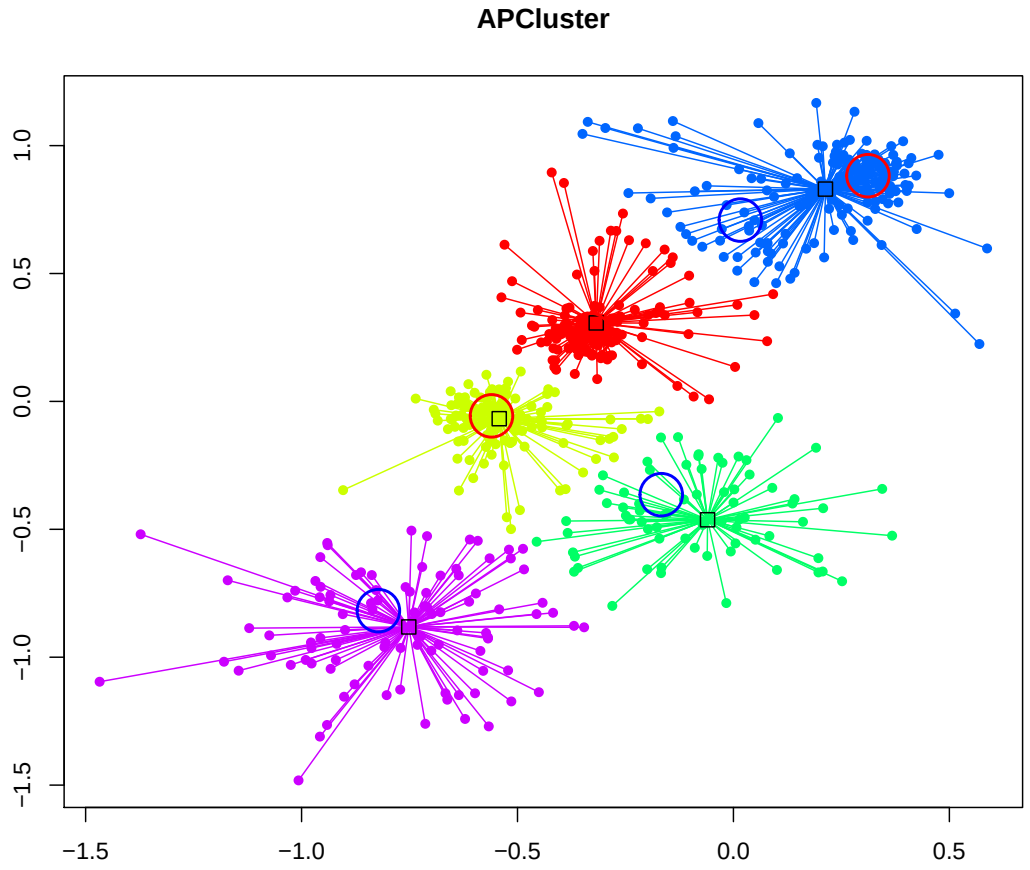


Figure 7.40: Affinity propagation applied to the x7A data set. Circles indicate the true cluster centers and black rectangles the exemplars found by affinity propagation. At the upper right, a large and a small cluster are wrongly merged. AP assumes sphered equal large clusters, therefore cannot detect the small cluster within the large cluster.

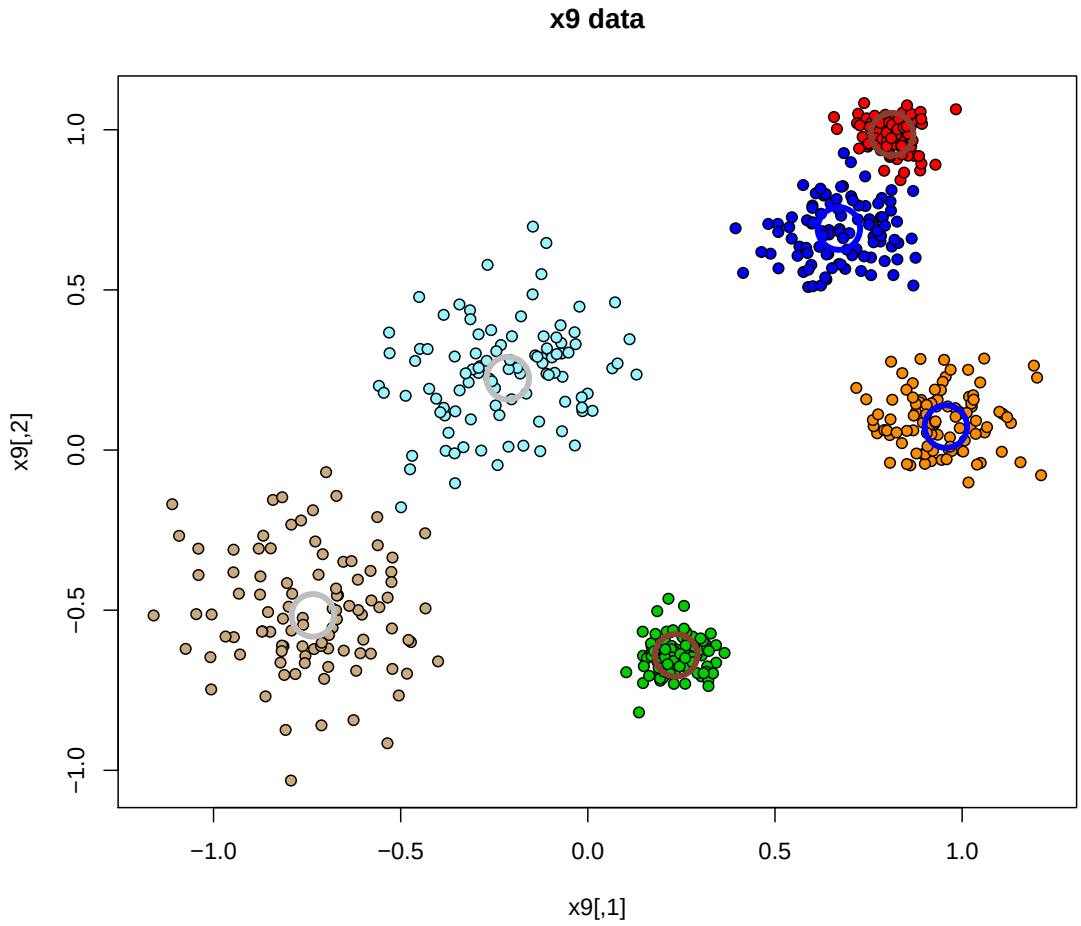


Figure 7.41: The x9 data set. 6 clusters are indicated by color coding of which 2 are small, 2 a medium, and 2 are large. Circles indicate the centers of the clusters.

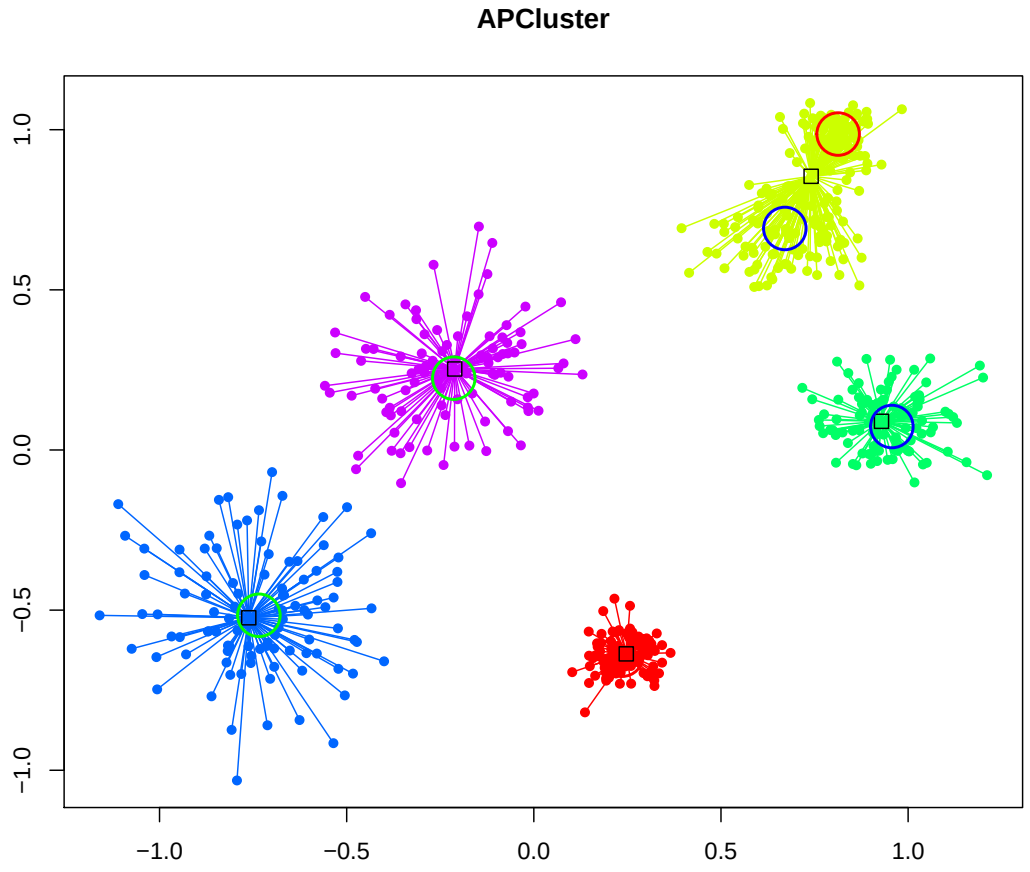


Figure 7.42: Affinity propagation applied to the x9 data set. Circles indicate the true cluster centers and black rectangles the exemplars found by affinity propagation. One cluster could not be separated into the two true clusters.

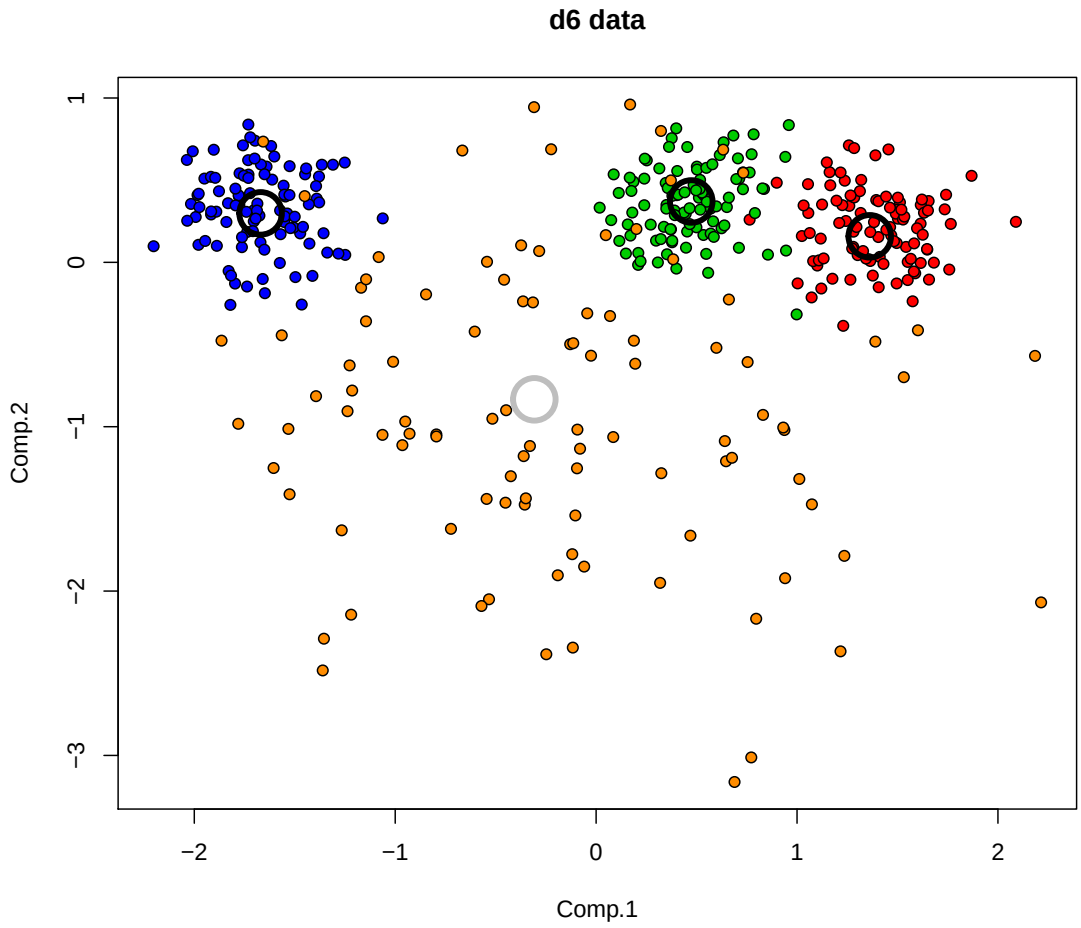


Figure 7.43: The d6 data set. 4 clusters in a 6-dimensional space, where one cluster is larger then the other three clusters.

Next we test affinity propagation on a 6-dimensional data set with different cluster sizes with respect to the variation of the elements in the clusters. We create one large cluster and three smaller clusters. For visualization, we project the data down by PCA to visualize the data. We generated two data sets which are depicted in Fig. 7.43 and Fig. 7.45 where the cluster centers are indicated by circles. The result is given in Fig. 7.44 where again circles indicate the true cluster centers and black rectangles the exemplars found by affinity propagation. Affinity propagation is not able to detect the large cluster. Elements of the large cluster are assigned to the smaller cluster. AP has problems with the different cluster sizes because it cannot adjust the variance. The same result for another data set (see Fig. 7.45) can be seen in Fig. 7.46. Affinity propagation does not adjust the variance therefore, again the data points of the large cluster are assigned to the smaller clusters. Equally sized clusters makes affinity propagation on the one hand very robust, however on the other hand it is AP's weakness if different cluster sizes are present.

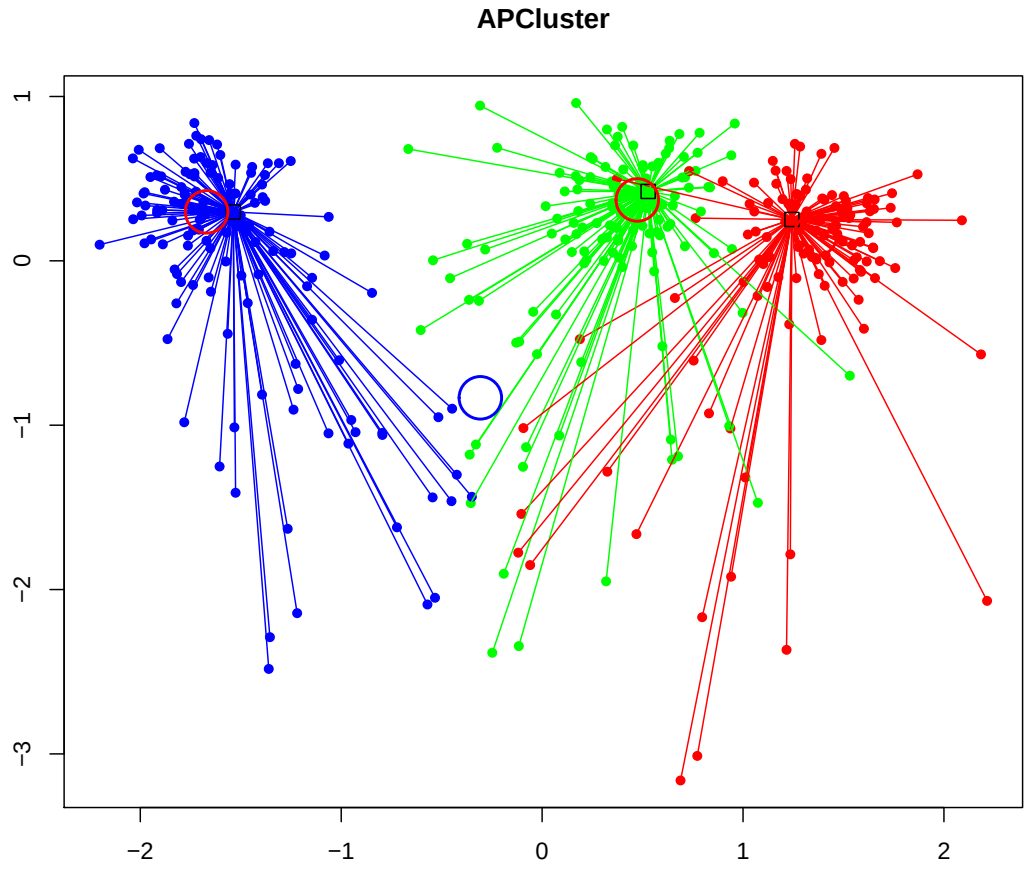


Figure 7.44: Affinity propagation applied to the d6 data set. Circles indicate the true cluster centers and black rectangles the exemplars found by affinity propagation. Affinity propagation is not able to detect the large cluster.

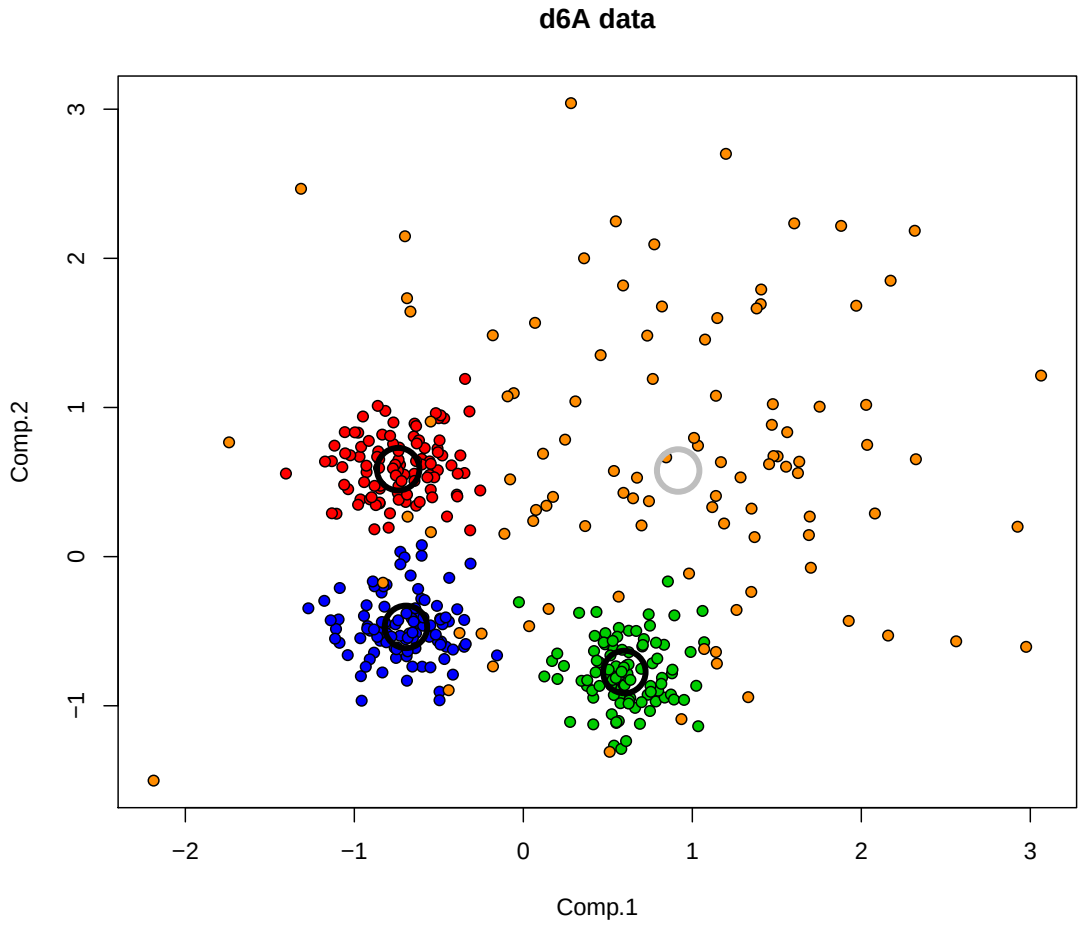


Figure 7.45: The d6A data set. Another variant of Fig. 7.43.

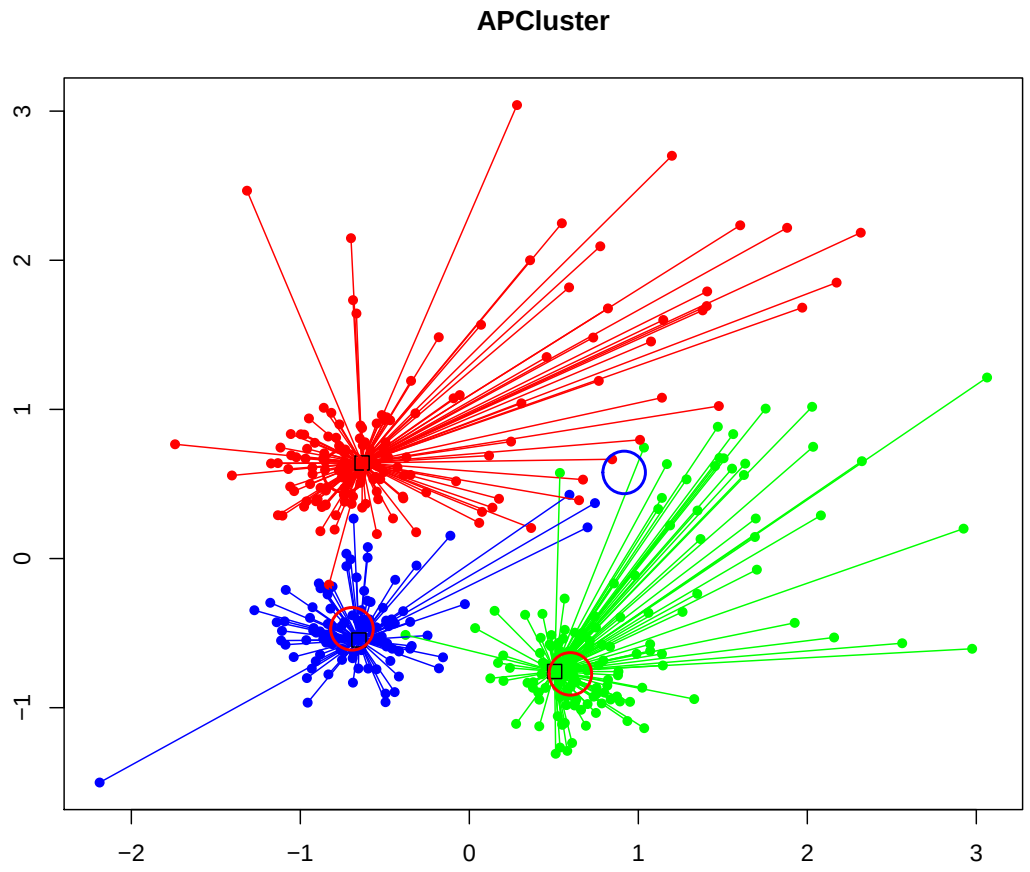


Figure 7.46: Affinity propagation applied to the d6A data set. Same result as shown in Fig. 7.44: the large cluster is not detected.

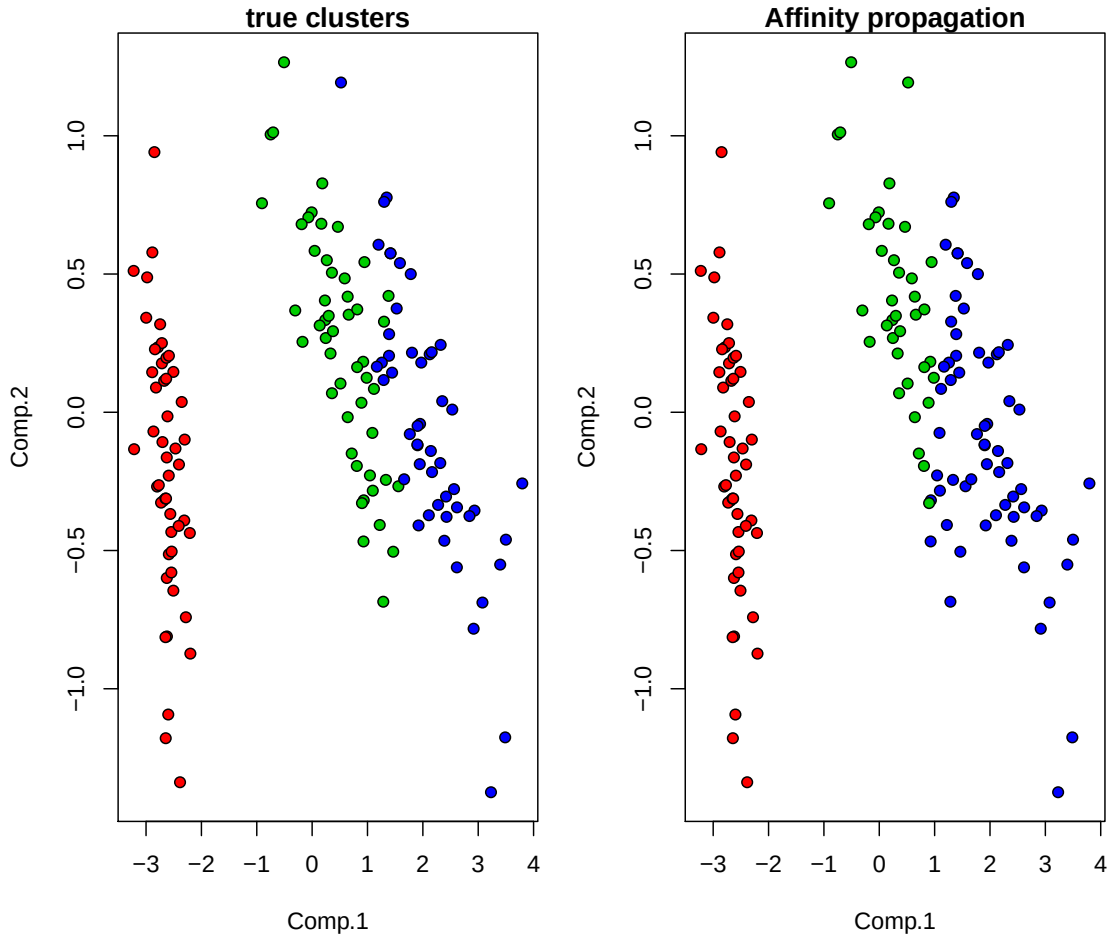


Figure 7.47: Affinity propagation applied to the iris data set. The left panels shows the true clusters and the right panel the clusters identified by affinity propagation. Affinity propagation gives quite good results.

Next we apply affinity propagation to the iris data set. Fig. 7.47 shows the results of affinity propagation. Affinity propagation gives quite good results.

Finally we applied affinity propagation to the multiple tissues data set. Fig. 7.48 shows the result of affinity propagation with $p = 0$. The result is quite good. Fig. 7.49 shows the result of affinity propagation with $p = 1$. Affinity propagation found too many clusters, where some cluster have only few members and one cluster is large. Fig. 7.50 shows the result of affinity propagation with $p = -1$. Only three clusters were found which cannot represent the 4 tissues. We saw that the quality of the result of affinity propagation depends on the choice of the hyperparameter p. Here the quality is sensitive to the choice of the hyperparameter.

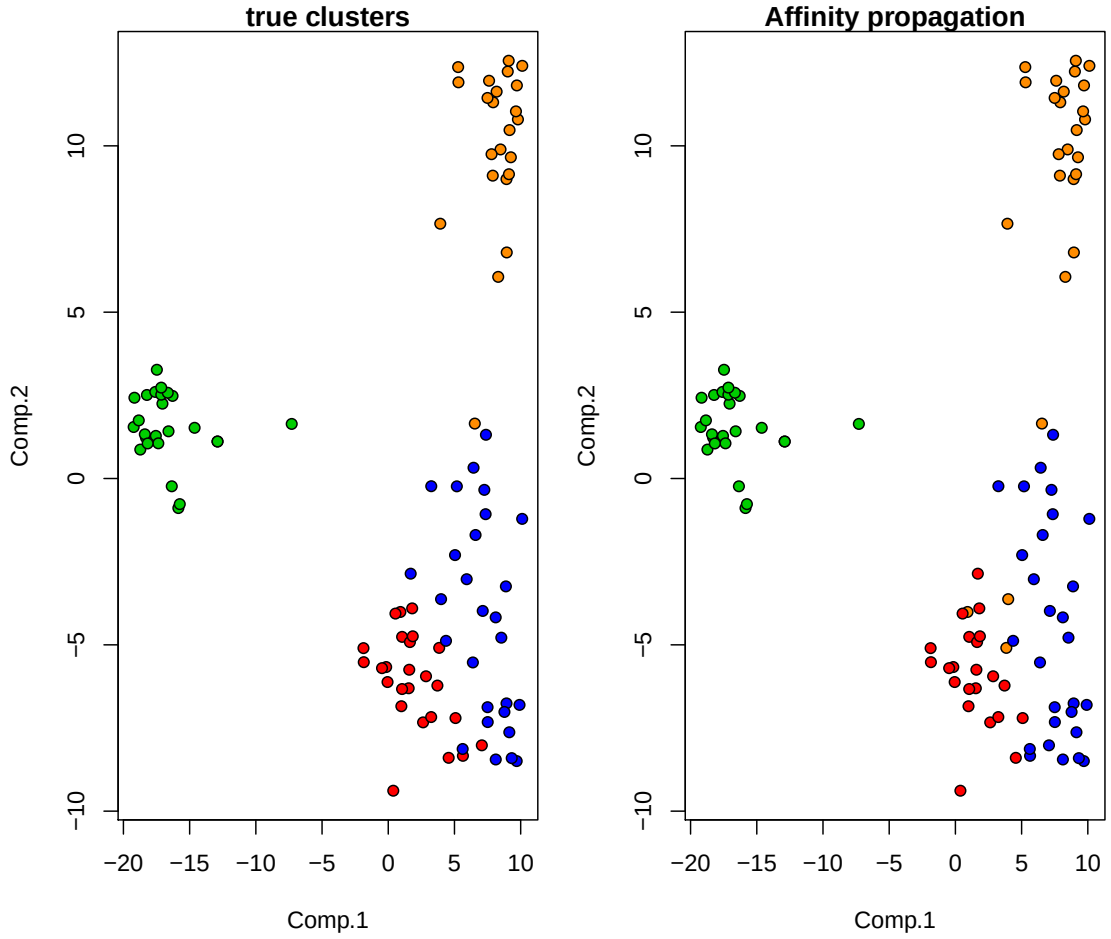


Figure 7.48: Affinity propagation applied to the multiple tissues data set with $p = 0$. The left panels shows the true clusters and the right panel the clusters identified by affinity propagation. The result is quite good.

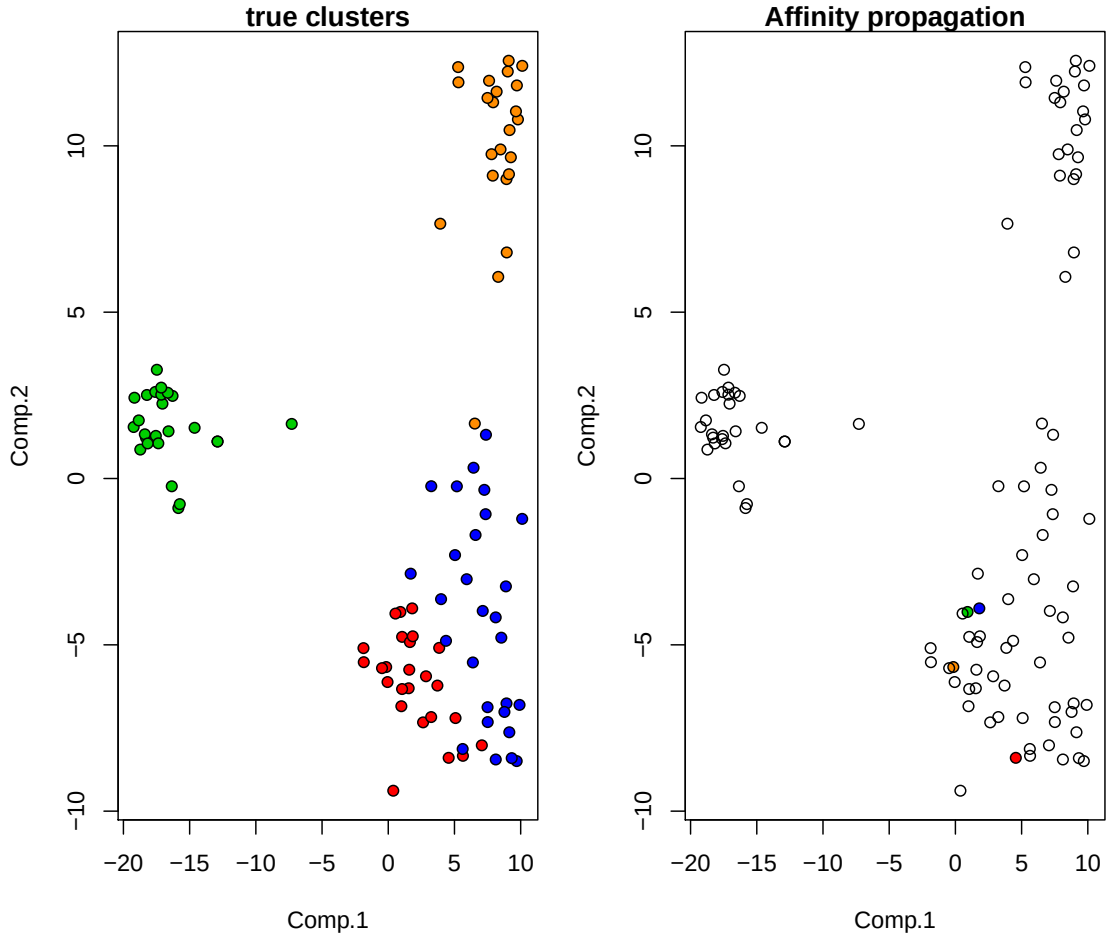


Figure 7.49: Affinity propagation applied to the multiple tissues data set with $p = 1$. The left panels shows the true clusters and the right panel the clusters identified by affinity propagation. Too many clusters, where some cluster have only few members and one cluster is large.

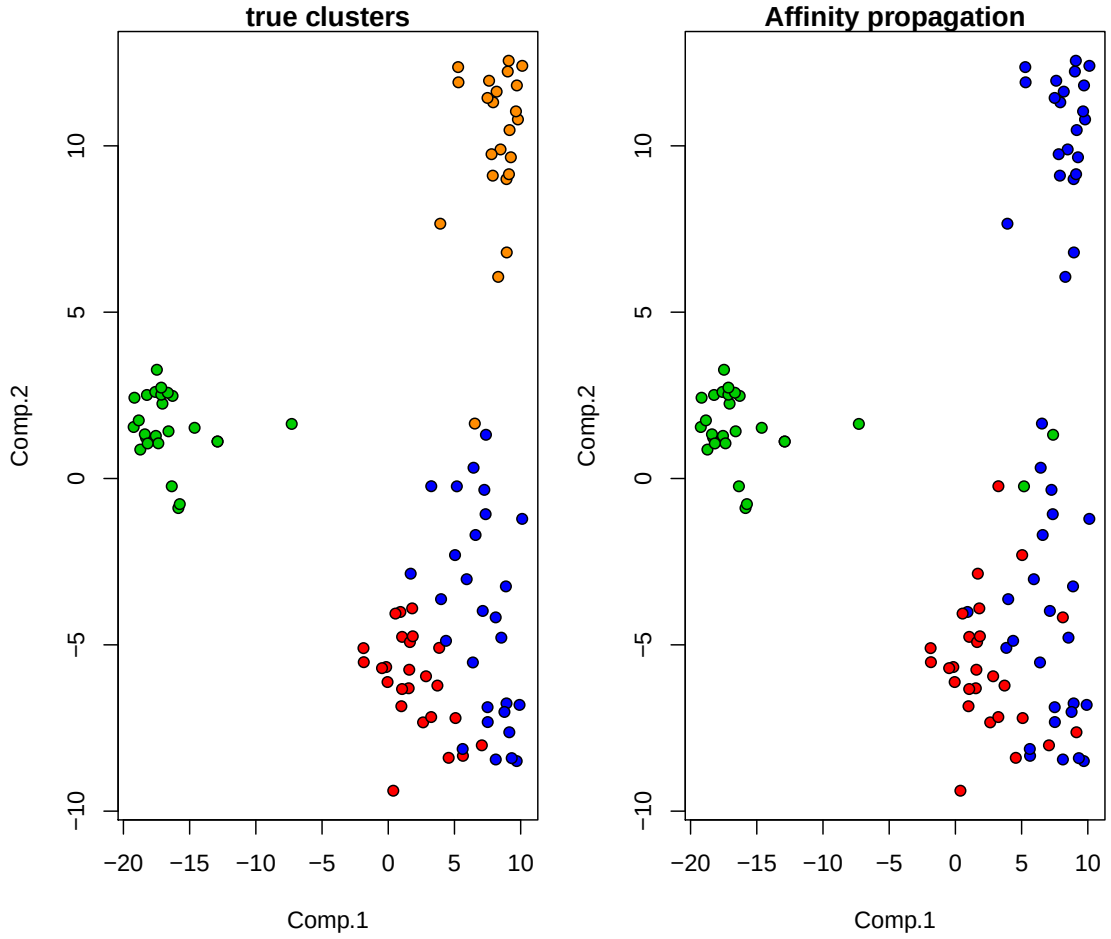


Figure 7.50: Affinity propagation applied to the multiple tissues data set with $p = -1$. The left panels shows the true clusters and the right panel the clusters identified by affinity propagation. Only three clusters were found which cannot represent the 4 tissues.

7.4.3 Similarity-based Mixture Models

Similarity-based mixture models are mixture models that use only similarities between objects but not feature vectors. The similarity between object i and object j is considered as the conditional probability $p(\boldsymbol{x}_j \mid \boldsymbol{x}_i)$. This idea was already introduced in stochastic neighbor embedding (SNE), where this was the probability that $\boldsymbol{x}_i$ would pick $\boldsymbol{x}_j$ as its neighbor. Another interpretation is: the more $\boldsymbol{x}_j$ is similar to $\boldsymbol{x}_i$, the less modifications are necessary to obtain $\boldsymbol{x}_j$ from $\boldsymbol{x}_i$, the higher is the probability to obtain $\boldsymbol{x}_j$ (randomly) from $\boldsymbol{x}_i$. In contrast to SNE, we now assume that the $p(\boldsymbol{x}_j \mid \boldsymbol{x}_i)$ are given to allow for similarity-based clustering.

7.4.3.1 Similarity-based Mixture of Gaussians

For similarity-based clustering, we assume that for each $\boldsymbol{x}$ the similarities $k(\boldsymbol{x}, \boldsymbol{x}_i)$ to each element $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ are given. In particular the similarities $k(\boldsymbol{x}_j, \boldsymbol{x}_i)$ for $1 \leq i, j \leq n$ are given to derive a clustering. Therefore, our goal is to express all update rules and parameters by these similarities.

We start with a Gaussian mixture model for density estimation and n observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. For K components, the estimated density at $\boldsymbol{x}$ is

$$p(\boldsymbol{x}) \;=\; \sum_{i=1}^{K} p(i)\; p(\boldsymbol{x} \mid i) \;=\; \sum_{i=1}^{K} p(i)\; k(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\;, \tag{7.92}$$

where

$$\sum_{i=1}^{K} p(i) \;=\; 1\;. \tag{7.93}$$

and $k(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is the Gaussian density with mean $\boldsymbol{\mu}_i$ and variance $\boldsymbol{\Sigma}_i$ evaluated at $\boldsymbol{x}$:

$$k(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \;=\; \frac{1}{(2\,\pi)^{m/2}\;|\boldsymbol{\Sigma}_i|^{1/2}} \exp\big(-\;\frac{1}{2}\;(\boldsymbol{x} - \boldsymbol{\mu}_i)^T\;\boldsymbol{\Sigma}_i^{-1}\;(\boldsymbol{x} - \boldsymbol{\mu}_i)\big)\;. \tag{7.94}$$

For $\boldsymbol{\Sigma}_i = \sigma_i^2 \boldsymbol{I}$ we obtain in an m-dimensional space ($\boldsymbol{x} \in \mathsf{R}^{\,m}$):

$$\begin{aligned} k(\boldsymbol{x}; \boldsymbol{\mu}_i, \sigma_i^2) \;&=\; \frac{1}{(2\,\pi)^{m/2}\;\sigma_i^m} \exp\big(-\;\frac{1}{2\,\sigma_i^2}\;\|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2\big) \\ &=\; \frac{1}{(2\,\pi)^{m/2}\;\sigma_i^m} \exp\big(-\;\frac{1}{2}\;\|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2\big)^{1/\sigma_i^2} \\ &=\; \frac{1}{\sigma_i^m} k(\boldsymbol{x}; \boldsymbol{\mu}_i, 1)^{1/\sigma_i^2}\;. \end{aligned} \tag{7.95}$$

Later we will replace the Gaussians by our similarities $k(\boldsymbol{x}_j, \boldsymbol{x}_i)$ for which the variance is unknown. Therefore, we replace $k(\boldsymbol{x}; \boldsymbol{\mu}_i, \sigma_i^2)$ by $\frac{1}{\sigma_i^m} k(\boldsymbol{x}; \boldsymbol{\mu}_i)^{1/\sigma_i^2}$ and obtain:

$$p(\boldsymbol{x}) \;=\; \sum_{j=1}^{K} p(j)\;\frac{1}{\sigma_j^m}\;k(\boldsymbol{x}, \boldsymbol{\mu}_j)^{1/\sigma_j^2}\;. \tag{7.96}$$

giving

$$p(\boldsymbol{x}_i) \;=\; \sum_{j=1}^{K} p(j)\; \frac{1}{\sigma_j^m}\; k(\boldsymbol{x}_i, \boldsymbol{\mu}_j)^{1/\sigma_j^2}\;. \tag{7.97}$$

This is the likelihood for observation $\boldsymbol{x}_i$.

We set

$$k(\boldsymbol{x}; \boldsymbol{\mu}_i, \sigma_i^2) \;=\; \frac{1}{\sigma_i^m}\; k(\boldsymbol{x}, \boldsymbol{\mu}_i)^{1/\sigma_i^2}\;. \tag{7.98}$$

The posterior gives the probability that $\boldsymbol{x}$ belongs to cluster i:

$$p(i \mid \boldsymbol{x}) \;=\; \frac{p(i)\; p(\boldsymbol{x} \mid i)}{\sum_{j=1}^{K} p(j)\; p(\boldsymbol{x} \mid j)} \;=\; \frac{p(i)\; k(\boldsymbol{x}; \boldsymbol{\mu}_i, \sigma_i^2)}{\sum_{j=1}^{K} p(j)\; k(\boldsymbol{x}; \boldsymbol{\mu}_j, \sigma_j^2)} \;=\; \frac{p(i)\; k(\boldsymbol{x}; \boldsymbol{\mu}_i, \sigma_i^2)}{p(\boldsymbol{x})}\;. \tag{7.99}$$

The posterior is invariant under scaling of the $p(i)$, that is the $p(i)$ need not sum to 1.

Note that with $w_i = p(i)$ and $w_{ik} = p(i \mid \boldsymbol{x}_k, \boldsymbol{w})$, we obtain

$$\begin{aligned} w_i \;&=\; p(i) \;=\; p(i \mid \boldsymbol{w}) \;=\; \int p(i, \boldsymbol{x} \mid \boldsymbol{w})\; d\boldsymbol{x} \\ &=\; \int p(i \mid \boldsymbol{x}, \boldsymbol{w})\; p(\boldsymbol{x} \mid \boldsymbol{w})\; d\boldsymbol{x} \;=\; \mathrm{E}_{p(\boldsymbol{x}|\boldsymbol{w})}(p(i \mid \boldsymbol{x}, \boldsymbol{w})) \\ &\approx\; \frac{1}{n} \sum_{k=1}^{n} p(i \mid \boldsymbol{x}_k, \boldsymbol{w}) \;=\; \frac{1}{n} \sum_{k=1}^{n} w_{ik}\;. \end{aligned} \tag{7.100}$$

Analog to Gaussian mixture models, we use a Dirichlet prior on $\boldsymbol{w}$ and Wishart prior on $\boldsymbol{\Sigma}_j$ (see Appendix 7.1.2).

The likelihood for one data point $\boldsymbol{x}$ is

$$p(\boldsymbol{x}) \;=\; \sum_{i=1}^{K} w_i\; \frac{1}{\sigma_i^m}\; k(\boldsymbol{x}_k; \boldsymbol{\mu}_i)^{1/\sigma_i^2} \;=\; \sum_{i=1}^{K} w_i\; k(\boldsymbol{x}_k; \boldsymbol{\mu}_i, \sigma_i^2)\;. \tag{7.101}$$

The objective is the negative log posterior, which is

$$\begin{aligned} B \;=\; &-\frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{K} \hat{w}_{ik}\; \log\left(k(\boldsymbol{x}_k; \boldsymbol{\mu}_i, \sigma_i^2)\right) \;-\; \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{K} \hat{w}_{ik}\; \log\left(w_i\right) \\ &-\; \frac{1}{n}\; \log p(\boldsymbol{w}) \;-\; \frac{1}{n}\; \log p(\boldsymbol{\sigma}^2) \;+\; \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{K} \hat{w}_{ik}\; \log \hat{w}_{ik} \\ &+\; \frac{1}{n} \sum_{k=1}^{n} \log c(\boldsymbol{x}_k)\;. \end{aligned} \tag{7.102}$$

In this objective $p(\boldsymbol{w})$ is the Dirichlet prior and the prior $p(\boldsymbol{\sigma}^2)$ is a Wishart distribution but for covariance matrices which are a multiple of the identity. $\boldsymbol{\sigma}^2$ is the vector $\boldsymbol{\sigma}^2 = (\sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2)$.

See Appendix 7.1.2 for the resulting update rules from this objective. The update rule for w_i is:

$$w_i^{\text{new}} \;=\; \frac{n\,\hat{w}_i \;+\; \gamma_i \;-1}{\gamma_s}\,, \tag{7.103}$$

where we used

$$\hat{w}_i \;=\; \frac{1}{n}\,\sum_{k=1}^{n}\hat{w}_{ik} \tag{7.104}$$

and $\gamma_s = \sum_{i=1}^{n}\gamma_i$. The hyper-parameters γ_i stem from the Dirichlet distribution. We now set

$$\hat{w}_{ik} \;=\; \frac{w_i^{\text{old}}\;k(\boldsymbol{x}_k;\boldsymbol{\mu}_i,\sigma_i^2)}{p(\boldsymbol{x}_k)}\,, \tag{7.105}$$

which is the posterior of w_i but with the actual estimate w_i^{old}.

The update rule for σ_i^2 is:

$$(\sigma_i^2)^{\text{new}} \;=\; \frac{\sum_{k=1}^{n}\hat{w}_{ik}\;(-\;2\;\log k(\boldsymbol{x}_k,\boldsymbol{\mu}_i))\;+\;w\;S}{n\;\hat{w}_i\;+\;w}\,, \tag{7.106}$$

7.4.3.2 Representing the Centers

Without a prior on the centers, the EM algorithm for the mixture of Gaussians has as update for its cluster centers (Appendix 7.1.2):

$$\boldsymbol{\mu}_i \;=\; \frac{\sum_{k=1}^{n}\hat{w}_{ik}\;\boldsymbol{x}_k}{\sum_{k=1}^{n}\hat{w}_{ik}}\,. \tag{7.107}$$

In the objective Eq. (7.102), only the term $k(\boldsymbol{x}_k;\boldsymbol{\mu}_i,\sigma_i^2)$ contains the centers $\boldsymbol{\mu}_i$, therefore we have for the optimal $\boldsymbol{\mu}_i$:

$$\frac{\partial B}{\partial\boldsymbol{\mu}_i} \;=\; -\,\frac{1}{n}\,\sum_{k=1}^{n}\hat{w}_{ik}\;\frac{1}{\sigma_i^2}\frac{\partial\log k(\boldsymbol{x}_k,\boldsymbol{\mu}_i)}{\partial\boldsymbol{\mu}_i} \;=\; 0\,, \tag{7.108}$$

where we used

$$\log k(\boldsymbol{x}_k;\boldsymbol{x}_i,\boldsymbol{\sigma}_i^2) \;=\; -\;n\;\log(\sigma_i)\;-\;\sigma_i^{-2}\;(-\;\log k(\boldsymbol{x}_k,\boldsymbol{x}_i))\,. \tag{7.109}$$

If

$$\log k(\boldsymbol{x}_k,\boldsymbol{\mu}_i) \;=\; -\frac{1}{2}\,\|\boldsymbol{x}_k-\boldsymbol{\mu}_i\| \;=\; -\frac{1}{2}\,(\boldsymbol{x}_k-\boldsymbol{\mu}_i)^T(\boldsymbol{x}_k-\boldsymbol{\mu}_i) \tag{7.110}$$

then

$$\frac{\partial\log k(\boldsymbol{x}_k,\boldsymbol{\mu}_i)}{\partial\boldsymbol{\mu}_i} \;=\; -\;\boldsymbol{x}_k\;+\;\boldsymbol{\mu}_i\,. \tag{7.111}$$

If we insert this derivative in Eq. (7.108) and solve for $\boldsymbol{\mu}_i$, then we obtain Eq. (7.107).

In the following, we want to express update formula Eq. (7.107) by the given similarities $k(\boldsymbol{x}_j, \boldsymbol{x}_k)$. For the update rules we do not require $\boldsymbol{\mu}_i$ but only $\|\boldsymbol{x}_l - \boldsymbol{\mu}_i\|$. Therefore, we will express $-2 \log k(\boldsymbol{x}_l, \boldsymbol{\mu}_i) = \|\boldsymbol{x}_l - \boldsymbol{\mu}_i\|$ by $k(\boldsymbol{x}_j, \boldsymbol{x}_k)$, that is, by the given similarities.

We assumed for the similarities

$$k(\boldsymbol{x}_k, \boldsymbol{x}_j) \;=\; \frac{1}{(2\,\pi)^{m/2}} \exp\big(-\;\frac{1}{2}\,\|\boldsymbol{x}_k - \boldsymbol{x}_j\|^2\big) \tag{7.112}$$

from which follows that

$$\begin{aligned} -\,2\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_j) \;&=\; n\log(2\pi) \;+\; \|\boldsymbol{x}_k - \boldsymbol{x}_j\|^2 \\ &=\; n\log(2\pi) \;+\; \boldsymbol{x}_k^T\boldsymbol{x}_k \;-\; 2\,\boldsymbol{x}_k^T\boldsymbol{x}_j \;+\; \boldsymbol{x}_j^T\boldsymbol{x}_j\;, \end{aligned} \tag{7.113}$$

and, therefore, we have

$$\boldsymbol{x}_k^T\boldsymbol{x}_j \;=\; \frac{1}{2}\,n\log(2\pi) \;+\; \frac{1}{2}\boldsymbol{x}_k^T\boldsymbol{x}_k \;+\; \frac{1}{2}\boldsymbol{x}_j^T\boldsymbol{x}_j \;+\; \log k(\boldsymbol{x}_k, \boldsymbol{x}_j)\;. \tag{7.114}$$

As mentioned, we want to express $\|\boldsymbol{x}_l - \boldsymbol{\mu}_i\|$ by the given similarities:

$$\begin{aligned} \|\boldsymbol{x}_l - \boldsymbol{\mu}_i\|^2 \;&=\; (\boldsymbol{x}_l - \boldsymbol{\mu}_i)^T(\boldsymbol{x}_l - \boldsymbol{\mu}_i) \;=\; \boldsymbol{x}_l^T\boldsymbol{x}_l \;-\; 2\,\boldsymbol{x}_l^T\boldsymbol{\mu}_i \;+\; \boldsymbol{\mu}_i^T\boldsymbol{\mu}_i \\ &=\; \boldsymbol{x}_l^T\boldsymbol{x}_l \;-\; 2\,\frac{\sum_{k=1}^n \hat{w}_{ik}\;\boldsymbol{x}_k^T\boldsymbol{x}_l}{\sum_{k=1}^n \hat{w}_{ik}} \;+\; \frac{\sum_{k,j=1}^n \hat{w}_{ik}\;\hat{w}_{ij}\;\boldsymbol{x}_k^T\boldsymbol{x}_j}{\big(\sum_{k=1}^n \hat{w}_{ik}\big)^2} \\ &=\; \boldsymbol{x}_l^T\boldsymbol{x}_l \;-\; \frac{\sum_{k=1}^n \hat{w}_{ik}\;\big(\boldsymbol{x}_k^T\boldsymbol{x}_k \;+\; 2\log k(\boldsymbol{x}_k, \boldsymbol{x}_l)\big)}{\sum_{k=1}^n \hat{w}_{ik}} \;-\; n\log\,(2\pi) \;-\; \boldsymbol{x}_l^T\boldsymbol{x}_l \\ &+\; \frac{\sum_{k,j=1}^n \hat{w}_{ik}\;\hat{w}_{ij}\;\Big(1/2\;n\log(2\pi) \;+\; 1/2\;\boldsymbol{x}_k^T\boldsymbol{x}_k \;+\; 1/2\;\boldsymbol{x}_j^T\boldsymbol{x}_j \;+\; \log k(\boldsymbol{x}_k, \boldsymbol{x}_j)\Big)}{\big(\sum_{k=1}^n \hat{w}_{ik}\big)^2} \\ &=\; -\,\frac{\sum_{k=1}^n \hat{w}_{ik}\;\boldsymbol{x}_k^T\boldsymbol{x}_k}{\sum_{k=1}^n \hat{w}_{ik}} \;-\; 2\,\frac{\sum_{k=1}^n \hat{w}_{ik}\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_l)}{\sum_{k=1}^n \hat{w}_{ik}} \;-\; n\log(2\pi) \\ &+\; 1/2\;n\log(2\pi) \;+\; \frac{1}{2}\,\frac{\sum_{k=1}^n \hat{w}_{ik}\;\boldsymbol{x}_k^T\boldsymbol{x}_k}{\sum_{k=1}^n \hat{w}_{ik}} \;+\; \frac{1}{2}\,\frac{\sum_{j=1}^n \hat{w}_{ij}\;\boldsymbol{x}_j^T\boldsymbol{x}_j}{\sum_{k=1}^n \hat{w}_{ik}} \\ &+\; \frac{\sum_{k,j=1}^n \hat{w}_{ik}\;\hat{w}_{ij}\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_j)}{\big(\sum_{k=1}^n \hat{w}_{ik}\big)^2} \\ &=\; -\;1/2\;n\log(2\pi) \;-\; 2\,\frac{\sum_{k=1}^n \hat{w}_{ik}\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_l)}{\sum_{k=1}^n \hat{w}_{ik}} \;+\; \frac{\sum_{k,j=1}^n \hat{w}_{ik}\;\hat{w}_{ij}\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_j)}{\big(\sum_{k=1}^n \hat{w}_{ik}\big)^2}\;. \end{aligned} \tag{7.115}$$

To obtain the second line, we inserted the expression for $\boldsymbol{\mu}_i$ in Eq. (7.107). For obtaining the third line, we inserted the dot product of Eq. (7.114). Using this expression for $\|\boldsymbol{x}_l - \boldsymbol{\mu}_i\|^2$, we can compute

$$k(\boldsymbol{x}_k, \boldsymbol{\mu}_i) \;=\; \frac{1}{(2\,\pi)^{m/2}} \exp\big(-\;\frac{1}{2}\,\|\boldsymbol{x}_k - \boldsymbol{\mu}_i\|^2\big)\;. \tag{7.116}$$

If we ignore the scaling factor $1/(2\pi)^{m/2}$ and define

$$k(\boldsymbol{x}, \boldsymbol{x}_i) \;=\; \exp\big(-\;\frac{1}{2}\,\|\boldsymbol{x} - \boldsymbol{x}_i\|^2\big) \tag{7.117}$$

then we would obtain

$$\|\boldsymbol{x}_l - \boldsymbol{\mu}_i\|^2 \;=\; -\,2\,\frac{\sum_{k=1}^{n} \hat{w}_{ik}\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_l)}{\sum_{k=1}^{n} \hat{w}_{ik}} \;+\; \frac{\sum_{k,j=1}^{n} \hat{w}_{ik}\;\hat{w}_{ij}\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_j)}{\big(\sum_{k=1}^{n} \hat{w}_{ik}\big)^2}\,. \tag{7.118}$$

and

$$k(\boldsymbol{x}_k, \boldsymbol{\mu}_i) \;=\; \exp\big(-\,\frac{1}{2}\,\|\boldsymbol{x}_k - \boldsymbol{\mu}_i\|^2\big)\,. \tag{7.119}$$

Scaling of k does not change the updates, because in the update rule for $\hat{w}_{ik}$, the scaling of $p(\boldsymbol{x}_k)$ cancels with the scaling of k.

7.4.3.3 Update Rules

The algorithm has following the update rules:

$$k(\boldsymbol{x}_k; \boldsymbol{\mu}_i, \sigma_i^2) \;=\; \frac{1}{\sigma_i^m}\,k(\boldsymbol{x}_k, \boldsymbol{\mu}_i)^{1/\sigma_i^2}\,. \tag{7.120}$$

$$p(\boldsymbol{x}_k) \;=\; \sum_{i=1}^{l} w_i^{\text{old}}\;k(\boldsymbol{x}_k; \boldsymbol{\mu}_i, \sigma_i^2)\,, \tag{7.121}$$

$$\hat{w}_{ik} \;=\; \frac{w_i^{\text{old}}\;k(\boldsymbol{x}_k; \boldsymbol{\mu}_i, \sigma_i^2)}{p(\boldsymbol{x}_k)}\,, \tag{7.122}$$

$$\hat{w}_i \;=\; \frac{1}{n}\sum_{k=1}^{n} \hat{w}_{ik}\,, \tag{7.123}$$

$$w_i^{\text{new}} \;=\; \frac{n\;\hat{w}_i \;+\; \gamma_i \;-1}{\gamma_s}\,, \tag{7.124}$$

$$(\sigma_i^2)^{\text{new}} \;=\; \frac{\sum_{k=1}^{n} \hat{w}_{ik}\;(-\,2\;\log k(\boldsymbol{x}_k, \boldsymbol{\mu}_i)) \;+\; w\;S}{n\;\hat{w}_i \;+\; w}\,, \tag{7.125}$$

$$\begin{aligned} -\,2\;\log k(\boldsymbol{x}_l, \boldsymbol{\mu}_i) \;&=\; \|\boldsymbol{x}_l - \boldsymbol{\mu}_i\|^2 \qquad (7.126)\\ &=\; \frac{\sum_{k=1}^{n} \hat{w}_{ik}\;(-\,2\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_l))}{n\;\hat{w}_i} \;-\; \frac{1}{2}\,\frac{\sum_{k,j=1}^{n} \hat{w}_{ik}\;\hat{w}_{ij}\;(-\,2\;\log k(\boldsymbol{x}_k, \boldsymbol{x}_j))}{n^2\;\hat{w}_i^2}\,, \end{aligned}$$

$$k(\boldsymbol{x}_l, \boldsymbol{\mu}_i)^{\text{new}} \;=\; \exp\big(-\,\frac{1}{2}\,\|\boldsymbol{x}_l - \boldsymbol{\mu}_i\|^2\big)\,. \tag{7.127}$$

The covariance matrix can also be represented by the similarities. However, we would require a matrix inversion where the dimension of this matrix is the number of observations. This is computationally very expensive and the algorithm is no longer stable.

7.4.3.4 Examples

We demonstrate similarity-based mixture clustering on the same data sets as affinity propagation. We start with the data set depicted in Fig. 7.37, which contains 6 clusters in a two-dimensional space. Three clusters are smaller than the other three clusters, that is, they have a smaller variance.

We cluster the x7 data set by similarity-based mixture clustering. The result of similarity-based mixture clustering is shown in Fig. 7.51. Circles indicate the true cluster centers and black rectangles the centers found by similarity-based mixture clustering. One cluster could not be separated into the two true clusters while affinity propagation did it. How the parameters $\boldsymbol{w}$ change during the iterative update of similarity-based mixture clustering is shown in Fig. 7.52. Fig. 7.53 shows the development of the variances of the components given by σ during the iterations of similarity based mixture clustering. For the analog data of Fig. 7.39, the result of similarity-based mixture clustering is shown in Fig. 7.54. The result is the same as with affinity propagation.

For the x9 data set of Fig. 7.41, the result of affinity propagation is depicted in Fig. 7.55. The cluster which affinity propagation was not able to separate is separated by similarity-based mixture clustering.

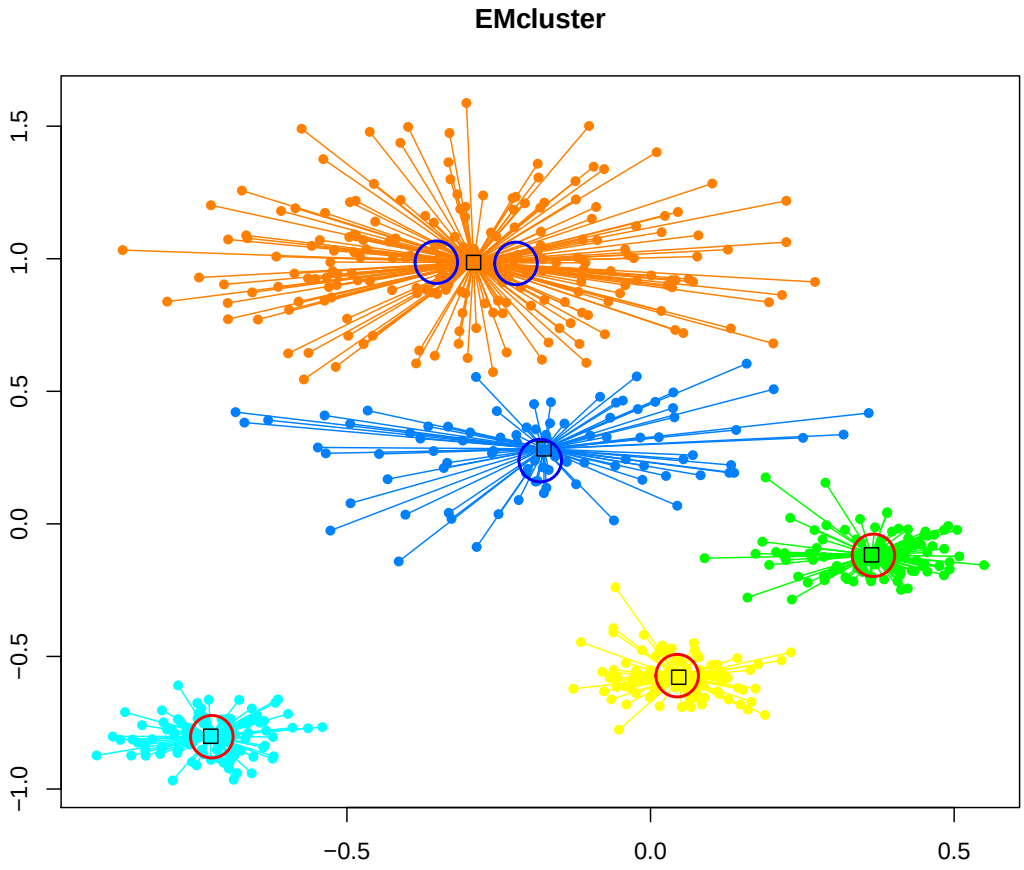


Figure 7.51: Similarity-based mixture clustering applied to the x7 data set. Circles indicate the true cluster centers and black rectangles the centers found by similarity-based mixture clustering. One cluster could not be separated into the two true clusters while affinity propagation did that.

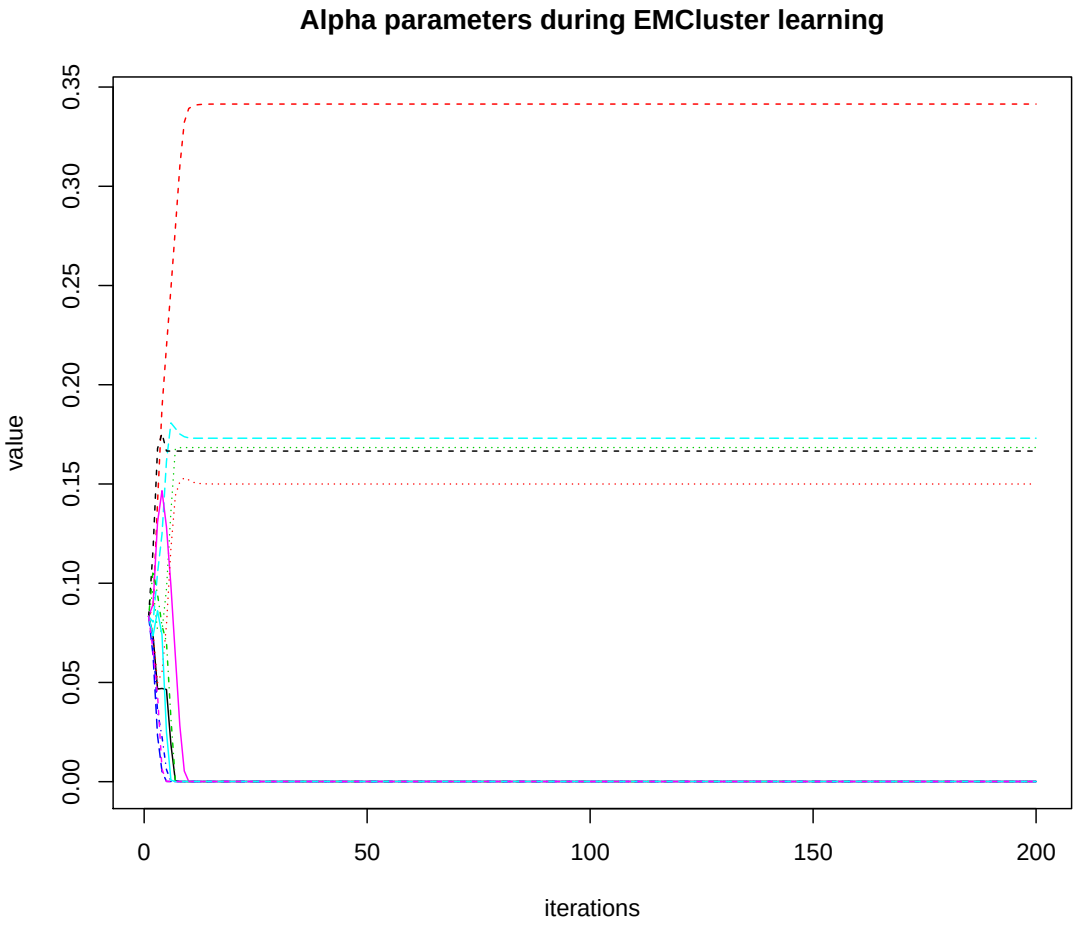


Figure 7.52: The changes of alpha during the iterations of similarity-based mixture clustering for the x7 data set.

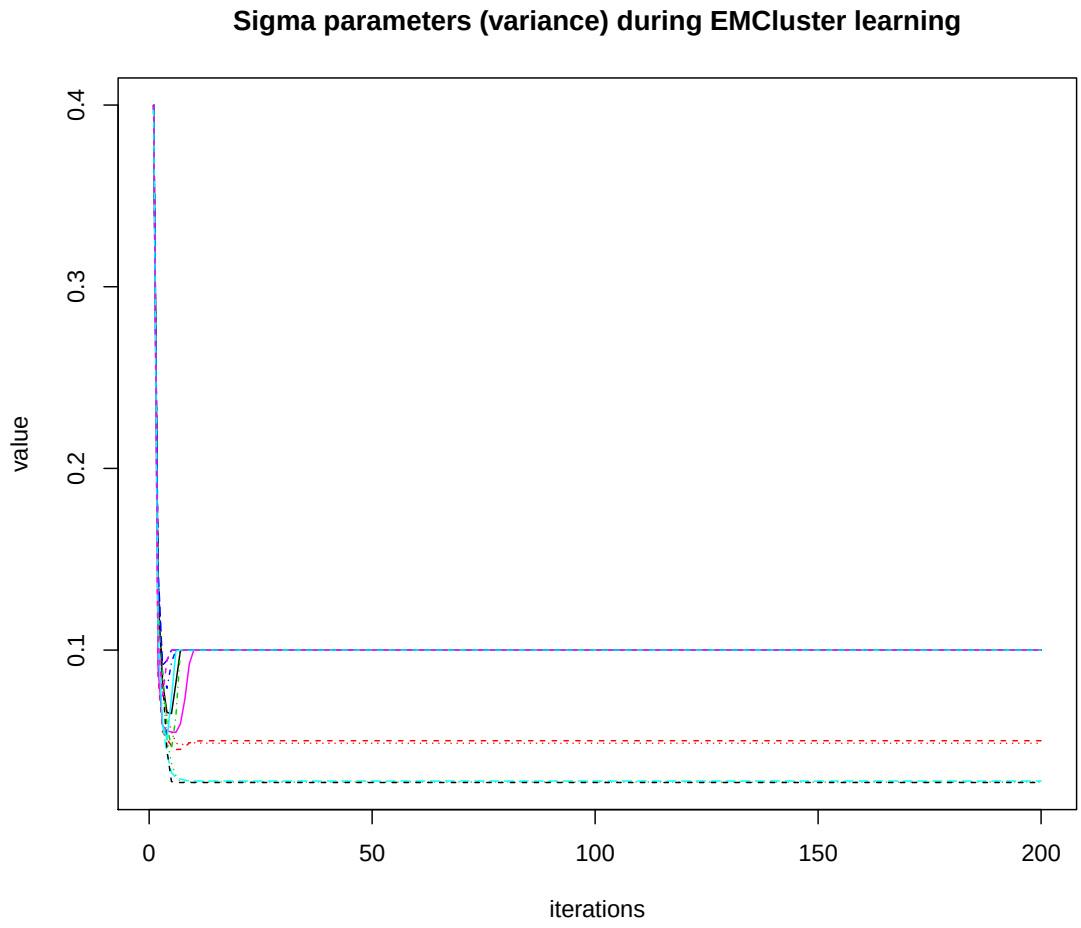


Figure 7.53: The changes of sigma (the variance of the components) during the iterations of similarity-based mixture clustering for the x7 data set.

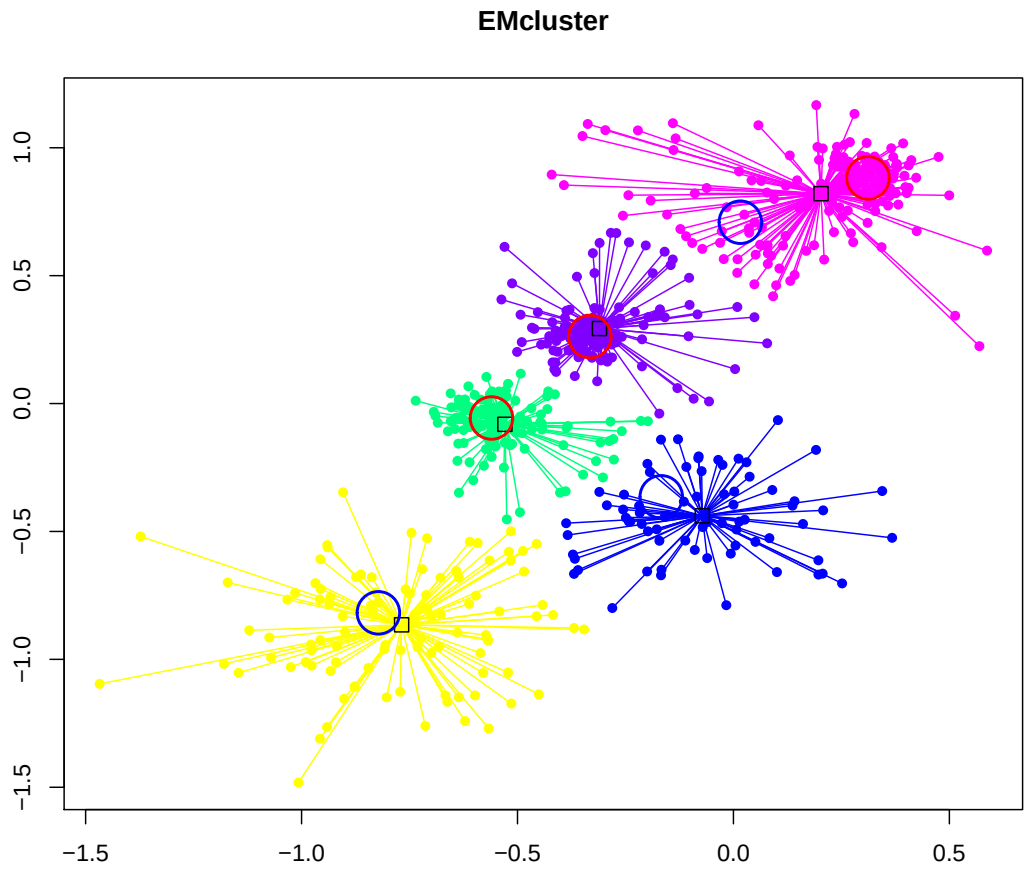


Figure 7.54: Similarity-based mixture clustering applied to the x7A data set. The result is the same as with affinity propagation.

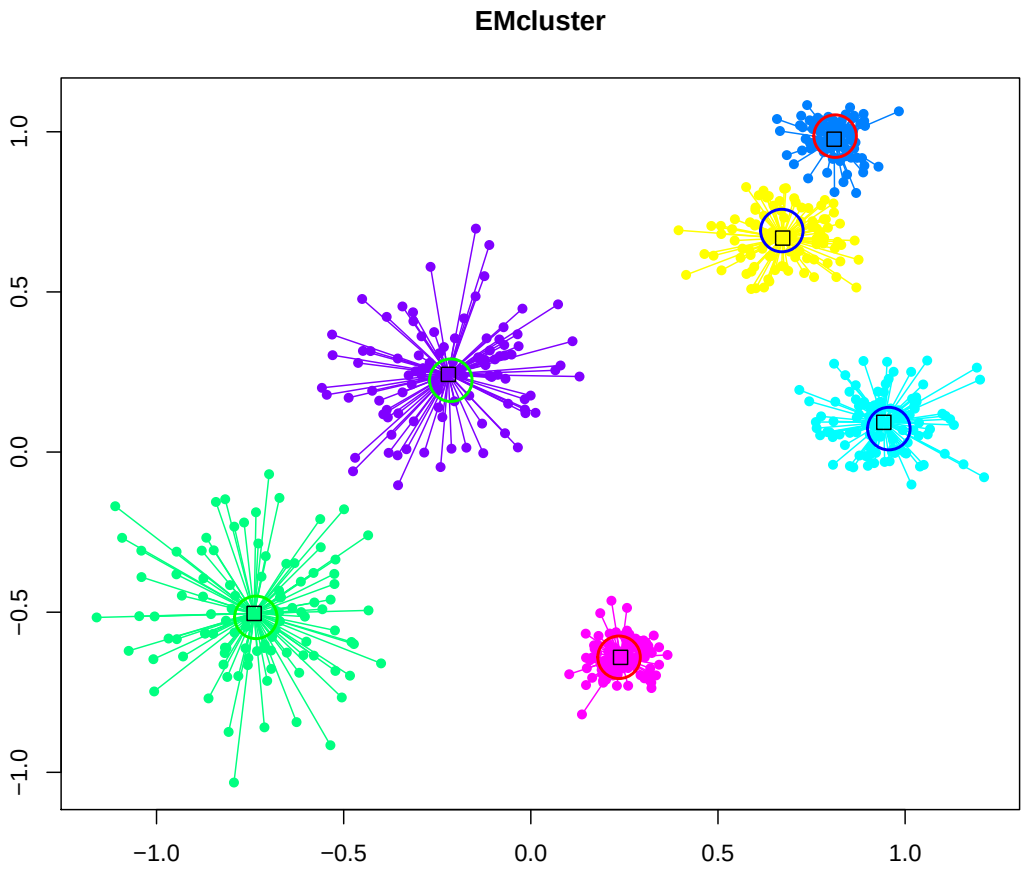


Figure 7.55: Similarity-based mixture clustering applied to the x9 data set. The cluster which affinity propagation was not able separate is separated by similarity-based mixture clustering.

Next we test similarity-based mixture clustering on the 6-dimensional data set of Fig. 7.43. It contains different cluster sizes with respect to the variation of the elements in the clusters. The data has one large cluster and three smaller clusters. We generated two data sets which are depicted in Fig. 7.43 and Fig. 7.45 where the cluster centers are indicated by circles. We perfrom similarity-based mixture clustering. The result is given in Fig. 7.56 where again circles indicate the true cluster centers and black rectangles the centers found by similarity-based mixture clustering. In contrast to affinity propagation, similarity-based mixture clustering is able to detect the large cluster because it adjusts the variance of the clusters. It even correctly assigns data points that are far away from the cluster center. The same result for another data set (see Fig. 7.45) can be seen in Fig. 7.57.

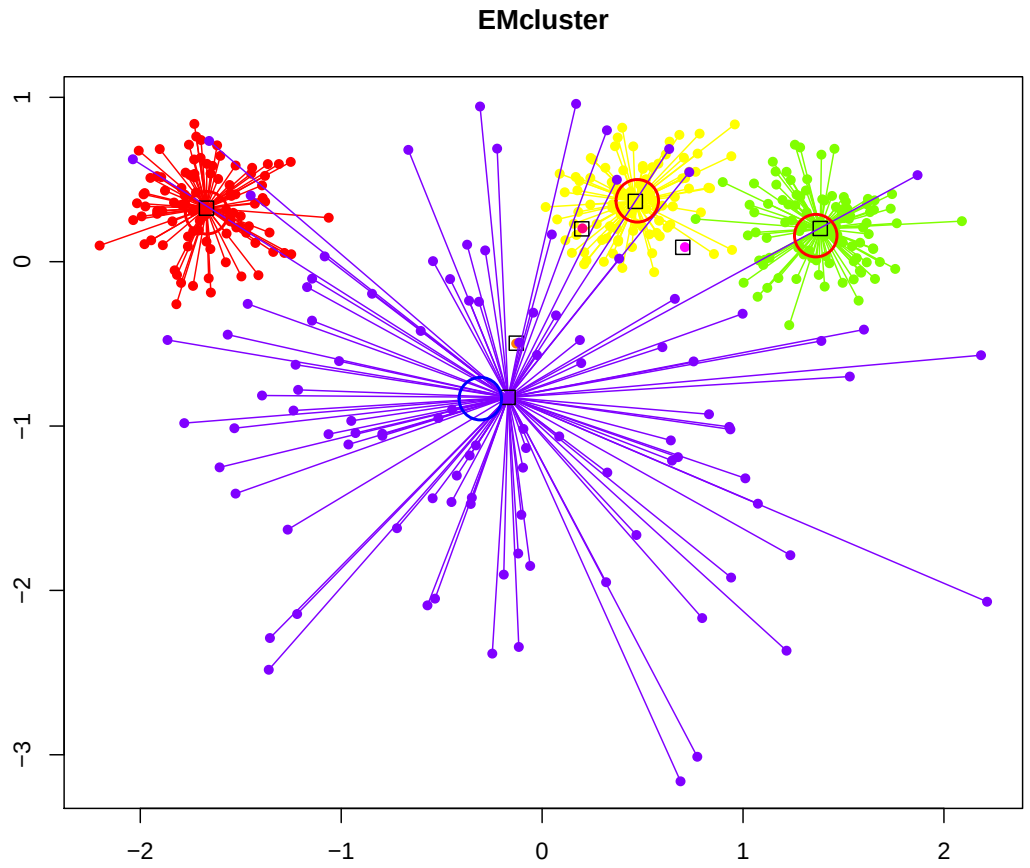


Figure 7.56: Similarity-based mixture clustering applied to the d6 data set. Circles indicate the true cluster centers and black rectangles the centers found by similarity-based mixture clustering. In contrast to affinity propagation, mixture clustering detects the large cluster because it adjusts the variance of the clusters. It even correctly assigns data points that are far away from the cluster center.

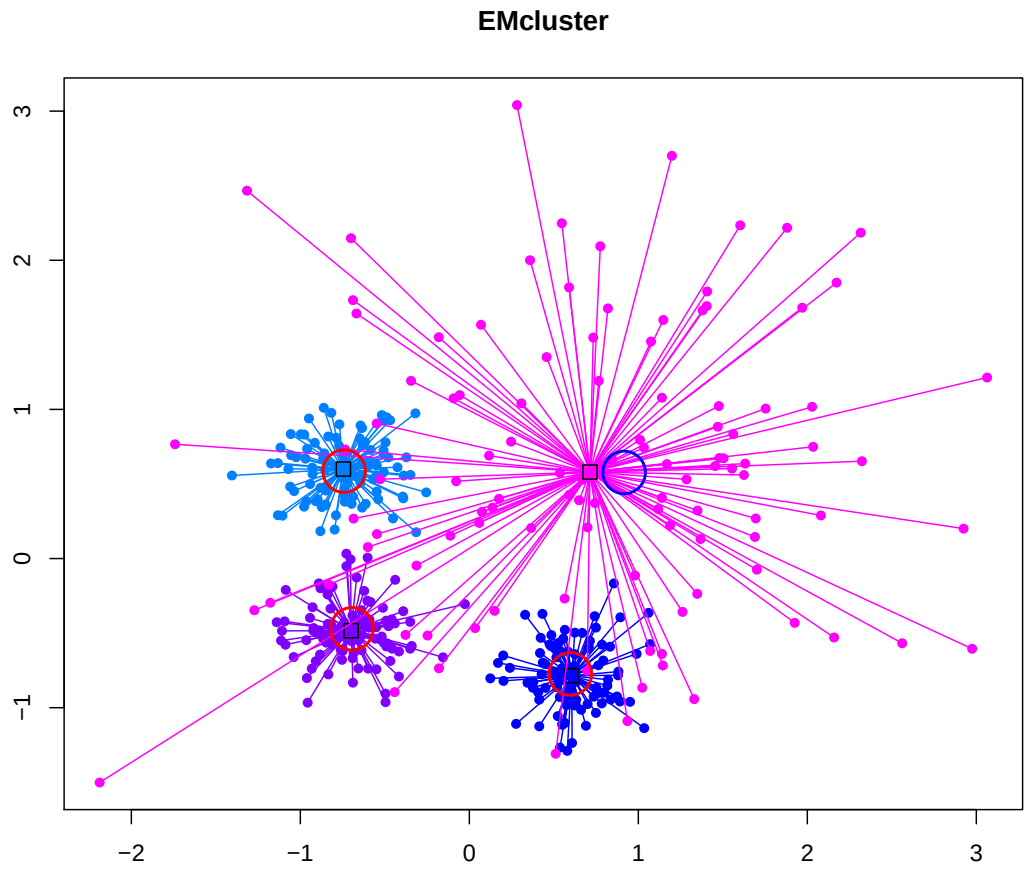


Figure 7.57: Similarity-based mixture clustering applied to the d6A data set. Same result as shown in Fig. 7.56: in contrast to affinity propagation, the large cluster is detected.

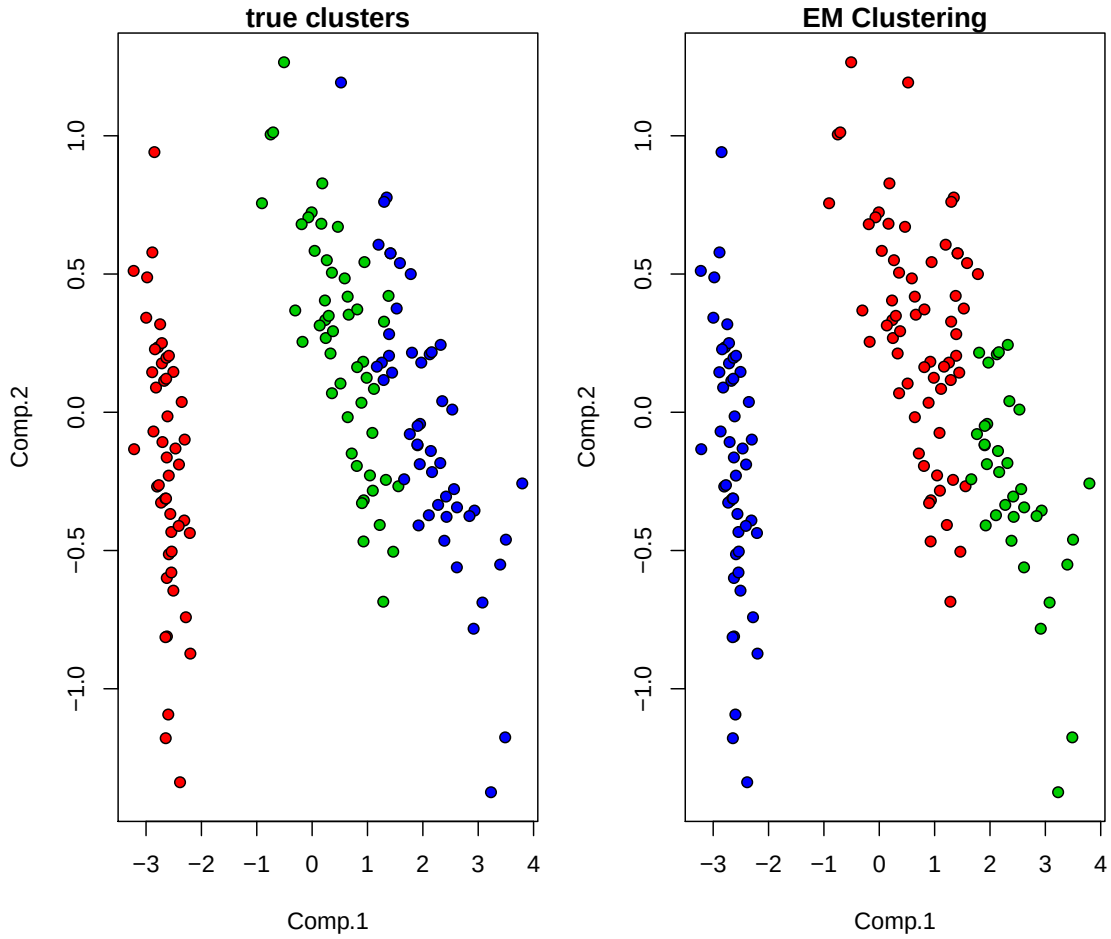


Figure 7.58: Similarity-based mixture clustering applied to the iris data set. The left panels shows the true clusters and the right panel the clusters identified by similarity-based mixture clustering.

Next we apply similarity-based mixture clustering to the iris data set. Fig. 7.58 shows the results of similarity-based mixture clustering. Similarity-based mixture clustering gives quite good results.

Finally we applied similarity-based mixture clustering to the multiple tissues data set. Fig. 7.59 shows the result of similarity-based mixture clustering. The result is quite good.

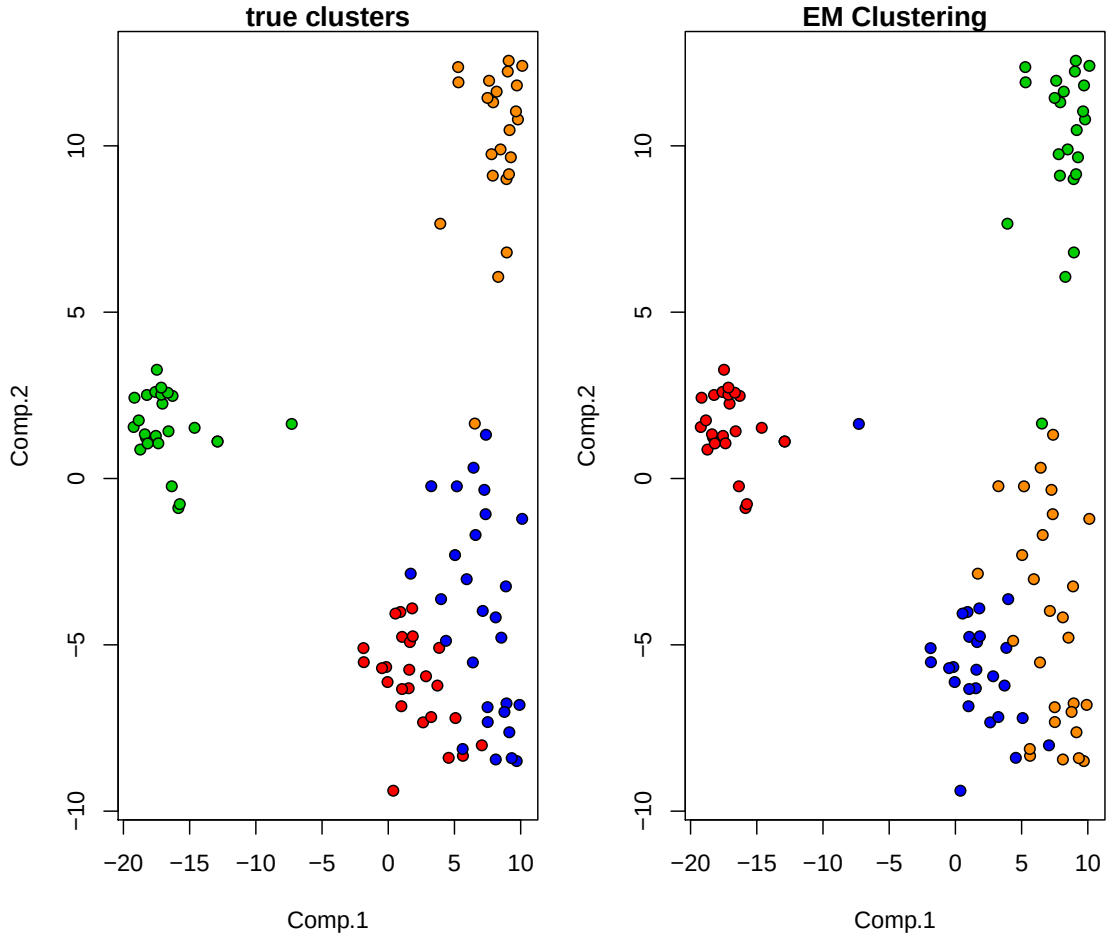


Figure 7.59: Similarity-based mixture clustering applied to the multiple tissue data set. The left panels shows the true clusters and the right panel the clusters identified by similarity-based mixture clustering.

Chapter 8

Biclustering

Biclustering simultaneously clusters the rows and the columns of a matrix. A *bicluster* of a matrix is a pair of a set of row elements that belong to the bicluster and a set of column elements that belong to the bicluster. For a bicluster its column elements are similar to each other on its row elements or/and vice versa. Each row element can belong to multiple biclusters or to no bicluster at all. Analogously, each column element can belong to multiple biclusters or to no bicluster at all. In contrast to standard clustering, the clustering of row elements (samples) is performed on only a subgroup of column elements (features). The role of row elements / samples and column elements / features can be interchanged in biclustering.

The noise free data from Fig. 6.9 shows biclusters which are once more shown in Fig. 8.1. The data matrix contains blocks of patterns which are biclusters. For visualization purposes only, the blocks are constructed by adjacent row and column elements.

8.1 Types of Biclusters

Biclusters can be be divided into biclusters with:

(a) constant values,

(b) constant rows values,

(c) constant column values,

(d) additive coherent values,

(e) multiplicative coherent values,

(f) general coherent values.

a) Bicluster with constant values:

2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0

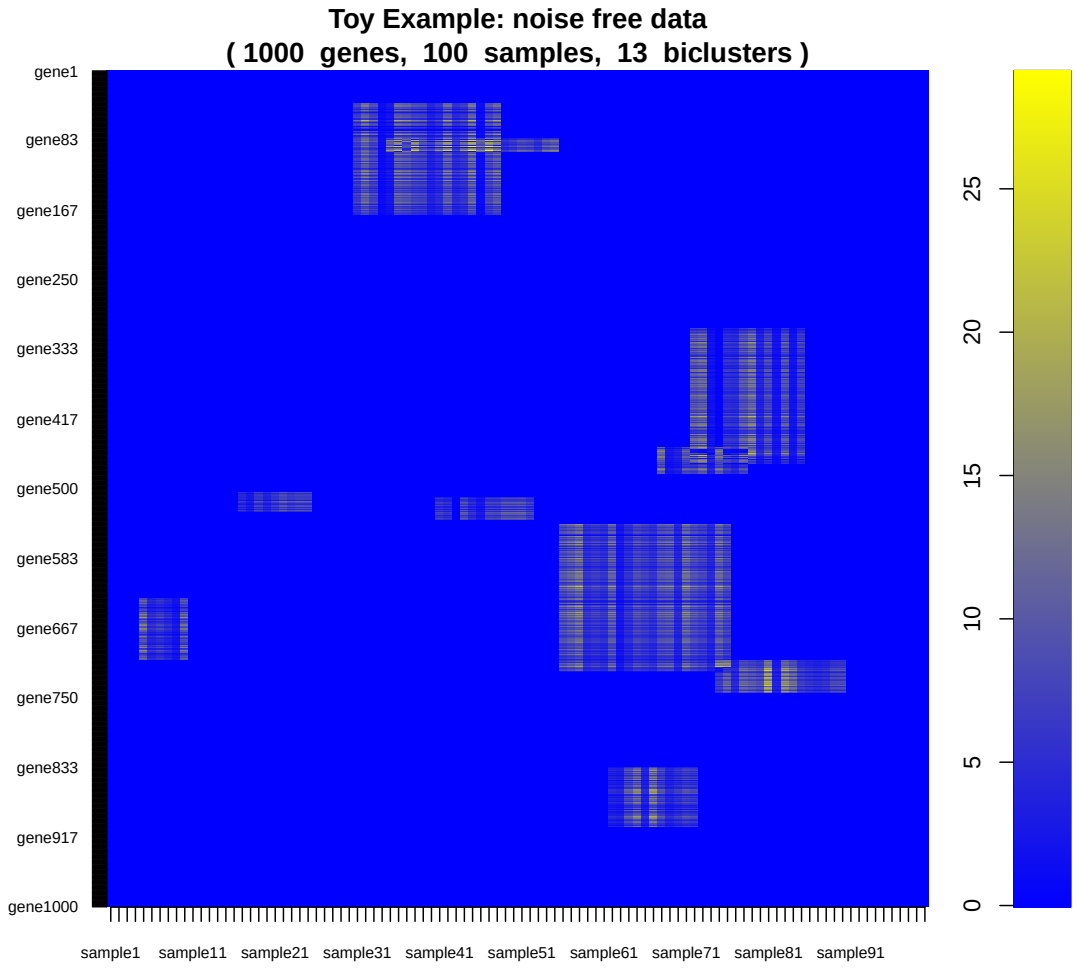


Figure 8.1: The data contains blocks of patterns which are biclusters. For visualization purposes only, the blocks are constructed by adjacent row and column elements.

b) Bicluster with constant values on rows and column pattern:

1.0	1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0	3.0
4.0	4.0	4.0	4.0	4.0
4.0	4.0	4.0	4.0	4.0

c) Bicluster with constant values on columns and row pattern:

1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0

d) Bicluster with coherent values (additive):

1.0	4.0	5.0	0.0	1.5
4.0	7.0	8.0	3.0	4.5
3.0	6.0	7.0	2.0	3.5
5.0	8.0	9.0	4.0	5.5
2.0	5.0	6.0	1.0	2.5

e) Bicluster with coherent values (multiplicative):

1.0	0.5	2.0	0.2	0.8
2.0	1.0	4.0	0.4	1.6
3.0	1.5	6.0	0.6	2.4
4.0	2.0	8.0	0.8	3.2
5.0	2.5	10.0	1.0	4.0

8.2 Overview of Biclustering Methods

A survey of biclustering approaches has been given by Madeira and Oliveira [2004]. In principle, there exist four categories of biclustering methods:

(1) variance minimization methods,

(2) two-way clustering methods,

(3) motif and pattern recognition methods, and

(4) probabilistic and generative approaches.

(1) Variance minimization methods define clusters as blocks in the matrix with minimal deviation of their elements. This definition has been already considered by Hartigan [1972] and extended by Tibshirani et al. [1999]. The δ-cluster methods search for blocks of elements having a deviation ("variance") from the mean below δ. One example are δ-ks clusters Califano et al. [2000], where the maximum and the minimum of each row need to differ less than δ on the selected columns. A second example are δ-pClusters Wang et al. [2002] which are defined as 2×2 sub-matrices with pairwise edge differences less than δ. A third example are the Cheng and Church [2000] δ-biclusters having a mean squared error below δ after fitting an additive model with a constant, a row, and a column effect. FLexible Overlapped biClustering (FLOC) extends Cheng-Church δ-biclusters by dealing with missing values via an occupancy threshold θ and by using both l_1 and l_2 norms Yang et al. [2005].

(2) Two-way clustering methods apply conventional clustering to the columns and rows and (iteratively) combine the results. Coupled Two-Way Clustering (CTWC) iteratively performs standard clustering of the rows (columns) using previously constructed columns (rows) clusters as features Getz et al. [2000]. Also Interrelated Two-Way Clustering (ITWC) using k-means Tang et al. [2001] and Double Conjugated Clustering (DCC) using self-organizing maps combine column and row clustering Busygin et al. [2002].

(3) Motif and pattern recognition methods define a bicluster as samples sharing a common pattern or motif. To simplify this task, some methods discretize the data in a first step, like xMOTIF Murali and Kasif [2003] or Bimax Prelic et al. [2006], the latter even binarizes the data and searches for blocks with an enrichment of ones. Order-Preserving Sub-Matrices (OPSM) searches for blocks having the same order of values in their columns Ben-Dor et al. [2003]. Using partial models, only the column order on subsets must be preserved. Spectral clustering (SPEC) performs a singular value decomposition of the data matrix after normalization Kluger et al. [2003]. SPEC extracts columns with the same pattern using the fact that they are linearly dependent and span a subspace associated with a certain singular value. The Iterative Signature Algorithm (ISA) selects samples that have a given pattern and then uses these samples to define a new sample signature Ihmels et al. [2004]. This sample signature, in turn, is used to select features and to define a new feature signature. For each bicluster to be extracted, this process is initialized by a randomly selected binary feature signature and repeated iteratively. A related approach uses a Hough transform for identifying groups of linearly dependent features and samples Gan et al. [2008]. CCC-biclustering Madeira and Oliveira [2009], Madeira et al. [2010] is a method for time series which finds patterns in contiguous columns.

(4) Probabilistic and generative methods use model-based techniques to define biclusters. Statistical-Algorithmic Method for Bicluster Analysis (SAMBA) uses a bi-partitioned graph, where both conditions and genes are nodes Tanay et al. [2002]. An edge from a gene to a condition means that the gene responds to the condition. With a probabilistic objective, subgraphs are found that have a significantly higher connectivity than the overall graph. In another approach, Sheng et al. [2003] use Gibbs sampling to estimate the parameters of a simple frequency model for the expression pattern of a bicluster. However, the data must first be discretized and then only one bicluster with constant column values at each step can be extracted. Probabilistic Relational Models (PRMs) Getoor et al. [2002] and their extension ProBic Van den Bulcke [2009] are fully generative

models which combine probabilistic modeling and relational logic. Another generative approach is cMonkey Reiss et al. [2006] which models biclusters by Markov chain processes. Both PRMs and cMonkey are able to integrate additional data sources.

In the plaid model family Lazzeroni and Owen [2002], the i-th bicluster is extracted by row and column indicator variables ρ_{ki} and κ_{ij}. The values of each bicluster are explained by a general additive model $\theta_{kij} = \mu_i + \alpha_{ki} + \beta_{ij}$. Parameters are estimated by a least square fit. Gu and Liu [2008] generalized the plaid models to fully generative models called Bayesian BiClustering model (BBC). To avoid the high percentage of overlap in the plaid models, BBC constrains the overlapping of biclusters to only one dimension. Further it allows different error variances per bicluster. Caldas and Kaski [2008] also extended the plaid model to a fully generative model using a Bayesian framework and found that the plaid model is equivalent to the PRM model for specific parameters.

Factor Analysis for Bicluster Acquisition (FABIA) is based on a multiplicative model Hochreiter et al. [2010]. FABIA accounts for linear dependencies between row elements and column elements, and also captures heavy-tailed distributions as observed in real-world data (transcriptomics and genetics). The generative framework allows to utilize well-founded model selection methods and to apply Bayesian techniques.

The latter models Gu and Liu [2008], Caldas and Kaski [2008], Hochreiter et al. [2010] are generative models which have the advantage that (1) they select models using well-understood model selection techniques like maximum likelihood, (2) hyperparameter selection methods (e.g. to determine the number of biclusters) can rely on the Bayesian framework, (3) signal-to-noise ratios can be computed, (4) they can be compared to each other via the likelihood or posterior, (5) tests like the likelihood ratio test are possible, and (6) they produce a global model to explain all data.

8.3 FABIA Biclustering

We assume that the data is given as a data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, where every row corresponds to a sample and every column to a feature; the value x_{kj} corresponds to the value of the j-th feature in the k-th sample. The matrix $\boldsymbol{X}$ is the input to biclustering methods like FABIA.

We defined a bicluster as a pair of a row set and a column set for which the rows are similar to each other on the columns and vice versa. In a multiplicative model, two vectors are similar if one is a multiple of the other, that is, the angle between them is zero or their correlation coefficient is (minus) one. It is clear that such a linear dependency on subsets of rows and columns can be represented as an outer product $\boldsymbol{u}\ \boldsymbol{y}^T$ of two vectors $\boldsymbol{u}$ and $\boldsymbol{y}$. The vector $\boldsymbol{u}$ corresponds to a *prototype column vector* that contains zeros for features not participating in the bicluster, whereas $\boldsymbol{y}$ is a vector of *factors* with which the prototype column vector is scaled for each sample; clearly $\boldsymbol{y}$ contains zeros for samples not participating in the bicluster. Vectors containing many zeros or values close to zero are called *sparse vectors*. Fig. 8.2 visualizes this representation of biclusters by sparse vectors schematically.

The overall model for l biclusters and additive noise is

$$\boldsymbol{X} \ = \ \sum_{j=1}^{l} \boldsymbol{u}_j \ \boldsymbol{y}_j^T \ + \ \boldsymbol{\Upsilon} \ = \ \boldsymbol{Y} \ \boldsymbol{U}^T \ + \ \boldsymbol{\Upsilon} \ , \tag{8.1}$$

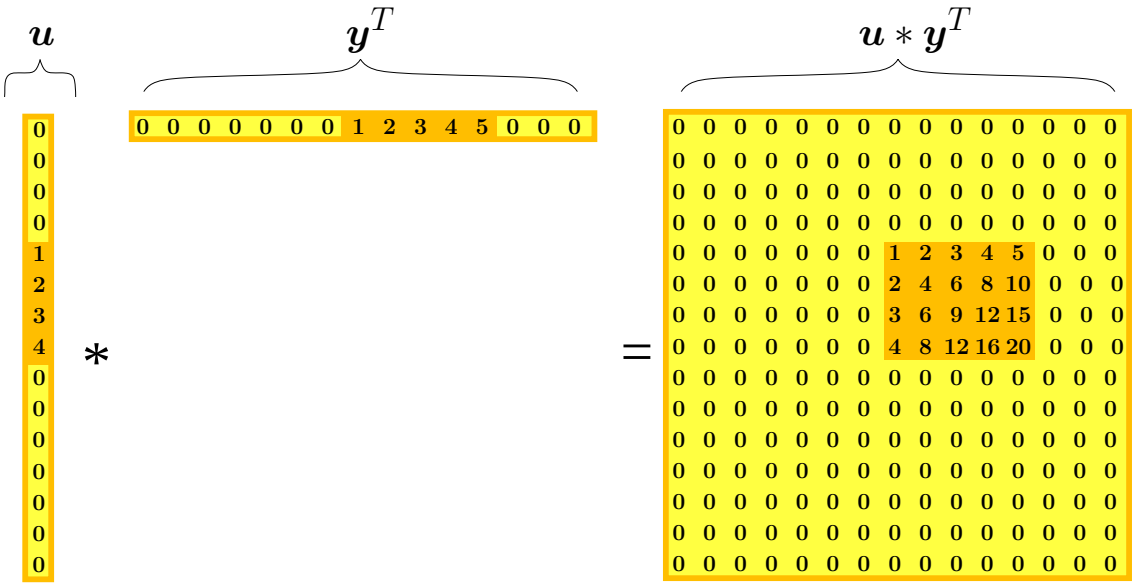


Figure 8.2: The outer product $\boldsymbol{u}\,\boldsymbol{y}^T$ of two sparse vectors defines a bicluster in a matrix. Note that the non-zero entries in the vectors are adjacent to each other for visualization purposes only.

where $\boldsymbol{\Upsilon} \in \mathbb{R}^{n \times m}$ is additive noise; $\boldsymbol{u}_i \in \mathbb{R}^m$ and $\boldsymbol{y}_j \in \mathbb{R}^n$ are the sparse prototype vector and the sparse vector of factors of the j-th bicluster, respectively. $\boldsymbol{U} \in \mathbb{R}^{m \times l}$ is the sparse prototype matrix containing the prototype vectors $\boldsymbol{u}_j$ as columns and $\boldsymbol{Y} \in \mathbb{R}^{n \times l}$ is the sparse factor matrix containing the transposed factors $\boldsymbol{y}_j^T$ as rows. Eq. (8.1) formulates biclustering as sparse matrix factorization.

According to Eq. (8.1), the i-th sample $\boldsymbol{x}_i$, i.e., the i-th row of $\boldsymbol{X}$, is

$$\boldsymbol{x}_i \;=\; \sum_{j=1}^{l} \boldsymbol{u}_j \, y_{ij} \;+\; \boldsymbol{\epsilon}_i \;=\; \boldsymbol{U}\,\tilde{\boldsymbol{y}}_i \;+\; \boldsymbol{\epsilon}_i \,, \tag{8.2}$$

where $\boldsymbol{\epsilon}_i$ is the i-th column of the noise matrix $\boldsymbol{\Upsilon}$ and $\tilde{\boldsymbol{y}}_i = (y_{i1}, \ldots, y_{il})^T$ denotes the i-th row of the matrix $\boldsymbol{Y}$. Recall that $\boldsymbol{y}_j^T = (y_{1j}, \ldots, y_{nj})$ is the vector of values that constitutes the j-th bicluster (one value per sample), while $\tilde{\boldsymbol{y}}_i$ is the vector of values that contribute to the i-th sample (one value per bicluster).

The formulation in Eq. (8.2) facilitates a generative interpretation by a factor analysis model with l factors Everitt [1984]:

$$\boldsymbol{x} \;=\; \sum_{j=1}^{l} \boldsymbol{u}_j \, \tilde{y}_j \;+\; \boldsymbol{\epsilon} \;=\; \boldsymbol{U}\,\tilde{\boldsymbol{y}} \;+\; \boldsymbol{\epsilon} \,, \tag{8.3}$$

where $\boldsymbol{x}$ are the observations, $\boldsymbol{U}$ is the loading matrix, $\tilde{y}_j$ is the value of the j-th factor, $\tilde{\boldsymbol{y}} = (\tilde{y}_1, \ldots, \tilde{y}_l)^T$ is the vector of factors, and $\boldsymbol{\epsilon} \in \mathbb{R}^m$ is the additive noise. Standard factor analysis assumes: the noise is independent of $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{y}}$ is $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$-distributed, and $\boldsymbol{\epsilon}$ is $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Psi})$-distributed. The covariance matrix $\boldsymbol{\Psi} \in \mathbb{R}^{l \times l}$ is diagonal — expressing independent Gaussian noise. The parameter $\boldsymbol{U}$ explains the dependent (common) and $\boldsymbol{\Psi}$ the independent variance in the observations $\boldsymbol{x}$ Hochreiter et al. [2006].

The covariance matrix for $\tilde{\boldsymbol{y}}$ is the unit matrix, which means that the biclusters are assumed to not be correlated. This assumption ensures that one true bicluster in the data will not be divided into dependent small model biclusters — thereby ensuring maximal model biclusters. Note, however, that this assumption still allows for overlapping biclusters.

Standard factor analysis does not consider sparse factors and sparse loadings which are essential in the FABIA formulation to represent biclusters. Sparseness is obtained by a component-wise independent *Laplace* distribution Hyvärinen and Oja [1999], which is now used as a prior on the factors $\tilde{\boldsymbol{y}}$ instead of the Gaussian:

$$p(\tilde{\boldsymbol{y}}) \;=\; \left(\tfrac{1}{\sqrt{2}}\right)^{l} \prod_{j=1}^{l} e^{-\sqrt{2}\,|\tilde{y}_j|} \tag{8.4}$$

Sparse loadings $\boldsymbol{u}_j$ and, therefore sparse $\boldsymbol{U}$, are achieved by a component-wise independent *Laplace* prior for the loadings (like for the factors):

$$p(\boldsymbol{u}_j) \;=\; \left(\tfrac{1}{\sqrt{2}}\right)^{m} \prod_{k=1}^{m} e^{-\sqrt{2}\,|u_{kj}|} \tag{8.5}$$

The FABIA model contains the product of Laplacian variables which is distributed proportionally to the 0-th order modified Bessel function of the second kind Bithas et al. [2007]. For large values, this Bessel function is a negative exponential function of the square root of the random variable. Therefore, the tails of the distribution are heavier than those of the Laplace distribution. The Gaussian noise, however, reduces the heaviness of the tails such that the heaviness is between Gaussian and Bessel function tails — about as heavy as the tails of the Laplacian distribution. These *heavy tails* are exactly the desired model characteristics to distinguish noise from signal (cf. projection pursuit).

To identify biclusters, the model parameters $\boldsymbol{U}$ and $\boldsymbol{\Psi}$ have to be selected. Maximum likelihood is the most common approach for selecting a generative model. Unfortunately, for FABIA, the likelihood is analytically intractable. The reason is that FABIA aims at generating sparse values, for which Laplacian priors are used (in contrast to the commonly used Gaussian priors). The resulting integral defining the likelihood cannot be computed analytically. In such situations, variational approaches can be applied, where a lower bound of the likelihood is maximized instead of the likelihood itself.

Expectation maximization (EM) is the most popular method for maximizing the likelihood Dempster et al. [1977]. The EM algorithm has been extended to variational expectation maximization Girolami [2001], Palmer et al. [2006]. For FABIA this approach is followed. However, also a prior on the loadings is assumed to be sparse in order to make the loadings sparse, too. Therefore, FABIA model selection is based on a variational expectation maximization for maximizing the posterior Hochreiter et al. [2006], Talloen et al. [2007].

Fig. 8.3 illustrates a FABIA result on a simulated data set, where the biclusters have been created as contiguous blocks for visualization purposes.

8.4 Examples

FABIA is implemented in the R package `fabia`. For the R package `fabia` demos can be started by `fabiaDemo()` Hochreiter et al. [2010].

We test FABIA on a 50-dimensional data set with linearly mixed super-Gaussians from Subsection 5.4. In contrast to factor analysis, `fabia` aims to make both the factors and the loadings sparse. Sparseness is related to ICA (see above), therefore `fabia` tends to results similar to ICA methods. Fig. 8.4 shows the first two components of the results of `fabia` (sparse factor analysis) applied to the mixture of 50-dimensional super-Gaussians.

The results of FABIA on a toy data from Section 6.3.2 have been presented there. Fig. 6.18 shows the reconstructed data and error. Fig. 6.19 shows the matrices into which the data matrix was factorized. FABIA performed much better than non-negative matrix factorization or sparse matrix factorization.

We apply FABIA biclustering to the iris data set. For this data set biclustering does not make much sense, because sparseness in the loadings is not justified as there are only four features. The FABIA result is shown in Fig. 8.5.

The loadings of FABIA are

```
             bicluster2 bicluster3 bicluster1
Sepal.Length  0.6490253 -0.2155847          0
Sepal.Width  -0.1828042 -0.3511878          0
Petal.Length  1.7465614  0.0000000          0
Petal.Width   0.7285857 -0.0453688          0
```

Only two biclusters have been found. We see that the first bicluster "bicluster2" focuses on "Petal.Length" which is correlated to "Petal.Width" and "Sepal.Length". The first bicluster is related to petal but includes also sepal length. The second bicluster "bicluster3" removed "Petal.Length" and "Petal.Width" due to sparseness and focused on "Sepal.Width" including also "Sepal.Length". The second bicluster is related to sepal.

The fabia package comes with a plot function which produces a biplot. A biplot plots both the features and the samples into one plot. The biplot for the FABIA result on the iris data set is shown in Fig. 8.6. Circles are features where red circles are the most relevant features and golden circles are less relevant features. Squares are samples which are colored according to the given prior knowledge. Two biclusters are plotted along the axis where both the samples and features are in one plot. Deviations from the center in vertical or horizontal directions means that the sample or feature is important for the according bicluster. If a sample or a feature is on the diagonal, then this means this sample or this feature belongs to both biclusters.

We test FABIA on the multiple tissue data set. We show the results for 200, 500, and 2,000 iterations (a separate run for each number of iterations).

Fig. 8.7 shows the result after 200 iterations. The first bicluster ("bicluster3") separates the prostate samples (green) and the colon samples (orange) from the rest. The second bicluster ("bicluster4") separates the colon samples (orange) from the rest. The fourth bicluster ("bicluster2") separates the breast samples (red) from the rest, though not perfectly.

Fig. 8.8 shows the result after 500 iterations. The first bicluster ("bicluster2") separates the prostate samples (green) from the rest. The second bicluster ("bicluster3") separates the colon samples (orange) from the rest. The fourth bicluster ("bicluster2") separates the lung samples (blue) from the rest, though not perfectly.

Fig. 8.9 shows the result after 2,000 iterations. The first bicluster ("bicluster4") clearly separates the prostate samples (green) from the rest. The second bicluster ("bicluster1") clearly separates the colon samples (orange) from the rest. The third and fourth bicluster ("bicluster3" and "bicluster2") can help to separate the lung samples (blue) and the breast samples (red) from one another but not very reliable.

With more iterations the solutions become more spares. More sparseness and the focus on the most strong signals leads to a more clear separation of the classes. Therefore, with more iterations the separation becomes more clear.

Since the FABIA solution is sparse, it allows for an interpretation. We investigate which genes drive the first bicluster. Therefore we extract the most relevant genes of the first bicluster.

The functions of these genes are described in the following:

1. **KLK3** is known as the prostate specific antigen. The GeneCards database `http://www.genecards.org` comments:

   > "Serum level of this protein, called PSA in the clinical setting, is useful in the diagnosis and monitoring of prostatic carcinoma."

2. **ACPP** The GeneCards database says:

   > ACPP "is synthesized under androgen regulation and is secreted by the epithelial cells of the prostate gland."

3. **KLK2** The GeneCards database says

   > KLK2 "is primarily expressed in prostatic tissue and is responsible for cleaving pro-prostate-specific antigen into its enzymatically active form."

This means that the most relevant genes of the first bicluster are all strongly associated with prostate tissue.

Fig. 8.10 shows a biplot of the first and second bicluster of FABIA applied to multiple tissue data with 4 components after 2000 iterations. The large red circles are the features (genes) driving the bicluster while the golden small circles are features with minor influence. Sparseness pushes most features to zero in one direction (the axes are golden due to the features pushed to zero). The turquoise prostate samples at the x-axis are separated from the other samples. Fig. 8.11 show the according biplot for the first and third bicluster. For the first bicluster at the x-axis the genes KLK3 (prostate specific antigen) and ACPP (secreted by the epithelial cells of the prostate gland) can be recognized. These genes separate the turquoise prostate samples at the x-axis from the other samples.

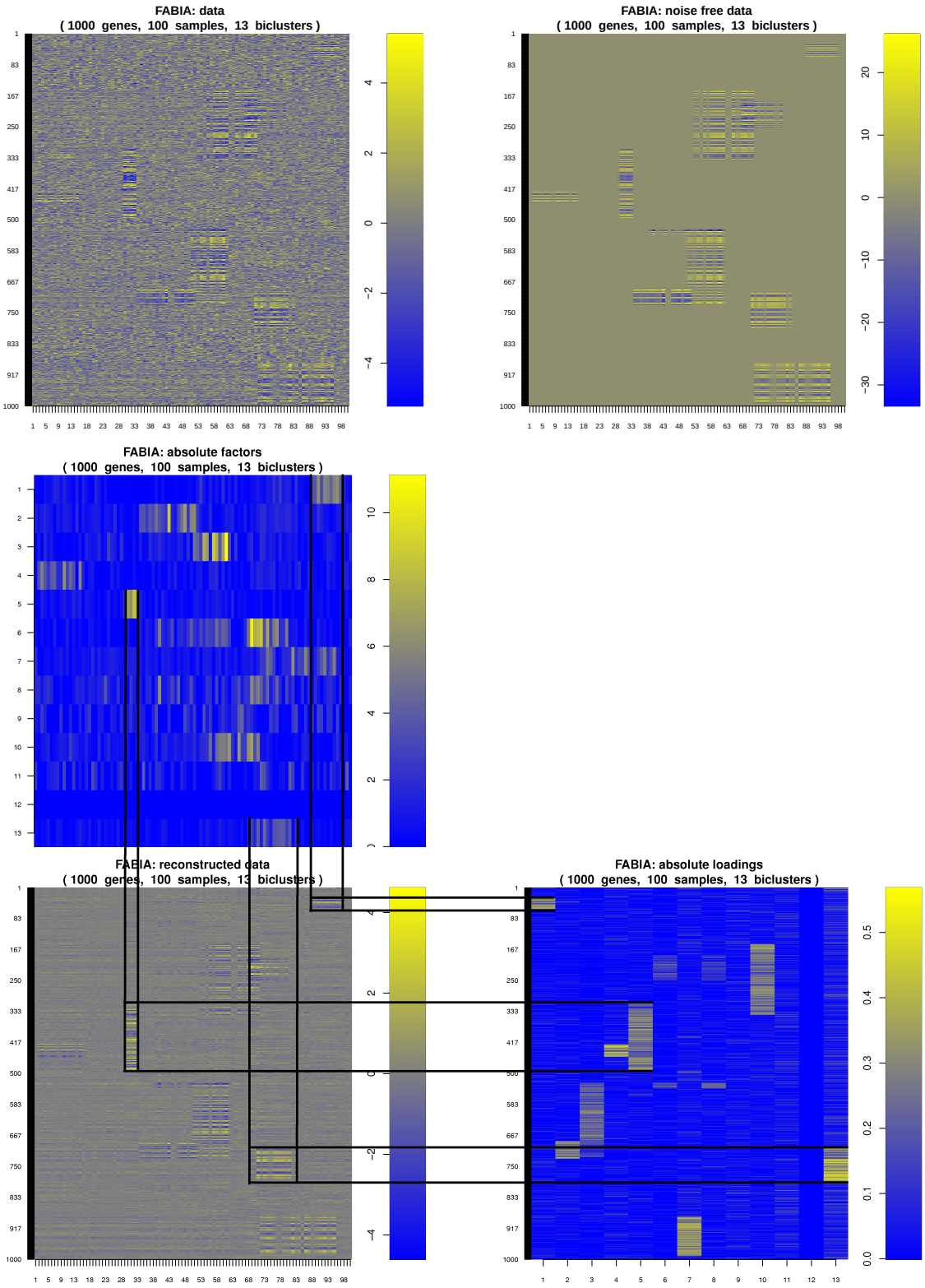


Figure 8.3: An example of FABIA model selection. The data have 10 true biclusters. We have trained the model with 13 biclusters. Only for visualization purposes, the biclusters are generated as contiguous blocks. Top: data (left) and noise-free data (right). Middle: factors $\boldsymbol{Y}$. Bottom: data reconstructed by the FABIA model as $\boldsymbol{Y}\,\boldsymbol{U}^T$ (left) and loadings $\boldsymbol{U}$ (right). The lines indicate three biclusters and connect each bicluster in the reconstructed data with its corresponding factors (middle) and loadings (bottom right).

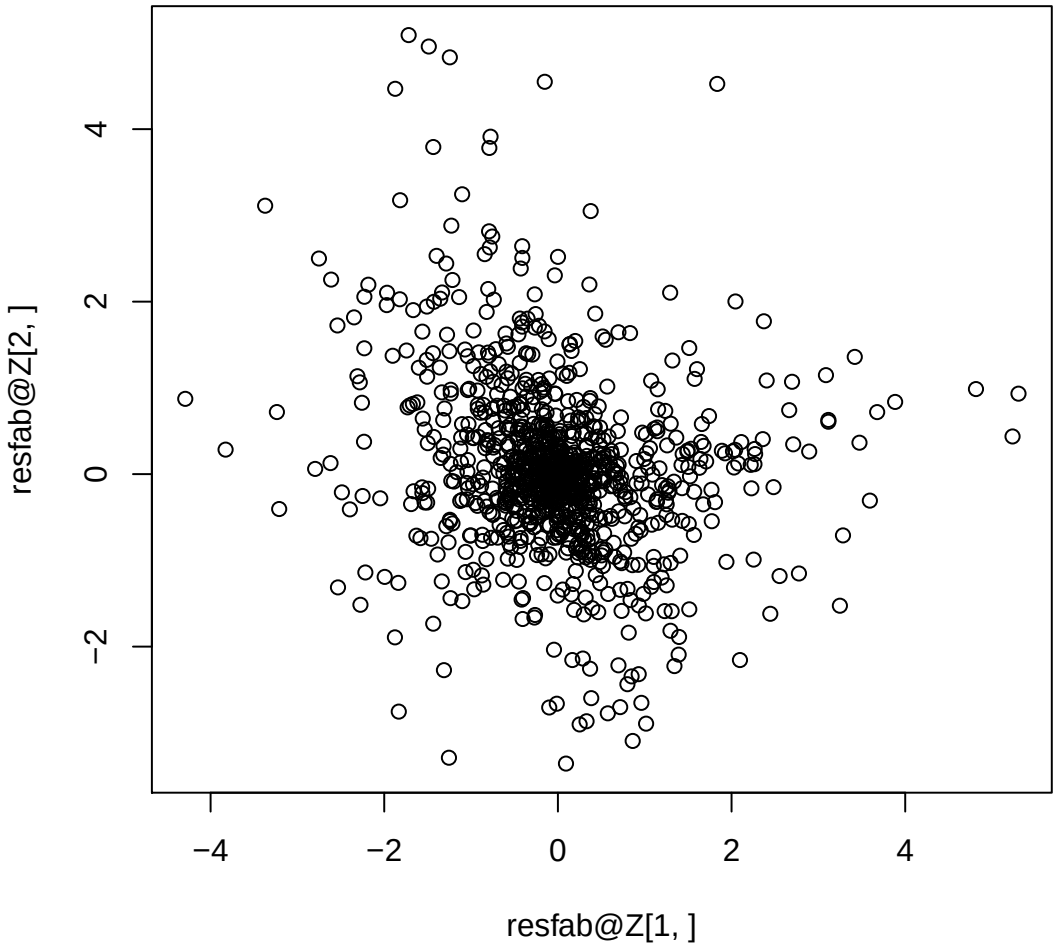


Figure 8.4: The first two components of the results of `fabia` (sparse factor analysis) applied to the mixture of 50-dimensional super-Gaussians.

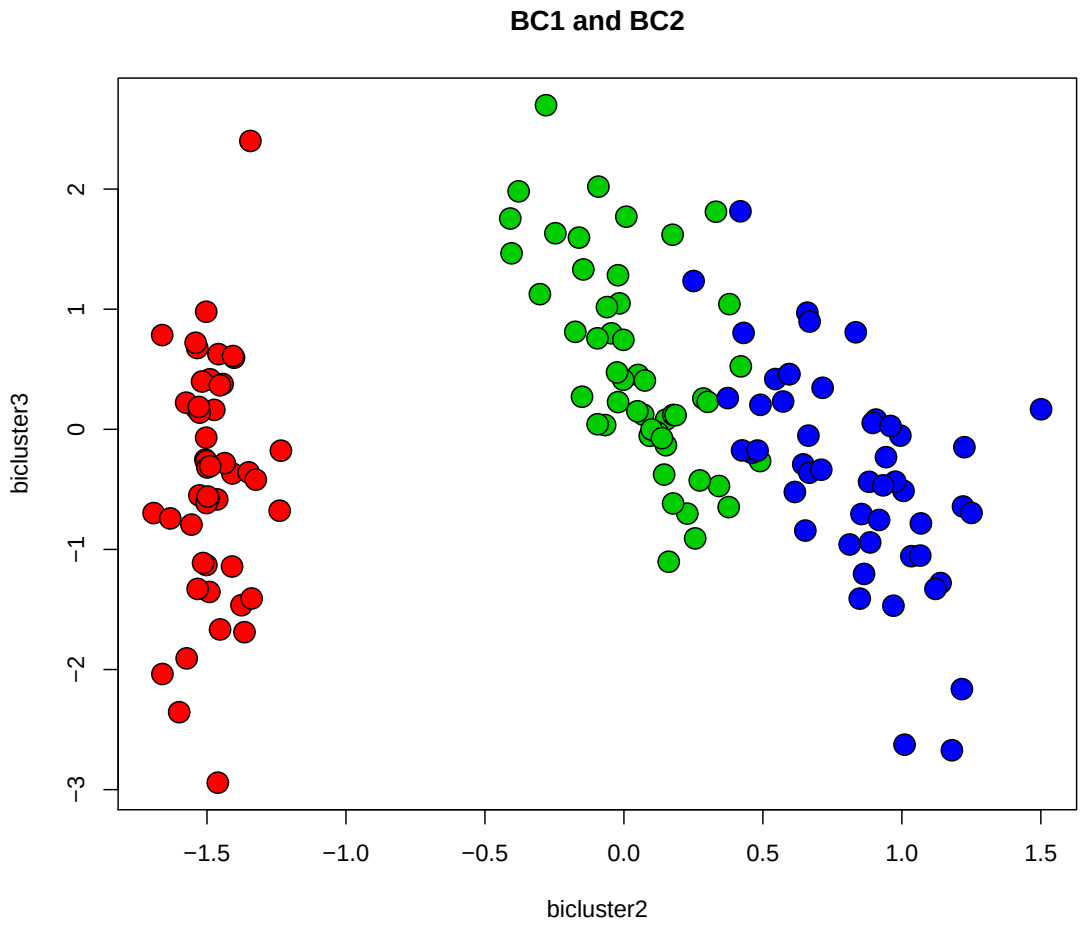


Figure 8.5: FABIA applied to the iris data set.

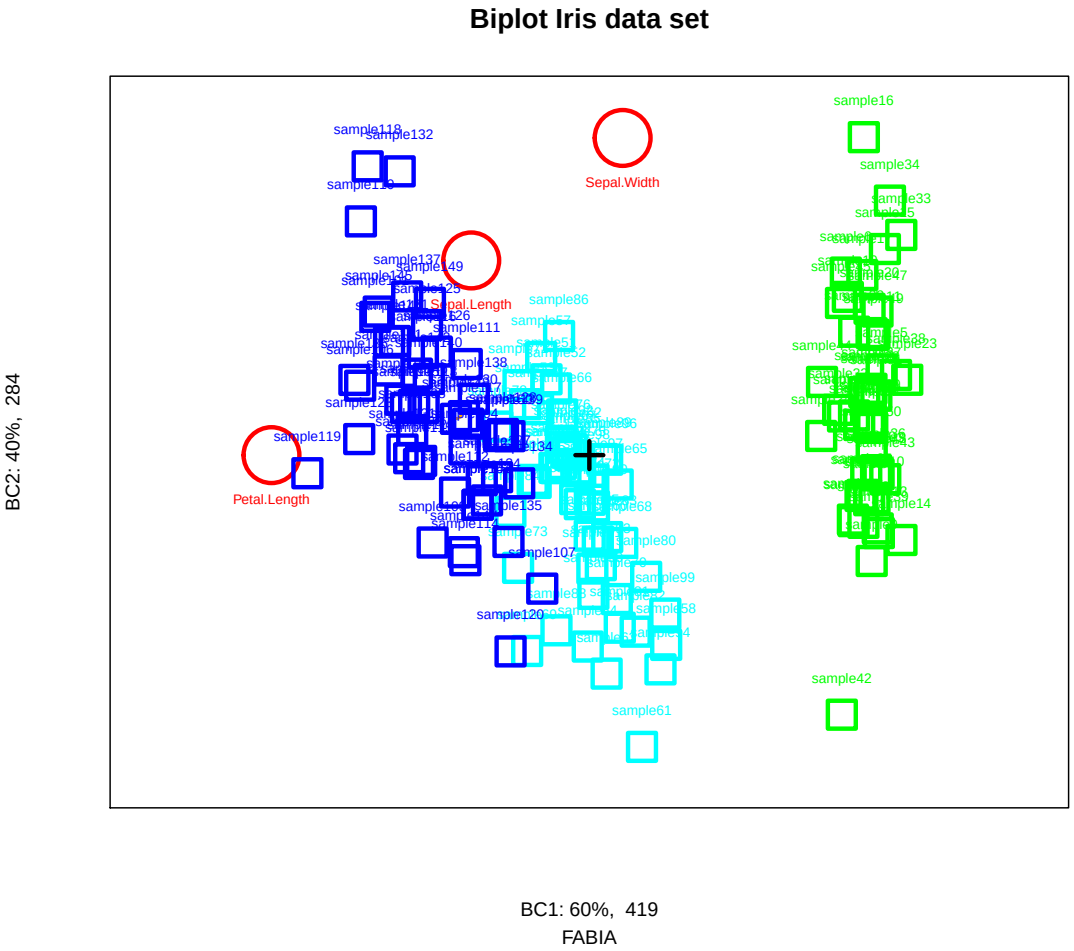


Figure 8.6: Biplot of the result of FABIA applied to the iris data set. The red circles are the most relevant features. From the location of the features we see that BC1 is driven by "Petal.Length" and BC2 by "Sepal.Width". "Sepal.Length" participates at both biclusters as it has both an x-axis and y-axis extension.

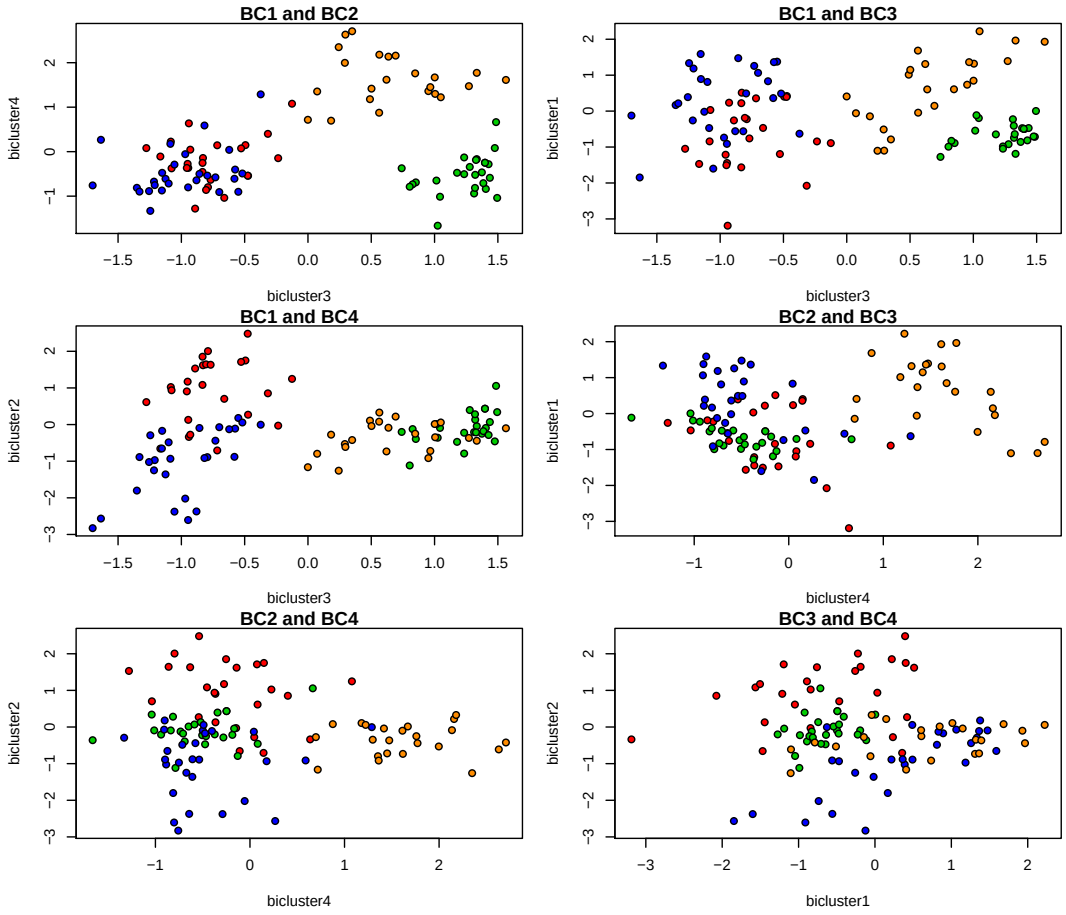


Figure 8.7: FABIA biclustering applied to multiple tissue data with 4 components after 200 iterations. The first bicluster ("bicluster2") separates the prostate samples (green) from the rest. The second bicluster ("bicluster3") separates the colon samples (orange) from the rest. The fourth bicluster ("bicluster1") separates the lung samples (blue) from the rest, though not perfectly.

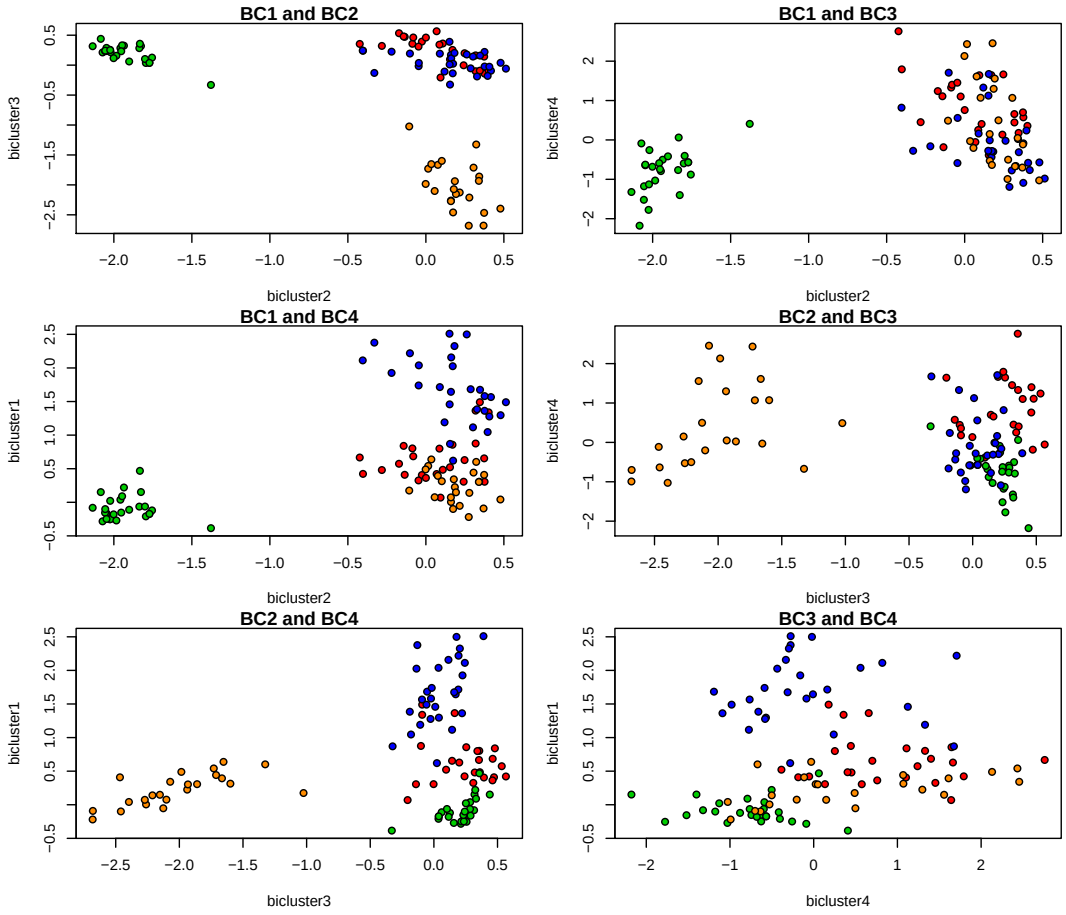


Figure 8.8: FABIA biclustering applied to multiple tissue data with 4 components after 500 iterations. The first bicluster ("bicluster2") separates the prostate samples (green) from the rest. The second bicluster ("bicluster3") separates the colon samples (orange) from the rest. The fourth bicluster ("bicluster1") separates the lung samples (blue) from the rest, though not perfectly.

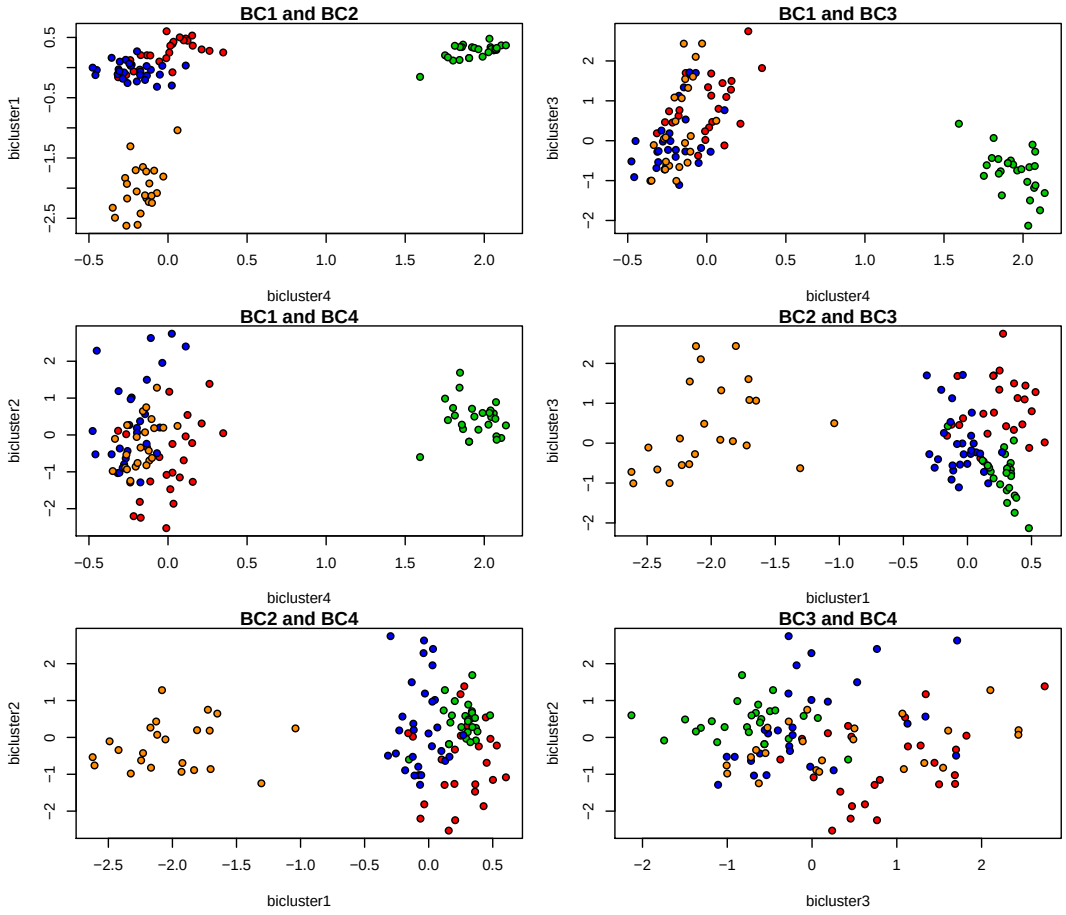


Figure 8.9: FABIA biclustering applied to multiple tissue data with 4 components after 2,000 iterations. The first bicluster ("bicluster2") separates the prostate samples (green) from the rest. The second bicluster ("bicluster3") separates the colon samples (orange) from the rest. The fourth bicluster ("bicluster1") separates the lung samples (blue) from the rest, though not perfectly.

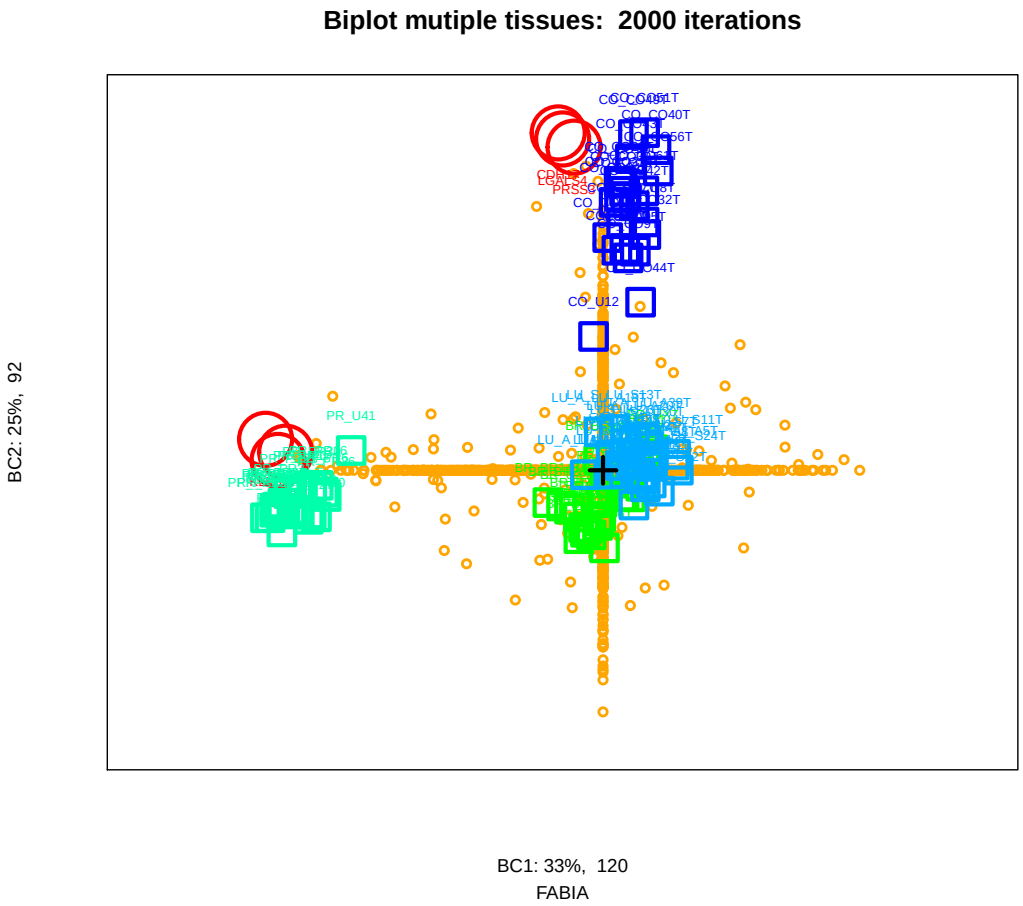


Figure 8.10: Biplot of the first and second bicluster of FABIA applied to multiple tissue data with 4 components after 2,000 iterations. The large red circles are the features (genes) driving the bicluster while the golden small circles are features with minor influence. Sparseness pushes most features to zero in one direction (the axes are golden due to the features pushed to zero). The turquoise prostate samples at the x-axis are separated from the other samples.

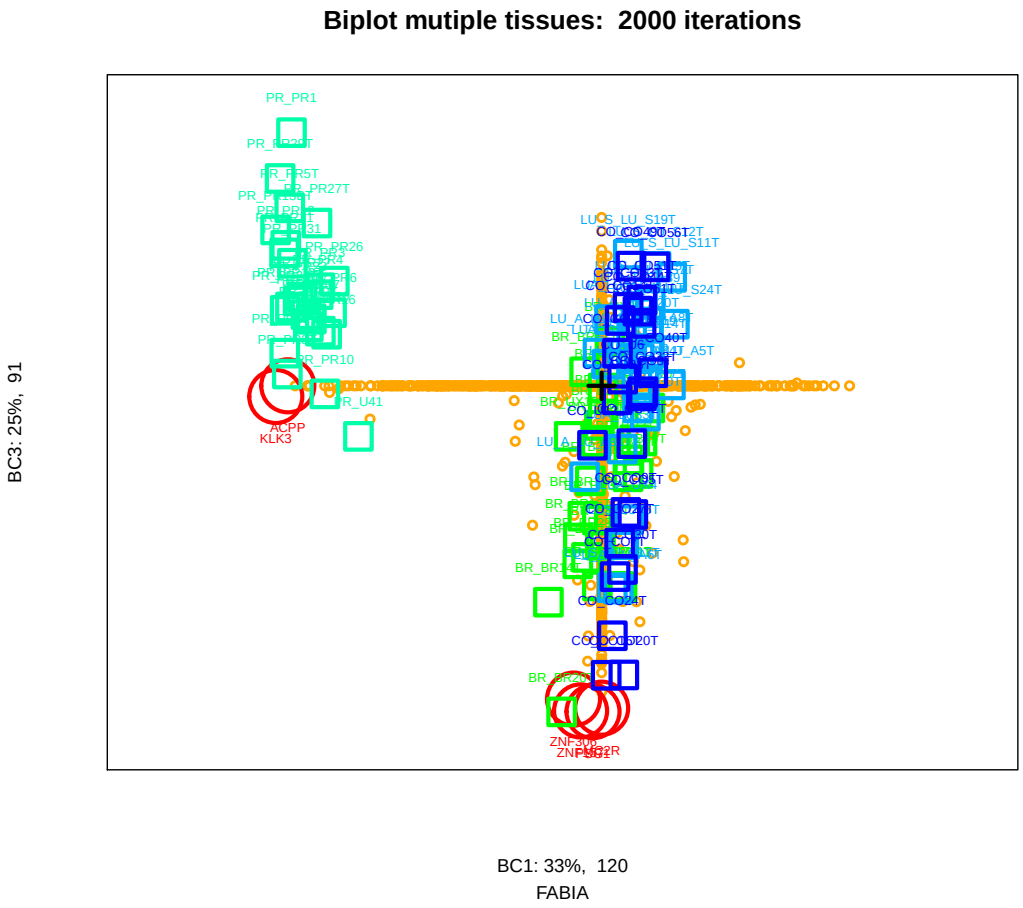


Figure 8.11: Biplot of the first and third bicluster of FABIA applied to multiple tissue data with 4 components after 2,000 iterations. For the first bicluster at the x-axis the genes KLK3 (prostate specific antigen) and ACPP (secreted by the epithelial cells of the prostate gland) can be recognized. These genes separate the turquoise prostate samples at the x-axis from the other samples.

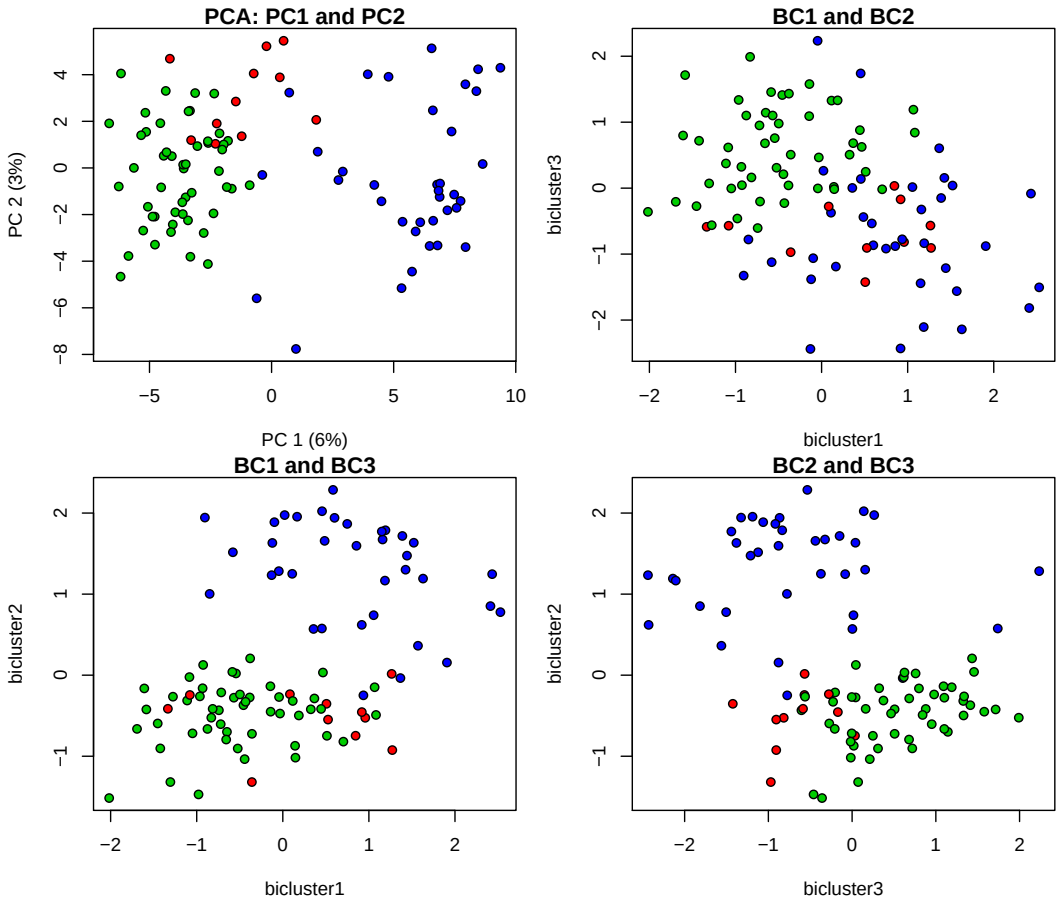


Figure 8.12: FABIA applied to the breast cancer data set with 3 biclusters. The upper left panel shows the projection to the first two principal components. Both PCA and biclustering separate the blue class but have problems to separate the red class.

The **breast cancer data set** consists of microarray data from the Broad Institute "Cancer Program Data Sets" which was produced by van't Veer et al. [2002]. Goal of van't Veer et al. [2002] was to find a gene signature to predict the outcome of a cancer therapy, that is, to predict whether metastasis will occur. A signature of 70 genes has been discovered. We removed array S54, because we identified it as an outlier. Thereafter, the data set contains 97 samples for which 1213 gene expression values are provided — these genes were selected by the authors.

Hoshida et al. [2007] found 3 subclasses and verified that 50/61 cases from class 1 and 2 were estrogen receptor (ER) positive and only 3/36 from class 3. The subclasses were reconfirmed by an independent second breast cancer data set. The three subclasses are indicated in the data set.

We test FABIA on the breast cancer data, where Hoshida et al. [2007] found 3 subclasses which are marked by colors in the plots. Fig. 8.12 shows the results of PCA (the first two principal components) and FABIA biclustering. Both FABIA and PCA separate the subclasses quite well, especially the blue subclass. Fig. 8.13 shows the biplot for bicluster 2 and 3. The subclasses are quite well separated.

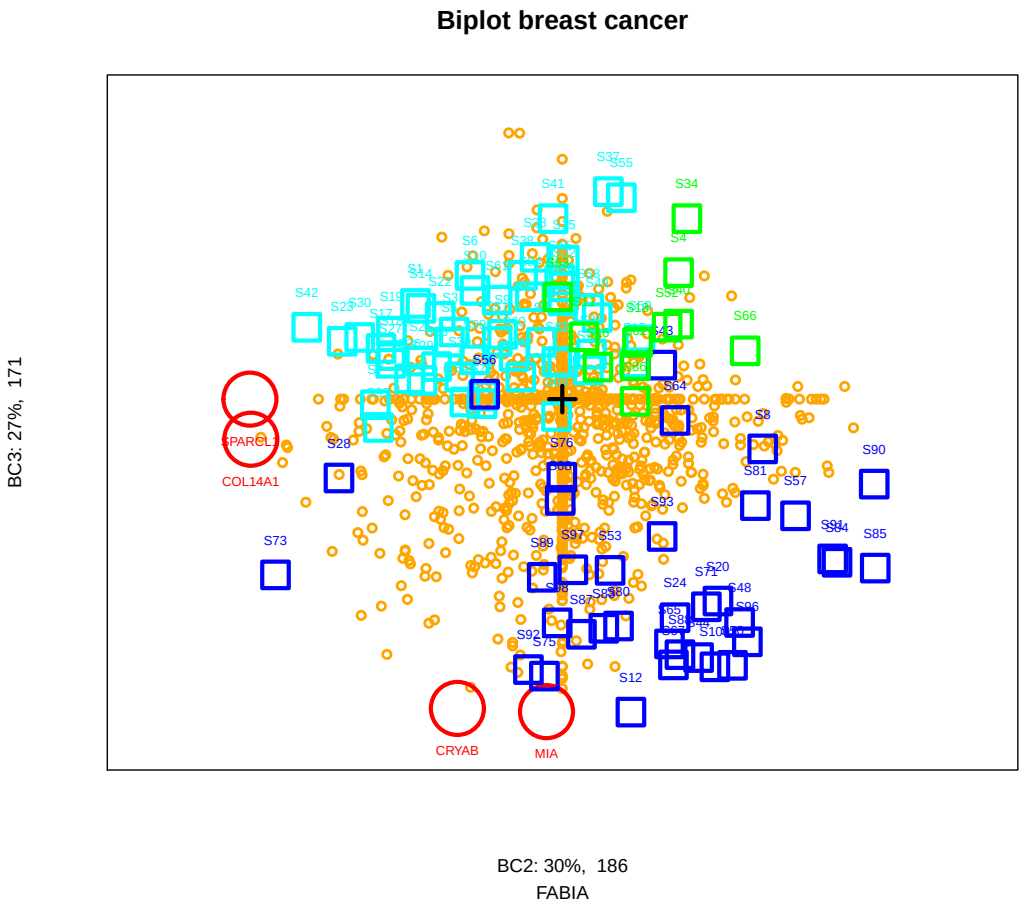


Figure 8.13: FABIA applied to the breast cancer data set with 3 biclusters. The biplot for bicluster 2 and 3. The subclasses are quite good separated.

Finally, we apply FABIA to the DLBCL data set. This is another microarray data set from the Broad Institute "Cancer Program Data Sets" which was produced by Rosenwald et al. [2002]. The gene expression profile of diffuse large-B-cell lymphoma (DLBCL) was measured. Goal was to predict the survival after chemotherapy. The data set consists of 180 DLBCL samples with 661 preselected genes.

Hoshida et al. [2007] divided the data set into three subclasses:

- "OxPhos" (oxidative phosphorylation),
- "BCR" (B-cell response), and
- "HR" (host response).

These subclasses were confirmed on independent DLBCL data. We mark these subclasses in the data set.

Fig. 8.14 shows the results of PCA (the first two principal components) and FABIA biclustering. PCA separates the blue subclass. Biclustering separates the subclasses better because the red cluster is more separated. Fig. 8.15 shows the biplot for bicluster 1 and 2. The subclasses are quite good separated.

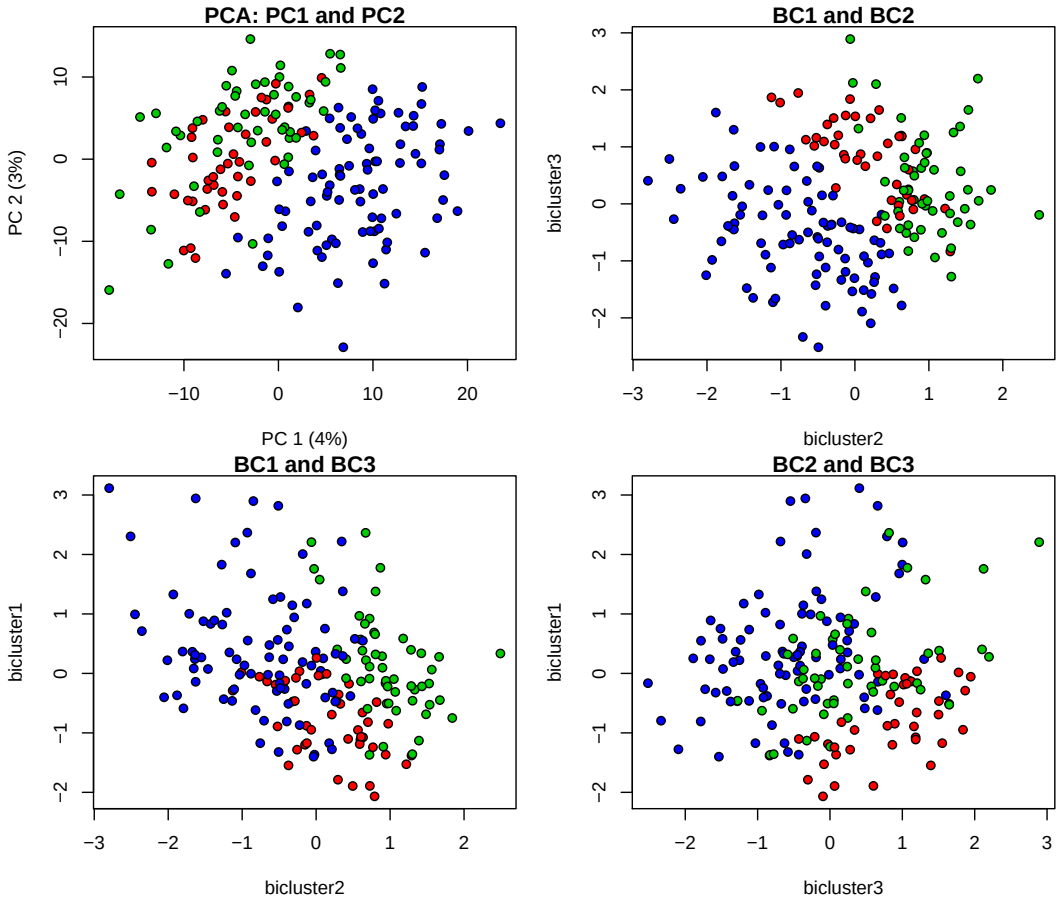


Figure 8.14: FABIA applied to the DLBCL data set with 3 biclusters. The upper left panel shows the projection to the first two principal components where the blue subclass is separated. Biclustering separates the subclasses better because the red cluster is more separated.

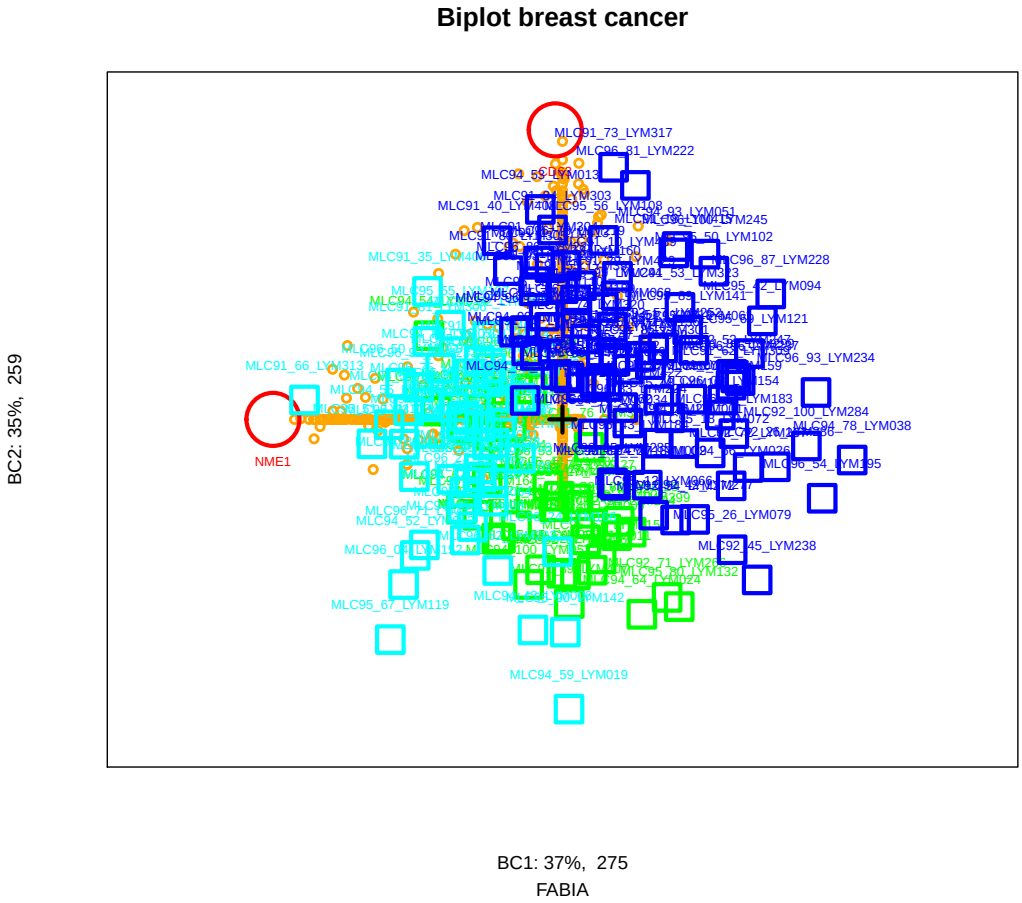


Figure 8.15: FABIA applied to the DLBCL data set with 3 biclusters. The biplot for bicluster 1 and 2. The subclasses are quite good separated.

Chapter 9

Hidden Markov Models

This is the first chapter devoted to unsupervised learning, especially to generative models. The most prominent generative model in bioinformatics is the hidden Markov model. It is well suited for analyzing protein or DNA sequences because of its discrete nature.

9.1 Hidden Markov Models in Bioinformatics

A hidden Markov model (HMM) is a generative approach for generating output sequences. The model is able to assign to each sequence a certain probability of being produced by the current model. The sequences of a class are used to build a model so that these sequences have high probability of being produced by the model. Thereafter the model can be utilized to search for sequences which also have high probability as being produced by the model. Therefore the new sequences with high probability are assumed to be similar to the sequences from which the model is build.

The HMM is able to model certain patterns in sequences and if those patterns are detected, the probability of the sequence is increased.

HMMs for gene prediction.

The DNA is scanned and exons and introns are identified from which the coding region of the gene can be obtain. Translating the coding regions of the gene gives a protein sequences. HMMs are a standard tool for identifying genes in a genome. GENSCAN [Burge and Karlin, 1997] and other HMM approaches to gene prediction [Kulp et al., 1996, Krogh, 1997, Krogh et al., 1994b] have a base-pair specificity between 50% and 80%.

Profile HMMs.

Profile HMMs [Krogh et al., 1994a] are used to store a multiple alignment in a hidden Markov model. An HMM is better suited for storing the alignment than a consensus string because it is a generative model. Especially new sequences can be be evaluated according to their likelihood of being produced by the model. Also the likelihood can be fine tuned after storing the alignment.

If HMMs are build from unaligned sequences, they often stick in local likelihood maxima. Approaches exist which try to avoid them, e.g. deterministic annealing ("Userguide" to HMMER version 1.8). Because of the poor results with unaligned sequences despite annealing approaches, in the new version of HMMER the HMMs are only initialized by alignment results.

The most common software packages for profile HMMs are HMMER [Eddy, 1998] (`http://hmmer.wustl.edu/`) and SAM [Krogh et al., 1994a] (`http://www.cse.ucsc.edu/compbio/sam.html`).

However HMMs have drawbacks as Sean Eddy writes [Eddy, 2004]:

> "HMMs are reasonable models of linear sequence problems, but they don't deal well with correlations between residues or states, especially long-range correlations. HMMs assume that each residue depends only on one underlying state, and each state in the state path depends only on one previous state; otherwise, residues and states are independent of each other." ... "The state path of an HMM has no way of remembering what a distant state generated when a second state generates its residue."

Real valued protein characteristics like hydrophobic profiles cannot be represented by HMMs. HMMs cannot detect higher order correlations, cannot consider dependencies between regions in the sequence, cannot deal with correlations of elements within regions, cannot derive or process real valued protein characteristics, cannot take into account negative examples during model selection, and do not work sufficiently well for unaligned sequences.

Other HMMs Applications.

HMMs were used for remote homology detection [Park et al., 1998, Karplus et al., 1998, 1999] which is weak sequence homology [Sjölander et al., 1996].

HMMs were used for scoring [Barrett et al., 1997] and are combined with trees [Lio et al., 1999].

A whole data base is build on HMMs, namely the PFAM (protein family database) [Bateman et al., 2004, 2000]. Here protein families are classified by HMMs. Software exists for large data sets or proteins like SMART (simple modular architecture research tool) [Schultz et al., 2000].

9.2 Hidden Markov Model Basics

A hidden Markov model (HMM) is a graph of connected hidden states $u \in \{1, \ldots, S\}$, where each state produces a probabilistic output.

Fig. 9.1 shows a hidden Markov model with two state values 1 and 0 which are associated with "on" and "off". If the switch is "on" then it can remain on or go to the value "off". If the switch is "off" then it can remain off or go to the value "on". The state may be hidden in the sense that the position of the switch cannot be observed.

The model evolves over time t (in bioinformatics time is replaced by sequence position). At each step the process jumps from the current state into another state or remains in the current state. The evolving of the state variable u over time can be expressed by introducing the variable u_t for each time point t. At each time t the variable u_t has a certain value $u_t \in \{1, \ldots, S\}$. Fig. 9.2 shows a hidden Markov model where the state variable evolves over time.

It is possible to present all possible sequences of values of the hidden state like in Fig. 9.3. Each path in the figure from left to right is a possible sequence of state values. The probability of taking a certain value at a certain time (e.g. $u_5 = 3$) is the sum over all path' from the left to this value and time.

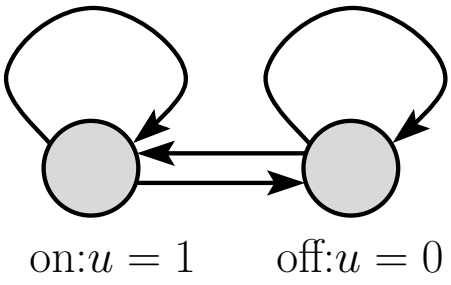


Figure 9.1: A simple hidden Markov model, where the state u can take on one of the two values 0 or 1. This model represents the switch for a light: it is either "on" or "off" and at every time point it can remain in its current state (reccurent connections) or go to the opposite state.

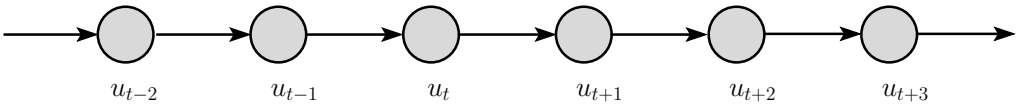


Figure 9.2: A simple hidden Markov model. The state u evolves over time and at each time t the state u takes on the value u_t.

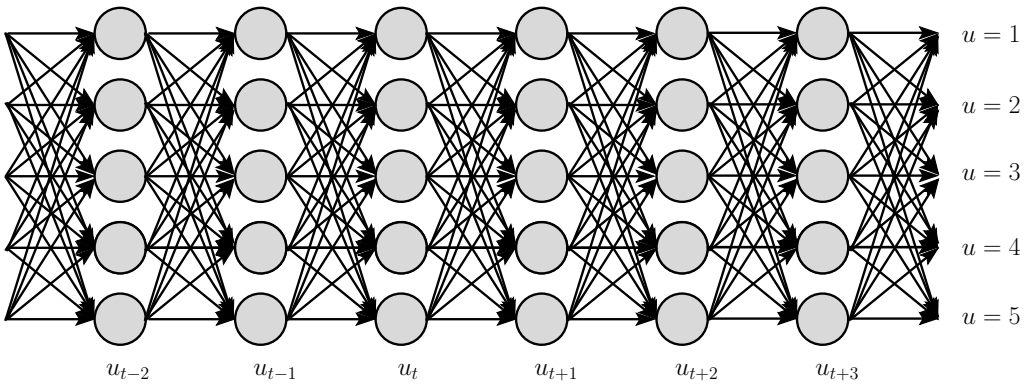


Figure 9.3: The hidden Markov model from Fig. 9.2 in more detail where also the state values $u = 1, \ldots, u = 5$ are given ($S = 5$). At each time t the state u_t takes on one of these values and if the state moves on the value may change. Each path from left to right has a certain probability and the probability of taking a certain value at a certain time is the sum over all paths from the left to this value and time.

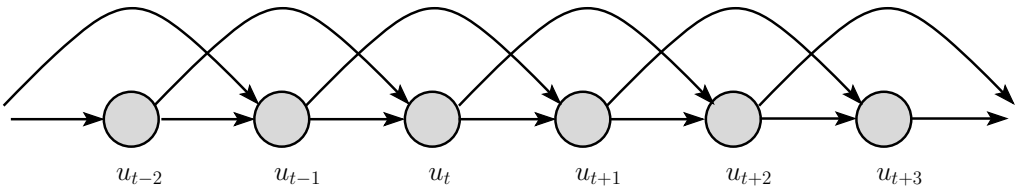


Figure 9.4: A second order hidden Markov model. The transition probability does not depend only on the current state value but also on the previous state value.

The hidden Markov model has transition probabilities $p(a \mid b)$, where $a, b \in \{1, \ldots, S\}$ and b is the current state and a the next state. Here the Markov assumption is that the next state only depends on the current state. Higher order hidden Markov models assume that the probability of going into the next state depends on the current state and previous states. For example in a second order Markov model the transition probability is $p(a \mid b, c)$, where $a, b, c \in \{1, \ldots, S\}$ and b is the current state, c the previous state, and a the next state. The second order Markov model is depicted in Fig. 9.4.

We will focus on a first order hidden Markov model, where the probability of going into a state depends only on the actual state.

At each time the state variable u takes on the value u_t and has a previous value given by u_{t-1}, therefore we observed the transition from u_{t-1} to u_t. This transition has probability of $p(u_t \mid u_{t-1})$,

Assume we have a certain start state probability $p_S(u_1)$ then the probability of observing the sequence $u^T = (u_1, u_2, u_3, \ldots, u_T)$ of length T is

$$p(u^T) \;=\; p_S(u_1) \prod_{t=2}^{T} p(u_t \mid u_{t-1}) \,. \tag{9.1}$$

For example a sequence may be $(u_1 = 3, u_2 = 5, u_3 = 2, \ldots, u_T = 4)$ that is the sequence $(3, 5, 2, \ldots, 4)$. Fig. 9.5 shows the hidden Markov model from Fig. 9.3 where now the transition probabilities are marked including the start state probability p_S.

Now we will consider Markov models which actually produce data, that means they are used as generative models. We assume that each state value has an emission probability $p_E(x_t \mid u_t)$ of emitting a certain output. Here u_t is a value of the state variable at time t (e.g. $u_t = 2$) and x_t is an element of the output alphabet of size P, for example $x_t \in \{A, T, C, G\}$. A specific emission probability may be $p_E(A \mid 2)$. Fig. 9.6 shows a hidden Markov model with output emission.

Fig. 9.7 shows a HMM where the output sequence is the Shine-Dalgarno pattern for ribosome binding regions.

Each output sequence has a certain probability of being produced by the current model. The joint probability of the output sequence $x^T = (x_1, x_2, x_3, \ldots, x_T)$ of length T and the hidden state value sequence $u^T = (u_1, u_2, u_3, \ldots, u_T)$ of length T is

$$p(u^T, x^T) \;=\; p_S(u_1) \prod_{t=2}^{T} p(u_t \mid u_{t-1}) \prod_{t=1}^{T} p_E(x_t \mid u_t) \,. \tag{9.2}$$

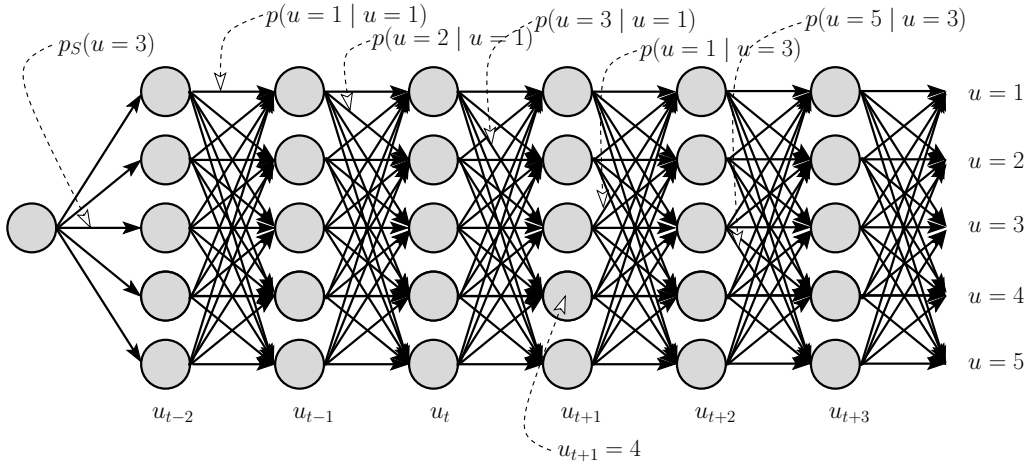


Figure 9.5: The hidden Markov model from Fig. 9.3 where now the transition probabilities are marked including the start state probability p_S. Also the state value $u_{t+1} = 4$ is marked.

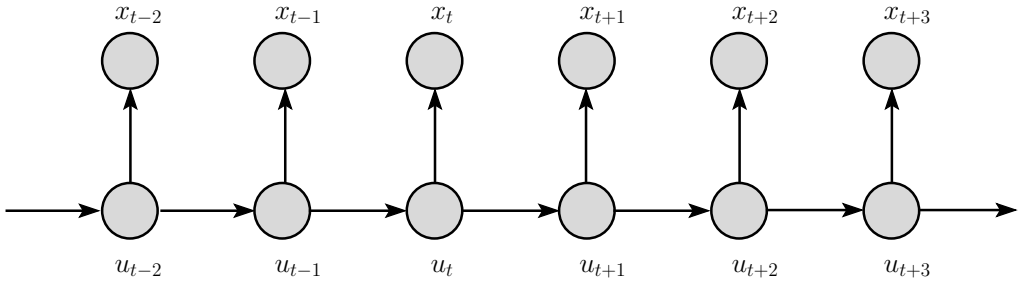


Figure 9.6: A simple hidden Markov model with output. At each time t the hidden state u_t has a certain probability of producing the output x_t.

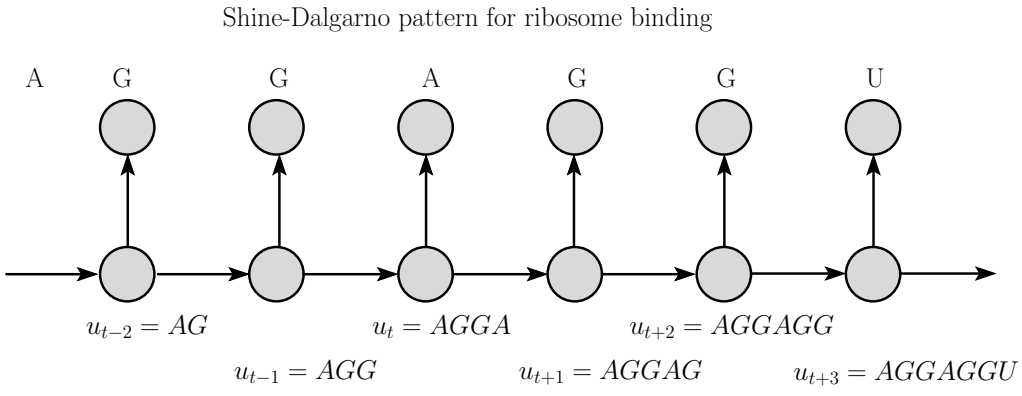


Figure 9.7: An HMM which supplies the Shine-Dalgarno pattern where the ribosome binds. Each state value is associated with a prefix sequence of the Shine-Dalgarno pattern. The state value for "no prefix" is omitted in the figure.

Through marginalization we obtain the probability of the output sequence x^T being produced by the HMM:

$$p(x^T) \;=\; \sum_{u^T} p(u^T, x^T) \;=\; \sum_{u^T} p_S(u_1) \prod_{t=2}^{T} p(u_t \mid u_{t-1}) \prod_{t=1}^{T} p_E(x_t \mid u_t)\;, \tag{9.3}$$

where $\sum_{u^T}$ denotes the sum over all possible sequences of length T of the values $\{1, \ldots, S\}$. The sum $\sum_{u^T}$ has S^T summands corresponding to different sequences (S values for u_1 multiplied by S values for u_2 etc.).

Fortunately, the (first order) Markov assumption allows to recursively compute above sum. We denote with $x^t = (x_1, x_2, x_3, \ldots, x_t)$ the prefix sequence of x^T of length t. We introduce the probability $p(x^t, u_t)$ of the model producing x^t and being in state u_t at the end.

$$\begin{aligned}
p(x^t, u_t) \;&=\; p(x_t \mid x^{t-1}, u_t)\, p(x^{t-1}, u_t) \;= \\
&p_E(x_t \mid u_t) \sum_{u_{t-1}} p(x^{t-1}, u_t, u_{t-1}) \;= \\
&p_E(x_t \mid u_t) \sum_{u_{t-1}} p(u_t \mid x^{t-1}, u_{t-1})\, p(x^{t-1}, u_{t-1}) \;= \\
&p_E(x_t \mid u_t) \sum_{u_{t-1}} p(u_t \mid u_{t-1})\, p(x^{t-1}, u_{t-1})\;,
\end{aligned} \tag{9.4}$$

where the Markov assumptions $p(x_t \mid x^{t-1}, u_t) \;=\; p_E(x_t \mid u_t)$ on the output emission and $p(u_t \mid x^{t-1}, u_{t-1}) \;=\; p(u_t \mid u_{t-1})$ on the transitions is used. Further marginalization $p(x^{t-1}, u_t) \;=\; \sum_{u_{t-1}} p(x^{t-1}, u_t, u_{t-1})$ and the definition of conditional probabilities $p(x^{t-1}, u_t, u_{t-1}) \;=\; p(u_t \mid x^{t-1}, u_{t-1})\, p(x^{t-1}, u_{t-1})$ were applied.

That means each recursion step needs only a sum over all u_{t-1} which is a sum over S values. However we have to do this for each value of u_t, therefore the recursion has complexity of $O(T\, S^2)$. The complexity can be reduced if transition probabilities are zero. The recursion starts with

$$p(x^1, u_1) \;=\; p_S(u_1)\, p_E(x_1 \mid u_1)\;. \tag{9.5}$$

The final probability of x^T can be computed as

$$p(x^T) \;=\; \sum_{u_T} p(x^T, u_T)\;. \tag{9.6}$$

This algorithm is called the "forward pass" or the "forward phase" and is used to compute the probability of x^T which is equal to the likelihood of x^T because we have discrete values. Alg. 9.1 shows the algorithm for the forward phase to compute the likelihood for one sequence for an HMM.

9.3 Expectation Maximization for HMM: Baum-Welch Algorithm

Now we focus on learning and parameter selection based on a training set.

Algorithm 9.1 HMM Forward Pass

Given: sequence $x^T = (x_1, x_2, x_3, \ldots, x_T)$, state values $u \in \{1, \ldots, S\}$, start probabilities $p_S(u_1)$, transition probabilities $p(u_t \mid u_{t-1})$, and emission probabilities $p_E(x_t \mid u_t)$; Output: likelihood $p(x^T)$ and $p(x^t, u_t)$

BEGIN initialization

$$p(x^1, u_1) \;=\; p_S(u_1)\, p_E(x_1 \mid u_1)$$

END initialization
BEGIN Recursion
for $(t = 2\,;\; t \;\leq\; T\,;\; t++)$ **do**
for $(a = 1\,;\; a \;\leq\; S\,;\; a++)$ **do**

$$p(x^t, u_t = a) \;=\; p_E(x_t \mid u_t = a) \sum_{u_{t-1}=1}^{S} p(u_t = a \mid u_{t-1})\, p(x^{t-1}, u_{t-1})$$

end for
end for
END Recursion
BEGIN Compute Likelihood

$$p(x^T) \;=\; \sum_{a=1}^{S} p(x^T, u_T = a)$$

END Compute Likelihood

The parameters of a hidden Markov model are the S start probabilities $p_S(u_1)$, the S^2 transitions probabilities $p(u_t \mid u_{t-1})$, and the $S\ P$ emission probabilities $p_E(x_t \mid u_t)$ (P is the number of output symbols). If we have a set of training sequences $\{\boldsymbol{x}^i\}, 1 \leq i \leq l$, then the parameters can be optimized by maximizing the likelihood. Instead of gradient based methods, we deduce an Expectation Maximization algorithm.

We define

$$\mathcal{F}(Q, \boldsymbol{w}) \ = \ \int_U Q(\boldsymbol{u} \mid \boldsymbol{x}) \ \ln p(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{w}) \ d\boldsymbol{u} \ - \tag{9.7}$$
$$\int_U Q(\boldsymbol{u} \mid \boldsymbol{x}) \ \ln Q(\boldsymbol{u} \mid \boldsymbol{x}) \ d\boldsymbol{u} \ ,$$

where $Q(\boldsymbol{u} \mid \boldsymbol{x})$ is an estimation for $p(\boldsymbol{u} \mid \boldsymbol{x}; \boldsymbol{w})$.

We have to adapt this formulation to discrete HMMs. For HMMs $\boldsymbol{u}$ is the sequence of hidden states, $\boldsymbol{x}$ the sequence of output states and $\boldsymbol{w}$ summarizes all probability parameters (start, transition, and emission) in the model. The integral $\int_U d\boldsymbol{u}$ can be replaced by a sum.

The estimation for the state sequence can be written as

$$Q(\boldsymbol{u} \mid \boldsymbol{x}) \ = \ p(u_1 = a_1, u_2 = a_2, \ldots, u_T = a_T \mid x^T; \boldsymbol{w}) \ . \tag{9.8}$$

We obtain

$$\mathcal{F}(Q, \boldsymbol{w}) \ = \ \sum_{a_1=1}^{S} \ldots \sum_{a_T=1}^{S} \tag{9.9}$$
$$p(u_1 = a_1, u_2 = a_2, \ldots, u_T = a_T \mid x^T; \boldsymbol{w}) \ \ln p(x^T, u^T; \boldsymbol{w}) \ -$$
$$\sum_{a_1=1}^{S} \ldots \sum_{a_T=1}^{S} p(u_1 = a_1, u_2 = a_2, \ldots, u_T = a_T \mid x^T; \boldsymbol{w})$$
$$\ln p(u_1 = a_1, u_2 = a_2, \ldots, u_T = a_T \mid x^T; \boldsymbol{w}) \ = \tag{9.10}$$
$$\sum_{a_1=1}^{S} \ldots \sum_{a_T=1}^{S} p(u_1 = a_1, u_2 = a_2, \ldots, u_T = a_T \mid x^T; \boldsymbol{w}) \ \ln p(x^T, u^T; \boldsymbol{w})$$
$$- \ c \ ,$$

where c is a constant independent of $\boldsymbol{w}$.

We have

$$\ln p(u^T, x^T; \boldsymbol{w}) \ = \ \ln p_S(u_1) \ + \ \sum_{t=2}^{T} \ln p(u_t \mid u_{t-1}) \ + \ \sum_{t=1}^{T} \ln p_E(x_t \mid u_t) \ . \tag{9.11}$$

Because most variables a_t can be summed out we obtain:

$$\begin{aligned}\mathcal{F}(Q, \boldsymbol{w}) \;=\; & \sum_{a=1}^{S} p(u_1 = a \mid x^T; \boldsymbol{w}) \;\ln p_S(u_1 = a) \;+ \\ & \sum_{t=1}^{T} \sum_{a=1}^{S} p(u_t = a \mid x^T; \boldsymbol{w}) \;\ln p_E(x_t \mid u_t = a) \;+ \\ & \sum_{t=2}^{T} \sum_{a=1}^{S} \sum_{b=1}^{S} p(u_t = a, u_{t-1} = b \mid x^T; \boldsymbol{w}) \;\ln p(u_t = a \mid u_{t-1} = b) \;- c\,.\end{aligned} \tag{9.12}$$

Note that the parameters $\boldsymbol{w}$ are the start probabilities $p_S(a)$, the emission probabilities $p_E(x \mid a)$, and the transition probabilities $p(a \mid b)$. We have as constraints $\sum_a p_S(a) \;=\; 1$, $\sum_x p_E(x \mid a) \;=\; 1$, and $\sum_a p(a \mid b) \;=\; 1$.

Consider the optimization problem

$$\begin{aligned}\min_{\boldsymbol{w}} \;\; & \sum_t \sum_a c_{ta} \;\ln w_a \\ \text{s.t.} \;\; & \sum_a w_a \;=\; 1\,.\end{aligned} \tag{9.13}$$

The Lagrangian is

$$L \;=\; \sum_t \sum_a c_{ta} \;\ln w_a \;-\; \lambda \left(\sum_a w_a \;-\; 1 \right)\,. \tag{9.14}$$

Optimality requires

$$\frac{\partial L}{\partial w_a} \;=\; \sum_t c_{ta} \;\frac{1}{w_a} \;-\; \lambda \;=\; 0 \tag{9.15}$$

therefore

$$\sum_t c_{ta} \;-\; \lambda \, w_a \;=\; 0 \tag{9.16}$$

and summing over a gives

$$\sum_a \sum_t c_{ta} \;=\; \lambda\,. \tag{9.17}$$

We obtain

$$w_a \;=\; \frac{\sum_t c_{ta}}{\sum_a \sum_t c_{ta}}\,. \tag{9.18}$$

The constraint maximization step (M-step) is therefore

$$p_S(a) \;=\; \frac{p(u_1 = a \mid x^T; \boldsymbol{w})}{\sum_{a'} p(u_1 = a' \mid x^T; \boldsymbol{w})} \tag{9.19}$$

$$p_E(x \mid a) \;=\; \frac{\sum_{t=1}^{T} \delta_{x_t = x} \; p(u_t = a \mid x^T; \boldsymbol{w})}{\sum_y \sum_{t=1}^{T} \delta_{x_t = y} \; p(u_t = a \mid x^T; \boldsymbol{w})} \tag{9.20}$$

$$p(a \mid b) \;=\; \frac{\sum_{t=2}^{T} p(u_t = a, u_{t-1} = b \mid x^T; \boldsymbol{w})}{\sum_{a'} \sum_{t=2}^{T} p(u_t = a', u_{t-1} = b \mid x^T; \boldsymbol{w})} \tag{9.21}$$

which is

$$p_S(a) \;=\; p(u_1 = a \mid x^T; \boldsymbol{w}) \tag{9.22}$$

$$p_E(x \mid a) \;=\; \frac{\sum_{t=1}^{T} \delta_{x_t = x}\, p(u_t = a \mid x^T; \boldsymbol{w})}{\sum_{t=1}^{T} p(u_t = a \mid x^T; \boldsymbol{w})} \tag{9.23}$$

$$p(a \mid b) \;=\; \frac{\sum_{t=2}^{T} p(u_t = a, u_{t-1} = b \mid x^T; \boldsymbol{w})}{\sum_{t=2}^{T} p(u_{t-1} = b \mid x^T; \boldsymbol{w})} \;. \tag{9.24}$$

We now consider the estimation step (E-step) in order to estimate $p(u_t = a \mid x^T; \boldsymbol{w})$ and $p(u_t = a, u_{t-1} = b \mid x^T; \boldsymbol{w})$. First we have to introduce the suffix sequence $x^{t \leftarrow T} = (x_t, x_{t+1}, \ldots, x_T)$. of length $T - t + 1$.

We use the probability $p(x^{t+1 \leftarrow T} \mid u_t = a)$ of the suffix sequence $x^{t+1 \leftarrow T} = (x_{t+1}, \ldots, x_T)$ being produced by the model if starting from $u_t = a$.

Now we can formulate an expression for $p(u_t = a \mid x^T; \boldsymbol{w})$

$$p(u_t = a \mid x^T; \boldsymbol{w}) \;=\; \frac{p(u_t = a, x^T; \boldsymbol{w})}{p(x^T)} \;= \tag{9.25}$$
$$\frac{p(x^t, u_t = a; \boldsymbol{w})\; p(x^{t+1 \leftarrow T} \mid u_t = a)}{p(x^T)} \;,$$

where the first "=" is the definition of conditional probability and the second "=" says that all paths of hidden values which have at time t the value a can be separated into a prefix path from start to time t ending in the value a and a suffix path starting at time t in a.

Similar we can formulate an expression for $p(u_t = a, u_{t-1} = b \mid x^T; \boldsymbol{w})$

$$p(u_t = a, u_{t-1} = b \mid x^T; \boldsymbol{w}) \;=\; \frac{p(u_t = a, u_{t-1} = b, x^T; \boldsymbol{w})}{p(x^T)} \;= \tag{9.26}$$
$$p(x^{t-1}, u_{t-1} = b; \boldsymbol{w})\; p(u_t = a \mid u_{t-1} = b)\; p_E(x_t \mid u_t = a)$$
$$p(x^{t+1 \leftarrow T} \mid u_t = a) \;/\; p(x^T) \;,$$

where again the first "=" is the conditional probability and the second "=" says all paths which are at time t in state value a and in time $(t-1)$ in state value b can be separated in a prefix path from start to time $(t-1)$ ending in b, a suffix path starting from t in value a to the end, the transition $b \leftarrow a$ with probability $p(u_t = a \mid u_{t-1} = b)$ and the emission of x_t given by $p_E(x_t \mid u_t = a)$.

Note that

$$p(u_t = a \mid x^T; \boldsymbol{w}) \;=\; \sum_{b=1}^{S} p(u_t = a, u_{t-1} = b \mid x^T; \boldsymbol{w}) \;. \tag{9.27}$$

Similar to eq. (9.4) we can derive a backward recursion for computing $p(x^{t+1 \leftarrow T} \mid u_t = a)$ by using the Markov assumptions:

$$p(x^{t+1 \leftarrow T} \mid u_t = a) \;= \tag{9.28}$$
$$\sum_{b=1}^{S} p_E(x_{t+1} \mid u_{t+1} = b)\; p(u_{t+1} = b \mid u_t = a)\; p(x^{t+2 \leftarrow T} \mid u_{t+1} = b) \;.$$

The starting conditions are

$$p(x^{T\leftarrow T} \mid u_{T-1} = a) \;=\; \sum_{b=1}^{S} p_E(x_T \mid u_T = b)\, p(u_T = b \mid u_{T-1} = a) \tag{9.29}$$

or, alternatively

$$\forall_a : \;\; p(x^{T+1\leftarrow T} \mid u_T = a) \;=\; 1 \tag{9.30}$$

In Alg. 9.2 an algorithm for the backward procedure for HMMs is given.

Algorithm 9.2 HMM Backward Pass

Given: sequence $x^T = (x_1, x_2, x_3, \ldots, x_T)$, state values $u \in \{1, \ldots, S\}$, start probabilities $p_S(u_1)$, transition probabilities $p(u_t \mid u_{t-1})$, and emission probabilities $p_E(x_t \mid u_t)$; Output: likelihood $p(x^T)$ and $p(x^{t+1\leftarrow T} \mid u_t = a)$

BEGIN initialization

$$\forall_a : \;\; p(x^{T+1\leftarrow T} \mid u_T = a) \;=\; 1$$

END initialization
BEGIN Recursion
for $(t = T-1 \;;\; t \;\geq\; 1 \;;\; t--)$ **do**
for $(a = 1 \;;\; a \;\leq\; S \;;\; a++)$ **do**

$$p(x^{t+1\leftarrow T} \mid u_t = a) \;=$$
$$\sum_{b=1}^{S} p_E(x_{t+1} \mid u_{t+1} = b)\, p(u_{t+1} = b \mid u_t = a)\, p(x^{t+2\leftarrow T} \mid u_{t+1} = b)\,.$$

end for
end for
END Recursion
BEGIN Compute Likelihood

$$p(x^T) \;=\; \sum_{a=1}^{S} p_S(u_1 = a)\, (p(x^{1\leftarrow T} \mid u_1 = a)$$

END Compute Likelihood

The EM algorithm for HMMs is given in Alg. 9.3 which is based on the forward procedure Alg. 9.1 and the backward procedure Alg. 9.2.

Algorithm 9.3 HMM EM Algorithm

Given: l training sequences $(x^T)^i = (x_1^i, x_2^i, x_3^i, \ldots, x_T^i)$ for $1 \leq i \leq l$, state values $u \in \{1, \ldots, S\}$, start probabilities $p_S(u_1)$, transition probabilities $p(u_t \mid u_{t-1})$, and emission probabilities $p_E(x \mid u)$; Output: updated values of $p_S(u)$, $p_E(x \mid u)$, and $p(u_t \mid u_{t-1})$

BEGIN initialization

initialize start probabilities $p_S(u_1)$, transition probabilities $p(u_t \mid u_{t-1})$, and emission probabilities $p_E(x \mid u)$; Output: updated values of $p_S(u)$, $p_E(x \mid u)$, and $p(u_t \mid u_{t-1})$

END initialization

Stop=false

while Stop=false **do**

for $(i = 1 \,;\, i \geq l \,;\, i++)$ **do**

Forward Pass

forward pass for $(x^T)^i$ according to Alg. 9.1

Backward Pass

backward pass for $(x^T)^i$ according to Alg. 9.2

E-Step

for $(a = 1 \,;\, a \leq S \,;\, a++)$ **do**

for $(b = 1 \,;\, b \leq S \,;\, b++)$ **do**

$$p(u_t = a, u_{t-1} = b \mid (x^T)^i; \boldsymbol{w}) = p((x^{t-1})^i, u_{t-1} = b; \boldsymbol{w})\, p(u_t = a \mid u_{t-1} = b)\, p_E(x_t^i \mid u_t = a)\, p((x^{t+1 \leftarrow T})^i \mid u_t = a) \,/\, p((x^T)^i)$$

end for

end for

for $(a = 1 \,;\, a \leq S \,;\, a++)$ **do**

$$p(u_t = a \mid (x^T)^i; \boldsymbol{w}) = \sum_{b=1}^{S} p(u_t = a, u_{t-1} = b \mid (x^T)^i; \boldsymbol{w})$$

end for

M-Step

for $(a = 1 \,;\, a \leq S \,;\, a++)$ **do**

$$p_S(a) = p(u_1 = a \mid (x^T)^i; \boldsymbol{w})$$

end for

for $(a = 1 \,;\, a \leq S \,;\, a++)$ **do**

for $(x = 1 \,;\, x \leq P \,;\, x++)$ **do**

$$p_E(x \mid a) = \frac{\sum_{t=1}^{T} \delta_{x_t^i = x}\, p(u_t = a \mid (x^T)^i; \boldsymbol{w})}{\sum_{t=1}^{T} p(u_t = a \mid (x^T)^i; \boldsymbol{w})}$$

end for

end for

for $(a = 1 \,;\, a \leq S \,;\, a++)$ **do**

for $(b = 1 \,;\, b \leq S \,;\, b++)$ **do**

$$p(a \mid b) = \frac{\sum_{t=2}^{T} p(u_t = a, u_{t-1} = b \mid (x^T)^i; \boldsymbol{w})}{\sum_{t=2}^{T} p(u_{t-1} = b \mid (x^T)^i; \boldsymbol{w})}$$

end for

end for

end for

if stop criterion fulfilled **then**

Stop=true

end if

end while

9.4 Viterby Algorithm

In the forward (and also backward) pass we computed $p(x^T)$, the probability of producing x^T by the model, that is the likelihood of x^T. The likelihood of x^T is an integral – more exactly a sum – over all probabilities of possible sequences of hidden states multiplied by the probability that the hidden sequence emits x^T.

In many cases a specific hidden sequence $(u^T)^* = (u_1^*, u_2^*, u_3^*, \ldots, u_T^*)$ and its probability of emitting x^T dominates the above sum. More formally

$$(u^T)^* \;=\; \arg\max_{u^T} p(u^T \mid x^T) \;=\; \arg\max_{u^T} p(u^T, x^T)\;. \tag{9.31}$$

$(u^T)^*$ is of interest if the hidden states have a semantic meaning, then one want to extract $(u^T)^*$.

In bioinformatics the extraction of $(u^T)^*$ is important to make an alignment of a sequence with a multiple alignment stored in an HMM.

Because $(u^T)^*$ can be viewed as an alignment, it is not surprising that it can be obtained through dynamic programming. The dynamic programming algorithm has to find a path in Fig. 9.5 from left to right. Towards this end the state values at a certain time which are the circles in Fig. 9.5 are represented by a matrix $\boldsymbol{V}$. $V_{t,a}$ contains the maximal probability of a sequence of length t ending in state value a:

$$V_{t,a} \;=\; \max_{u^{t-1}} p(x^t, u^{t-1}, u_t = a)\;. \tag{9.32}$$

The Markov conditions allow now to formulate $V_{t,a}$ recursively:

$$V_{t,a} \;=\; p_E(x_t \mid u_t = a)\; \max_b p(u_t = a \mid u_{t-1} = b)\; V_{t-1,b} \tag{9.33}$$

with initialization

$$V_{1,a} \;=\; p_S(a) p_E(x_1 \mid u_1 = a) \tag{9.34}$$

and the result

$$\max_{u^T} p(u^T, x^T) \;=\; \max_a V_{T,a}\;. \tag{9.35}$$

The best sequence of hidden states can be found by back-tracing using

$$b(t,a) \;=\; \arg\max_b p(u_t = a \mid u_{t-1} = b)\; V_{t-1,b} \tag{9.36}$$

The complexity of the Viterby algorithm is $O(T\ S^2)$ because all $S\ T$ values $V_{t,a}$ must be computed and for computing them, the maximum over S terms must be determined.

The Viterby algorithm can be used to iteratively improve a multiple alignment:

1 initialize the HMM

2 align all sequences to the HMM via the Viterby algorithm

3 make frequency counts per column and compute the transition probabilities to update the HMM

4 if not converged go to step 2

Algorithm 9.4 HMM Viterby

Given: sequence $x^T = (x_1, x_2, x_3, \ldots, x_T)$, state values $u \in \{1, \ldots, S\}$, start probabilities $p_S(u_1)$, transition probabilities $p(u_t \mid u_{t-1})$, and emission probabilities $p_E(x_t \mid u_t)$; Output: most likely sequence of hidden state values $(u^T)^*$ and its probability $p\left(x^T, (u^T)^*\right)$

BEGIN initialization

$$V_{1,a} = p_S(a) p_E(x_1 \mid u_1 = a)$$

END initialization
BEGIN Recursion
for $(t = 2\ ;\ t \leq T\ ;\ t++)$ **do**
for $(a = 1\ ;\ a \leq S\ ;\ a++)$ **do**

$$V_{t,a} = p_E(x_t \mid u_t = a) \max_b p(u_t = a \mid u_{t-1} = b)\ V_{t-1,b}$$

$$b(t, a) = \arg\max_b p(u_t = a \mid u_{t-1} = b)\ V_{t-1,b}$$

end for
end for
END Recursion
BEGIN Compute Probability

$$p\left(x^T, (u^T)^*\right) = \max_{a=1}^{S} V(T, a)$$

END Compute Probability
BEGIN Back-tracing

$$s = \arg\max_{a=1}^{S} V(T, a)$$

print s

for $(t = T\ ;\ t \geq 2\ ;\ t--)$ **do**

$$s = b(t, s)$$

print s

end for
END Back-tracing

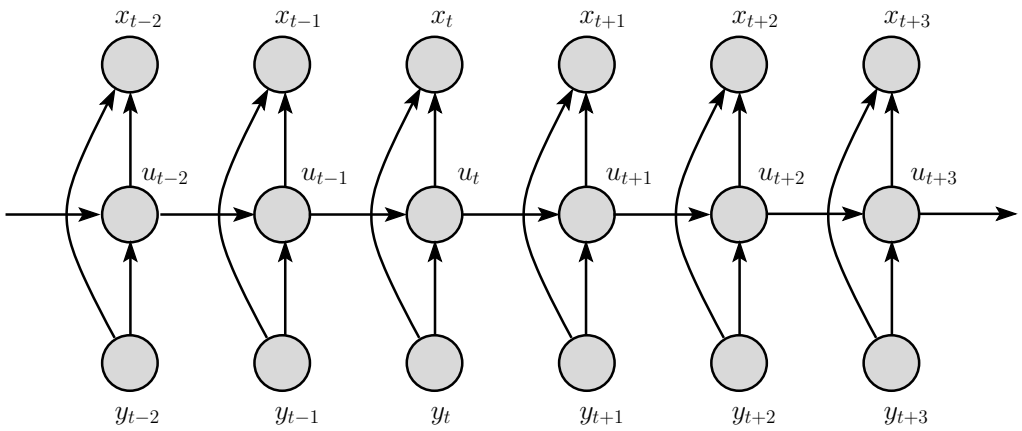


Figure 9.8: An input output HMM (IOHMM) where the output sequence $x^T = (x_1, x_2, x_3, \ldots, x_T)$ is conditioned on the input sequence $y^T = (y_1, y_2, y_3, \ldots, y_T)$.

9.5 Input Output Hidden Markov Models

Input Output Hidden Markov Models (IOHMMs) generate an output sequence $x^T = (x_1, x_2, x_3, \ldots, x_T)$ of length T conditioned on an input sequence $y^T = (y_1, y_2, y_3, \ldots, y_T)$ of length T.

The difference between standard HMMs and input output HMMs is that the probabilities are conditioned on the input. Start probabilities are $p_S(u_1 \mid y_1)$, the transition probabilities $p(u_t \mid y_t, u_{t-1})$, and the emission probabilities $p_E(x_t \mid y_t, u_t)$.

Using IOMMs also negative examples can be used by setting for all y_t either a don't care or a fixed value and setting $y_T = 1$ for the positive class and $y_T = -1$ for the negative class. Whether a model for the negative class can be built is not clear but at least a subclass of the negative class which is very similar to the positive class can be better discriminated.

The number of parameters increase proportional to the number of input symbols, which may make it more difficult to estimate the probabilities if not enough data is available.

Learning via the likelihood is as with the standard HMM with the probabilities additionally conditioned on the input.

9.6 Factorial Hidden Markov Models

The HMM architecture Fig. 9.6 is extended to Fig. 9.9 where the hidden state is divided into more components u_i (three in the figure).

The transition probability of u_i is conditioned on all u_k with $k \leq i$ and the emission probability depends on all hidden states. In the HMM architecture in Fig. 9.9 u_1 evolves very slowly, u_2 evolves faster, and u_3 evolves fastest of all hidden variables. Fast evolving variables do not influence slow evolving ones but slow evolving variables influence fast evolving variables.

If the factorial HMM has h hidden state variables u_i and each one of them can take on S values then the emission probability distribution consists of $P\ S^h$ emission probabilities. Therefore

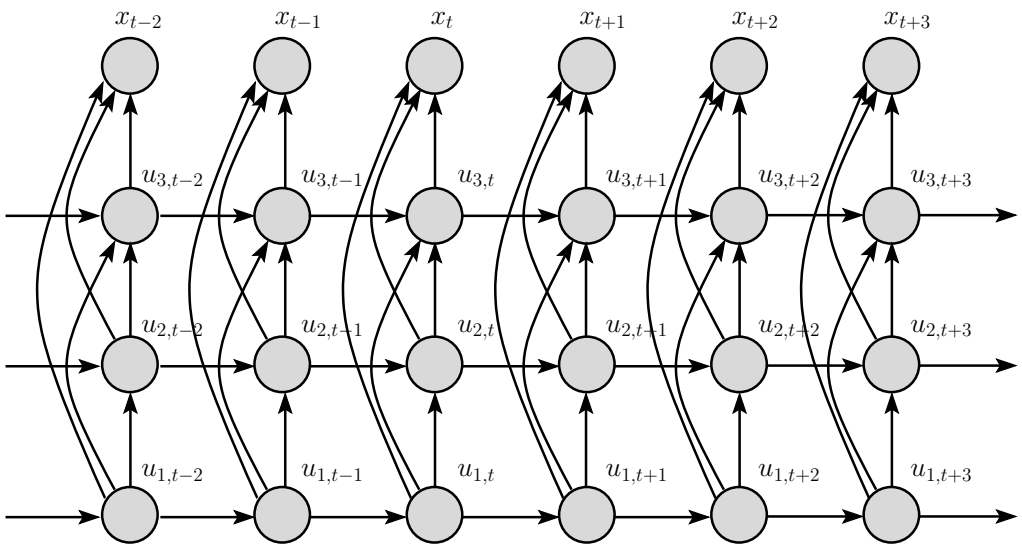


Figure 9.9: A factorial HMM with three hidden state variables u_1, u_2, and u_3. The transition probability of u_i is conditioned on all u_k with $k \leq i$ and the emission probability depends on all hidden states.

learning factorial HMMs is computational expensive. However approximative methods have been developed to speed up learning [Ghahramani and Jordan, 1996, 1997].

9.7 Memory Input Output Factorial Hidden Markov Models

Remember that we quoted Sean Eddy [Eddy, 2004]:

> "HMMs are reasonable models of linear sequence problems, but they don't deal well with correlations between residues or states, especially long-range correlations. HMMs assume that each residue depends only on one underlying state, and each state in the state path depends only on one previous state; otherwise, residues and states are independent of each other." ... "The state path of an HMM has no way of remembering what a distant state generated when a second state generates its residue."

The only way a HMM can store information over time is to go into a certain state value and don't change it any more. The state is fixed and the event which led the HMM enter the fixed state is memorized. Instead of a state a set of states can be entered from which the escape probability is zero.

To realize a state with a non-escaping value which can memorize past events is

$$p(u_t = a \mid u_{t-1} = a) \; = \; 1 \; .$$

That means if the state takes on the value a then the state will not take any other value.

In principle the storage of past events can be learned but the likelihood of storing decreases exponentially with the time of storage. Therefore learning to store is practically impossible because

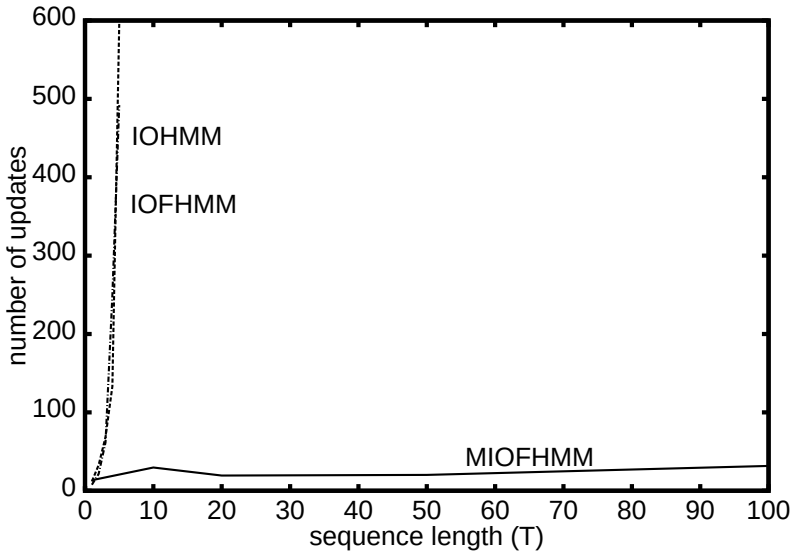


Figure 9.10: Number of updates required to learn to remember an input element until sequence end for three models: input output HMM (IOHMM), input output factorial HMM (IOFHMM), and "Memory-based Input-Output Factorial HMM" (MIOFHMM) as a function of the sequence length T.

these small likelihood differences are tiny in comparison to local minima resulting from certain input / output patterns or input / output distributions.

Therefore memory is enforced by setting $p(u_t = a \mid u_{t-1} = a) = 1$ and not allowing this probability to change.

However after the storage process (taking on the value a) the model is fixed and neither future systems dynamics nor other events to memorize can be dealt with.

To overcome this problem a factorial HMM can be used where some of the hidden state variables can store information and others extract the dynamics of the system to model.

Storing events is especially suited for input output HMMs where input events can be stored.

An architecture with memory state variable and using the input output architecture is the "Memory-based Input-Output Factorial HMM" (MIOFHMM, [Hochreiter and Mozer, 2001c]).

Initially, all state variables have "uncommitted" values then various inputs can trigger the memory state variables to take on values from which the state variables cannot escape – they behave as a memory for the occurrence of an input event. Fig. 9.10 shows the number of updates required to train three models: input output HMM (IOHMM), input output factorial HMM (IOFHMM), and "Memory-based Input-Output Factorial HMM" (MIOFHMM) as a function of the sequence length T.

9.8 Tricks of the Trade

- Sometimes the HMM and its algorithms must be adjusted for bioinformatics applications for example to handle delete states which do not emit symbols in the forward pass.
- HMMs can be used for variable length of the sequences; however care must be take if comparing likelihoods because there are more longer sequences than shorter and the likelihood decreases exponentially with the length
- To deal with small likelihood and probability values is is recommended to compute the values in the log-space
- To avoid zero probabilities for certain sequences which makes certain minima unreachable all probabilities can be kept above a threshold ϵ.
- The EM-algorithm cannot reach probabilities which are exact zero, therefore, as an after-learning postprocessing all small probabilities $\leq \epsilon$ can be set to zero. This often helps to generalize from short to very long sequences.
- HMMs are prone to local minima, for example if HMMs are build from unaligned sequences. Global optimization strategies try to avoid theses minima, e.g. deterministic annealing was suggested in the "Userguide" to HMMER version 1.8.

9.9 Profile Hidden Markov Models

Profile Hidden Markov Models code a multiple sequence alignment into an HMM as a position-specific scoring system which can be used to search databases for remote homologous sequences. Fig. 9.11 shows a HMM which can be used for homology search. The top row with states indicated with circles are a pattern. The diamond states are inserted strings. The bottom row with states indicated as squares are deletions, where a letter from the pattern is skipped.

To learn an HMM from a set of unaligned positive examples suffers from the problem of local minima. Therefore expensive global optimization strategies must be used to avoid theses minima, e.g. deterministic annealing was suggested in the "Userguide" to HMMER version 1.8. Therefore in most applications an HMM is at least initialized by a multiple alignment of the positive examples.

The use of profile HMMs was made very convenient by the free HMMER package by Sean Eddy [Eddy, 1998] which allows to build and apply HMMs. HMMER supplies a log-odds likelihood of the model compared to a random model to access the significance of the score of a new sequence. Fig. 9.12 shows the architecture of the models used by HMMER. The states indicated by squares and denoted by "Mx" are the consensus string. The circled states denoted by "Dx" are deletion states (non-emitting states), where part of the consensus string can be skipped. The diamond states denoted by "Ix" are insertion states where a substring can be inserted into the consensus string

The other package which enabled a convenient use of HMMs for biological sequences is Sequence Alignment and Modeling system (SAM – `http://www.cse.ucsc.edu/research/compbio/sam.html`) which allows creating, refining, and using HMMs for biological sequence

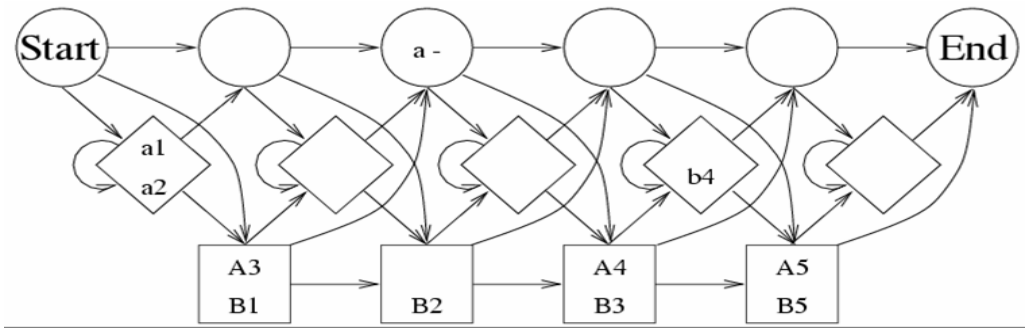


Figure 9.11: Hidden Markov model for homology search. The top row with states indicated with circles are a pattern. The diamond states are inserted strings. The bottom row with states indicated as squares are deletions, where a letter from the pattern is skipped.

analysis. Also the SAM models represent a refinement of a multiple alignment. Models can be used to both generate multiple alignments and search databases for new members of the family.

Also databases like Protein FAMily database (Pfam) are based on HMMs. 67% of proteins contain at least one Pfam profile HMM and 45% of residues in the protein database are covered in total by the HMMs.

Another HMM application which is not associated with profile HMMs is shown in Fig. 9.13, where the HMM is used for splice site detection.

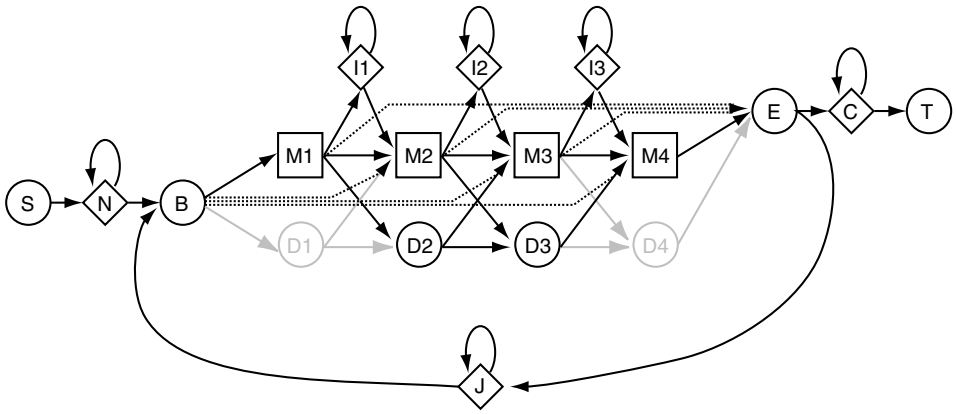


Figure 9.12: The HMMER hidden Markov architecture. The states indicated by squares and denoted by "Mx" form a pattern (consensus string). The circled states denoted by "Dx" are deletion states (non-emitting), where a letter from the pattern can be skipped. The diamond states denoted by "Ix" are insertion states where a substring between letters of the pattern has been inserted. "B" and "E" denote the begin and end state of the pattern, respectively.

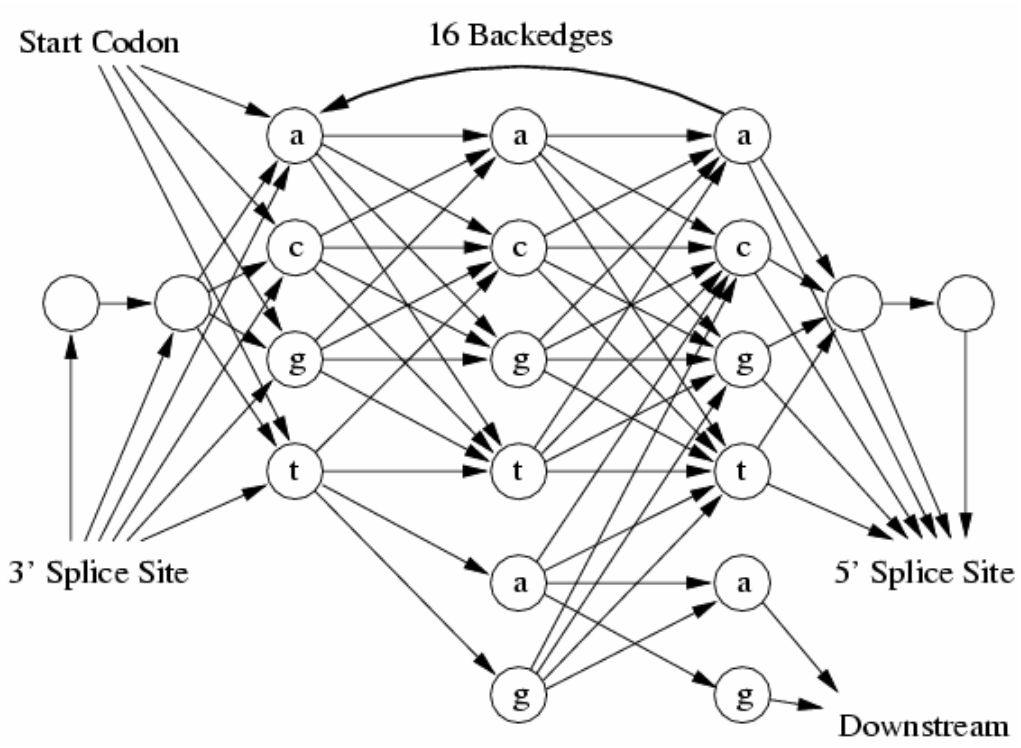


Figure 9.13: An HMM for splice site detection.

Chapter 10

Boltzmann Machines

10.1 The Boltzmann Machine

A *Boltzmann machine* is a stochastic recurrent neural network proposed by Geoffrey Hinton and Terry Sejnowski in 1985. Boltzmann machines with unconstrained connectivity have not proven useful for practical problems in machine learning or inference. However recently *restricted Boltzmann machines* gained high popularity in the context of *deep learning*. The name stems from the Boltzmann distribution in statistical mechanics, which is the normalizing distribution and used for sampling.

A Boltzmann machine is a network of units with an "energy" associated with its current state (see Fig. 10.1). It consists of binary units that are stochastic. The global energy, E, is

$$E \;=\; -\left(\sum_{i,j;i<j} w_{ij}\, s_i\, s_j + \sum_i \theta_i\, s_i\right) . \tag{10.1}$$

Where:

- w_{ij} is the connection strength between unit j and unit i.
- s_i is the state, $s_i \in \{0, 1\}$, of unit i.
- θ_i is the bias or the activation threshold of unit i.

Restriction on the connections of the Boltzmann machine are:

- $w_{ii} = 0 \qquad \forall i$, that is, units do not have self-connections,
- $w_{ij} = w_{ji} \qquad \forall i, j$, that is, connections are symmetric.

Typically the weights are written by a symmetric matrix $\boldsymbol{W}$, with zeros along the diagonal ($\boldsymbol{W}$ is indefinite).

The energy difference of a single unit i being 0 (off) versus 1 (on) is ΔE_i:

$$\Delta E_i \;=\; \sum_j w_{ij}\, s_j \;+\; \theta_i\, . \tag{10.2}$$

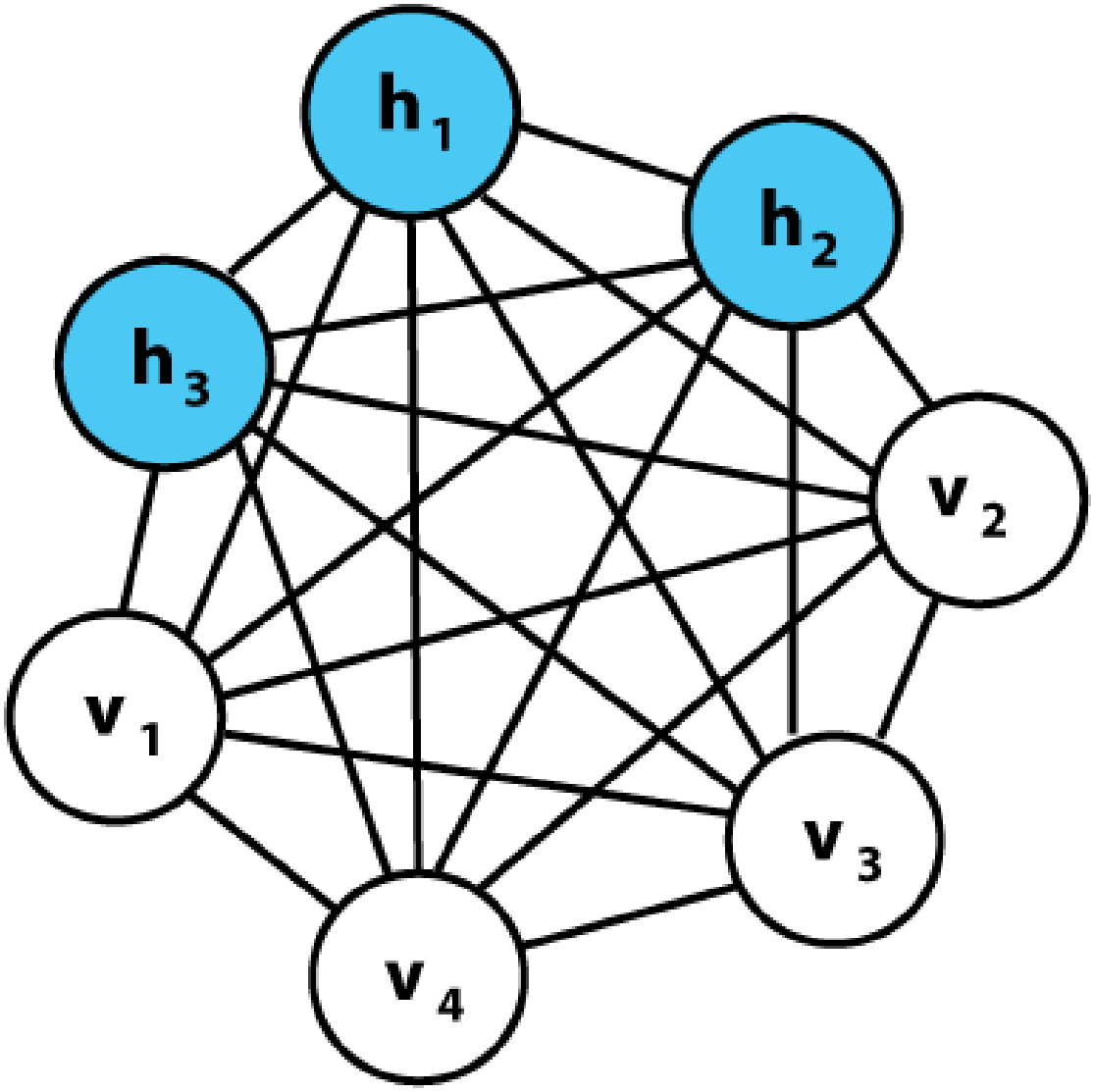


Figure 10.1: A graphical representation of an example Boltzmann machine. Each undirected edge represents dependency. In this example there are 3 hidden units and 4 visible units.

For the Boltzmann distribution the energy of a state is proportional to the negative log probability of that state. We can compute the Boltzmann Factor which is the energy difference if one unit is flipped:

$$\Delta E_i \;=\; -k_B\,T\ln(p_{\text{i=off}}) \;-\; (-k_B\,T\ln(p_{\text{i=on}})) \tag{10.3}$$

where k_B is Boltzmann's constant and is absorbed into the temperature T.

We obtain for the energy difference at temperature T:

$$\begin{aligned}
& \frac{\Delta E_i}{T} \;=\; \ln(p_{\text{i=on}}) \;-\; \ln(p_{\text{i=off}}) \qquad (10.4)\\
\Leftrightarrow\quad & \frac{\Delta E_i}{T} \;=\; \ln(p_{\text{i=on}}) \;-\; \ln(1-p_{\text{i=on}})\\
\Leftrightarrow\quad & \frac{\Delta E_i}{T} \;=\; \ln\left(\frac{p_{\text{i=on}}}{1-p_{\text{i=on}}}\right)\\
\Leftrightarrow\quad & -\frac{\Delta E_i}{T} \;=\; \ln\left(\frac{1-p_{\text{i=on}}}{p_{\text{i=on}}}\right)\\
\Leftrightarrow\quad & -\frac{\Delta E_i}{T} \;=\; \ln\left(\frac{1}{p_{\text{i=on}}}-1\right)\\
\Leftrightarrow\quad & \exp\left(-\frac{\Delta E_i}{T}\right) \;=\; \frac{1}{p_{\text{i=on}}} \;-\; 1\,.
\end{aligned}$$

Solving for $p_{\text{i=on}}$ gives the probability that the i-th unit is on:

$$p_{\text{i=on}} \;=\; \frac{1}{1+\exp(-\frac{\Delta E_i}{T})} \tag{10.5}$$

where T is the temperature of the system. Therefore the logistic function is used in activation probabilities of the Boltzmann machine.

The network is run iteratively choosing a unit and setting its state according to its probability from above. When the machine is "at thermal equilibrium" the probability distribution of global states has converged. Often the network starts with a high temperature which gradually decreases until a thermal equilibrium is reached with a low temperature. The energy level may fluctuate around a global minimum.

10.2 Learning in the Boltzmann Machine

The goal of training the network is to ensure that it will converge to a global state given an external distribution. Hence, training searches for weights so that the global states with the highest probabilities will get the lowest energies.

The Boltzmann Machine consists of "visible" units, V, and "hidden" units, H. The visible units are those, which receive input from the "environment", thus the training set is a set of binary vectors over V. The distribution over the training set is $P^+(V)$. The distribution over global states converges as the Boltzmann machine reaches thermal equilibrium. After marginalization over the hidden units, this distribution is $P^-(V)$.

The aim in Boltzmann Machine learning is to approximate the "true" distribution $P^+(V)$ using the model distribution $P^-(V)$. The Kullback-Leibler divergence measures the difference between these distributions:

$$D_{\mathrm{KL}}(P^+ \parallel P^-) \;=\; \sum_v P^+(v)\;\ln\left(\frac{P^+(v)}{P^-(v)}\right)\;, \tag{10.6}$$

where the sum is over all the possible states of V. D is a function of the weights, therefore this objective can be minimized by gradient descent algorithms.

Analog to the EM algorithm, Boltzmann machine training has two phases that are performed alternating. During the "positive" phase the visible units' states are clamped to the training data. During the "negative" phase the network runs freely, i.e. the training data does not influence the network.

The gradient of the objective D with respect to a given weight, w_{ij}, is given by:

$$\frac{\partial D}{\partial w_{ij}} \;=\; -\,\frac{1}{\eta}\,\left(p_{ij}^{+} - p_{ij}^{-}\right)\;, \tag{10.7}$$

where:

- p_{ij}^{+} is the probability of units i and j both being on when the machine is at equilibrium on the positive phase,
- p_{ij}^{-} is the probability of units i and j both being on when the machine is at equilibrium on the negative phase,
- η denotes the learning rate.

Interestingly only information needed to change the weights is provided by "local" information which makes the Boltzmann Machine a biological plausible model.

The training the bias weights is:

$$\frac{\partial D}{\partial \theta_i} \;=\; -\frac{1}{\eta}\left(p_i^{+} - p_i^{-}\right)\;. \tag{10.8}$$

Problems with the Boltzmann machine:

- the time the machine must be run in order to collect equilibrium statistics grows exponentially with the machine's size, and with the magnitude of the connection strengths
- connection strengths are more plastic when the units being connected have activation probabilities intermediate between zero and one, leading to a so-called variance trap. The net effect is that noise causes the connection strengths to random walk until the activities saturate.

10.3 The Restricted Boltzmann Machine

A quite efficient architecture is the "restricted Boltzmann machine" (RBM) which does not have intralayer connections between hidden or visible units (see Fig. 10.2). After training one RBM, the activities of its hidden units can be viewed as representation of the visible units. This idea is used in deep learning where the hidden units of one RBM are the visible units of a higher-level RBM. Therefore it is possible to stack RBMs and to train many layers of hidden units efficiently. This is a common unsupervised deep learning strategy.

The weight $w_{i,j}$ is the connection between hidden unit h_j and visible unit v_i and the bias weights (offsets) are a_i for the visible units and b_j for the hidden units. The energy can be computed as

$$E(\boldsymbol{v}, \boldsymbol{h}) \;=\; -\sum_i a_i\, v_i \;-\; \sum_j b_j\, h_j \;-\; \sum_{i,j} h_j\, v_i\, w_{i,j}\;. \tag{10.9}$$

or, in matrix form,

$$E(\boldsymbol{v}, \boldsymbol{h}) \;=\; -\,\boldsymbol{a}^T\boldsymbol{v} \;-\; \boldsymbol{b}^T\boldsymbol{h} \;-\; \boldsymbol{h}^T\boldsymbol{W}\boldsymbol{v}\;. \tag{10.10}$$

Probability distributions over hidden and visible vectors are defined as:

$$p(\boldsymbol{v}, \boldsymbol{h}) \;=\; \frac{1}{Z}\, e^{-E(\boldsymbol{v},\boldsymbol{h})}\;, \tag{10.11}$$

where Z is a normalizing constant or the partition function defined as the sum of $e^{-E(\boldsymbol{v},\boldsymbol{h})}$ over all possible configurations. Marginalizing gives the probability of a visible (input) vector of booleans:

$$p(\boldsymbol{v}) \;=\; \frac{1}{Z}\, \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \tag{10.12}$$

Since the RBM is a bipartite graph, the hidden unit activations are mutually independent given the visible unit activations and conversely:

$$p(\boldsymbol{v} \mid \boldsymbol{h}) \;=\; \prod_{i=1}^{m} p(v_i \mid \boldsymbol{h}) \tag{10.13}$$

$$p(\boldsymbol{h} \mid \boldsymbol{v}) \;=\; \prod_{j=1}^{n} p(h_j \mid \boldsymbol{v}) \tag{10.14}$$

with the individual activation probabilities:

$$p(h_j = 1 \mid \boldsymbol{v}) \;=\; \sigma(b_j + \sum_{i=1}^{m} w_{i,j}\, v_i) \tag{10.15}$$

$$p(v_i = 1 \mid \boldsymbol{h}) \;=\; \sigma(a_i + \sum_{j=1}^{n} w_{i,j}\, h_j)\;, \tag{10.16}$$

where σ is the sigmoid function.

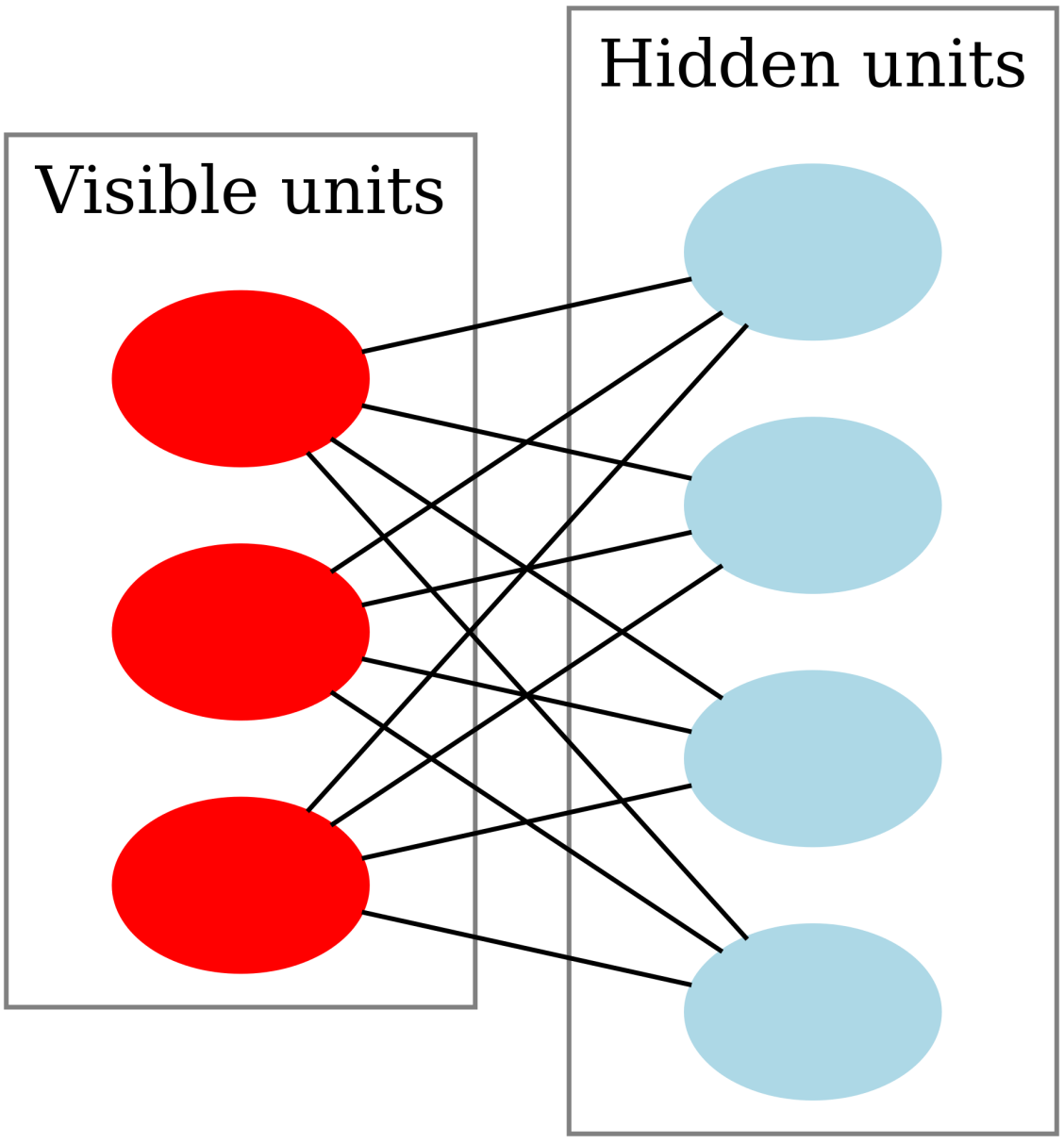


Figure 10.2: Graphical representation of a restricted Boltzmann machine. The four blue units represent hidden units, and the three red units represent visible states. In restricted Boltzmann machines there are only connections (dependencies) between hidden and visible units, and none between units of the same type.

Training aims at maximizing the product of probabilities assigned to some training set V:

$$\arg\max_{\boldsymbol{W}} \prod_{\boldsymbol{v} \in V} p(\boldsymbol{v}) \tag{10.17}$$

or equivalently, to maximize the expected log probability

$$\arg\max_{\boldsymbol{W}} \mathrm{E}\left(\sum_{\boldsymbol{v} \in V} \log p(\boldsymbol{v})\right) \tag{10.18}$$

The algorithm most often used to train RBMs, that is, to optimize the weight vector $\boldsymbol{W}$, is the contrastive divergence (CD) algorithm. The single-step contrastive divergence procedure for a single sample is:

1. Take a training sample $\boldsymbol{v}$, compute the probabilities of the hidden units and sample a hidden activation vector $\boldsymbol{h}$ from this probability distribution.
2. Compute the outer product of $\boldsymbol{v}$ and $\boldsymbol{h}$ and call this the "positive gradient".
3. From $\boldsymbol{h}$, sample a reconstruction $\boldsymbol{v}'$ of the visible units, then resample the hidden activations $\boldsymbol{h}'$ from this.
4. Compute the outer product of $\boldsymbol{v}'$ and $\boldsymbol{h}'$ and call this the "negative gradient".
5. Let the weight update to $w_{i,j}$ be the "positive gradient" minus the "negative gradient", times some learning rate η.

Bibliography

E. Anderson. The irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935.

H. Attias. Independent Factor Analysis. *Neural Computation*, 11(4):803–851, 1999.

C. Barrett, R. Hughey, and K. Karplus. Scoring hidden marov models. *CABIOS*, 13(2):191–19, 1997.

O. A. Bashkirov, E. M. Braverman, and I. B. Muchnik. Potential function algorithms for pattern recognition learning machines. *Automation and Remote Control*, 25:629–631, 1964.

A. Bateman, E. Birney, R. Durbin, S. R. Eddy, K. L. Howe, and E. L. L. Sonnhammer. The Pfam protein families database. *Nucleic Acids Res.*, 28:263–266, 2000.

A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. L. Sonnhammer, D. J. Studholme, C. Yeats, and S. R. Eddy. The pfam protein families database. *Nucleic Acids Research*, 32:D138–141, 2004.

A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. *J. Comput. Biol.*, 10(3-4):373–384, 2003.

C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

P. S. Bithas, N. C. Sagias, T. A. Tsiftsis, and G. K. Karagiannidis. Distributions involving correlated generalized gamma variables. In *Proc. Int. Conf. on Applied Stochastic Models and Data Analysis*, volume 12, Chania, 2007.

U. Bodenhofer, A. Kothmeier, and S. Hochreiter. APCluster: an R package for affinity propagation clustering. *Bioinformatics*, 27(17):2463–2464, 2011.

C. Burge and S. Karlin. Prediction of complete gene structures in human genomic dna. *J. Mol. Biol.*, 268:78–94, 1997.

S. Busygin, G. Jacobsen, and E. Kramer. Double conjugated clustering applied to leukemia microarray data. In *Proc. 2nd SIAM Int. Conf. on Data Mining / Workshop on Clustering High Dimensional Data*, 2002.

J. Caldas and S. Kaski. Bayesian biclustering with the plaid model. In *Proc. IEEE Int. Workshop on Machine Learning for Signal Processing*, volume XVIII, pages 291–296, 2008.

A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarays for phenotype classification. In *Proc. Int. Conf. on Computational Molecular Biology*, pages 75–85, 2000.

J.-F. Cardoso. Infomax and maximum likelihood for source separation. *IEEE Signal Processing Letters*, 4:112–114, 1997.

J.-F. Cardoso and B. Laheld. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing*, 44:3017–3030, 1996.

J.-F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings-F*, 140(6):362–370, 1993.

Y. Cheng and G. M. Church. Biclustering of expression data. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology*, volume 8, pages 93–103, 2000.

A. Cichocki, R. Unbehauen, L. Moszczynski, and E. Rummert. A new on-line adaptive algorithm for blind separation of source signals. In *Proc. Int. Symposium on Artificial Neural Networks, ISANN-94*, pages 406–411, 1994.

D.-A. Clevert, A. Mitterecker, A. Mayr, G. Klambauer, M. Tuefferd, A. DeBondt, W. Talloen, H. Göhlmann, and S. Hochreiter. cn.FARMS: a latent variable model to detect copy number variations in microarray data with a low false discovery rate. *Nucleic Acids Res.*, 39(12):e79, 2011.

P. Comon. Independent component analysis – a new concept? *Signal Processing*, 36(3):287–314, 1994.

R. Condit, N. Pitman, E. G. Leigh, J. Chave, J. Terborgh, R. B. Foster, P. Nunez, S. Aguilar, R. Valencia, G. Villa, H. C. Muller-Landau, E. Losos, and S. P. Hubbell. Beta-diversity in tropical forest trees. *Science*, 295:666–669, 2002.

M. Cottrell, J. C. Fort, and G. Pages. Comments about "Analysis of the convergence properties of topology preserving neural networks". *IEEE Transactions on Neural Networks*, 6(3):797–799, 1995.

G. Darmois. Analyse générale des liaisons stochastiques. *Rev. Inst. Internat. Stat.*, 21:2–8, 195.

G. Deco and W. Brauer. Higher order statistical decorrelation without information loss. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 247–254. The MIT Press, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–22, 1977.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2nd edition, 2001.

S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14:755–763, 1998.

S. R. Eddy. What is a hidden markov model? *Nature Biotechnology*, 22:1315–1316, 2004.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *P. Natl. Acad. Sci. USA*, 95(25):14863–14868, 1998.

E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67:47–55, 1992.

B. S. Everitt. *An introduction to latent variable models*. Chapman and Hall, London, 1984.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 (Part II):179–188, 1936.

G. W. Flake and B. A. Pearlmutter. Differentiating functions of the Jacobian with respect to the weights. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 435–441. The MIT Press, Cambridge, MA, London, England, 2000.

P. Földiák. Forming sparse representations by local anti-hebbian learning. *Biological Cybernetics*, 64:165–170, 1990.

E. W. Forgy. Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, 21:768–769, 1965.

B. J. Frey and D. Dueck. Mixture modeling by affinity propagation. *Neural Information Processing Systems*, 18:379–386, 2006.

B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315: 972–976, 2007.

J. H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397):249–266, 1987.

J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, 1981.

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23(9):881–890, 1974.

X. Gan, A. W.-C. Liew, and H. Yan. Discovering biclusters in gene expression data based on high-dimensional linear geometries. *BMC Bioinformatics*, 9(209), 2008.

J. P. Garay and B. H. Park. Androgen receptor as a targeted therapy for breast cancer. *Am. J. Cancer Res.*, 2(4):434–445, 2012.

L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *J. Mach. Learn. Res.*, 3:679–707, 2002.

G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *P. Natl. Acad. Sci. USA*, 97(22):12079–12084, 2000.

Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 472–478. The MIT Press, 1996.

Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29:245, 1997.

M. Girolami. A variational method for learning sparse and overcomplete representations. *Neural Comput.*, 13(11):2517–2532, 2001.

I. E. Givoni and B. J. Frey. A binary variable model for affinity propagation. *Neural Computation*, 21(6):1589–1600, 2009.

J. C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53:325–328, 1966.

J. Gu and J. S. Liu. Bayesian biclustering of gene expression data. *BMC Genomics*, 9(Suppl. 1): S4, 2008.

J. A. Hartigan. Direct clustering of a data matrix. *J. Am. Stat. Assoc.*, 67(337):123–129, 1972.

J. A. Hartigan. *Clustering Algorithms*. Wiley, New York, 1975.

J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

V. Hasselblad. Estimation of parameters for a mixture of normal distributions. *Technometrics*, 8: 431–444, 1966.

T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84: 502–516, 1989.

G. Hinton and T. J. Sejnowski. Introduction. In G. Hinton and T. J. Sejnowski, editors, *Unsupervised Learning: Foundations of Neural Computation*, pages VII–XVI. MIT Press, Cambridge, MA, London, England, 1999.

G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society* **B**, 352:1177–1190, 1997.

G. E. Hinton and T. E. Sejnowski. Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing*, volume 1, pages 282–317. MIT Press, 1986.

S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. VanSanden, D. Lin, W. Talloen, L. Bijnens, H. W. H. Göhlmann, Z. Shkedy, and D.-A. Clevert. FABIA: factor analysis for bicluster acquisition. *Bioinformatics*, 26(12):1520–1527, 2010.

S. Hochreiter, D.-A. Clevert, and K. Obermayer. A new summarization method for Affymetrix probe level data. *Bioinformatics*, 22(8):943–949, 2006.

S. Hochreiter and M. C. Mozer. An electric field approach to independent component analysis. In P. Pajunen and J. Karhunen, editors, *Proceedings of the Second International Workshop on Independent Component Analysis and Blind Signal Separation, Helsinki, Finland*, pages 45–50. Otamedia, Espoo, Finland, ISBN: 951-22-5017-9, 2000.

S. Hochreiter and M. C. Mozer. Beyond maximum likelihood and density estimation: A sample-based criterion for unsupervised learning of complex models. In T. K. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, 2001a.

S. Hochreiter and M. C. Mozer. Coulomb classifiers: Reinterpreting SVMs as electrostatic systems. Technical Report CU-CS-921-01, Department of Computer Science, University of Colorado, Boulder, 2001b.

S. Hochreiter and M. C. Mozer. A discrete probabilistic memory model for discovering dependencies in time. In G. Dorffner, H. Bischof, and K. Hornik, editors, *International Conference on Artificial Neural Networks*, pages 661–668. Springer, 2001c.

S. Hochreiter, M. C. Mozer, and K. Obermayer. Coulomb classifiers: Generalizing support vector machines via an analogy to electrostatic systems. In S. Beckers, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 545–552. MIT Press, Cambridge, MA, 2003.

S. Hochreiter and K. Obermayer. Gene selection for microarray data. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel Methods in Computational Biology*, pages 319–355. MIT Press, 2004.

S. Hochreiter and K. Obermayer. Nonlinear feature selection with the potential support vector machine. In I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature extraction, Foundations and Applications*. Springer, 2005.

S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997a.

S. Hochreiter and J. Schmidhuber. Low-complexity coding and decoding. In K. M. Wong, I. King, and D. Yeung, editors, *Theoretical Aspects of Neural Computation (TANC 97), Hong Kong*, pages 297–306. Springer, 1997b.

S. Hochreiter and J. Schmidhuber. Lococode versus PCA and ICA. In L. Niklasson, M. Boden, and T. Ziemke, editors, *Proceedings of the International Conference on Artificial Neural Networks, Skövde, Sweden*, pages 669–674. Springer, 1998.

S. Hochreiter and J. Schmidhuber. Feature extraction through LOCOCODE. *Neural Computation*, 11(3):679–714, 1999a.

S. Hochreiter and J. Schmidhuber. LOCOCODE performs nonlinear ICA without knowing the number of sources. In J.-F. Cardoso, C. Jutten, and P. Loubaton, editors, *Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation, Aussois, France*, pages 149–154, 1999b.

S. Hochreiter and J. Schmidhuber. LOCOCODE performs nonlinear ICA without knowing the number of sources. In J.-F. Cardoso, C. Jutten, and P. Loubaton, editors, *Proceedings of the First International Workshop on Independent Component Analysis and Signal Separation, Aussois, France*, pages 149–154, 1999c.

S. Hochreiter and J. Schmidhuber. Nonlinear ICA through low-complexity autoencoders. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS'99)*, volume 5, pages 53–56. IEEE, 1999d.

S. Hochreiter and J. Schmidhuber. Nonlinear ICA through low-complexity autoencoders. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS'99)*, volume 5, pages 53–56. IEEE, 1999e.

S. Hochreiter and J. Schmidhuber. Source separation as a by-product of regularization. In M. S. Kearns, S. A. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 459–465. MIT Press, Cambridge, MA, 1999f.

S. Hochreiter and J. Schmidhuber. Source separation as a by-product of regularization. In M. S. Kearns, S. A. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 459–465. MIT Press, Cambridge, MA, 1999g.

T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 668–693, 1999.

T. Hofmann, J. Puzicha, and M. I. Jordan. Learning from dyadic data. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 466–472, 1999.

Y. Hoshida, J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov. Subclass mapping: Identifying common subtypes in independent disease data sets. *PLoS ONE*, 2(11):e1195, 2007.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.*, 5:1457–1469, 2004.

P. J. Huber. Projection pursuit. *Annals of Statistics*, 13(2):435–475, 1985.

A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

A. Hyvärinen, J. Hurri, and P. O. Hoyer. *Natural Image Statistics — A probabilistic approach to early computational vision*. ISBN 9781848824911. Springer, Dordrecht, Heidelberg, London, New York, 2009.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Adaptive and Learning Systems for Signal Processing, Communications, and Control, S. Haykin, editor. John Wiley & Sons, New York, 2001.

A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1999.

A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

J. Ihmels, S. Bergmann, and N. Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13):1993–2003, 2004.

N. Intrator. Combining exploratory projection pursuit and projection pursuit regression with application to neural networks. *Neural Computation*, 5(3):443–457, 1993.

J. E. Jackson. *A User's Guide to Principal Components*. Wiley, New York, 1991.

I. T. Jolliffe. *Principal Component Analysis*. Springer, New York, 1986.

L. K. Jones. On a conjecture of Huber concerning the convergence of projection pursuit regression. *Annals of Statistics*, 15:880–882, 1987.

M. C. Jones and R. Sibson. What is projection pursuit? *Journal of the Royal Statistical Society, series A*, 150:1–36, 1987.

K. G. Jöreskog. Some contributions to maximum likelihood factor analysis. *Psychometrika*, 32: 443–482, 1967.

K. G. Jöreskog. Some contributions to maximum likelihood factor analysis. *Psychometrika*, 32: 443–482, 1967.

C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10, 1991.

A. M. Kagan, Y.V. Linnik, and C.R. Rao. *Characterization Problems in Mathematical Statistics*. Wiley, New York, 1973.

K. Karplus, C. Barrett, M. Cline, M. Diekhans, L. Grate, and R. Hughey. Predicting protein structure using only sequence information. *Proteins: Structure, Function, and Genetics*, 37 (S3):121–125, 1999.

K. Karplus, C. Barrett, and R. Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

A. Kasim, D. Lin, S. VanSanden, D.-A. Clevert, L. Bijnens, H. Göhlmann, D. Amaratunga, S. Hochreiter, Z. Shkedy, and W. Talloen. Informative or noninformative calls for gene expression: A latent variable approach. *Statistical Applications in Genetics and Molecular Biolog*, 9 (1):1544–6115, 2010.

G. Klambauer, K. Schwarzbauer, A. Mayr, D.-A. Clevert, A. Mitterecker, U. Bodenhofer, and S. Hochreiter. cn.MOPS: mixture of Poissons for discovering copy number variations in next-generation sequencing data with a low false discovery rate. *Nucleic Acids Res.*, 40(9):e69, 2012.

G. Klambauer, T. Unterthiner, and S. Hochreiter. DEXUS: Identifying differential expression in RNA-Seq studies with unknown conditions. *Nucleic Acids Res.*, 2013.

Y. Kluger, R. Basri, J. T. Chang, and M. B. Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Res.*, 13:703–716, 2003.

T. Kohonen. Analysis of a simple self-organizing process. *Biological Cybernetics*, 43:135–140, 1982.

T. Kohonen. *Self-Organization and Associative Memory*. Springer, second ed., 1988.

T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480, 1990.

T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1995a.

T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1995b.

A. Krogh. Two methods for improving performance of a hmm and their application for gene finding. In T. Gaasterland, P. Karp, K. Karplus, C. Ouzounis, C. Sander, and A. Valencia, editors, *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, pages 179–186. AAAI Press, Menlo Park, CA, 1997.

A. Krogh, M. Brown, I. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994a.

A. Krogh, I. S. Mian, and D. Haussler. A hidden Markov model that finds genes in E. coli DNA. *Nucl. Acids Res.*, 22:4768–4778, 1994b.

D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden markov model for the recognition of human genes in dna. In D. J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R. F. Smith, editors, *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 134–142. AAAI Press, Menlo Park, CA, 1996.

L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Stat. Sinica*, 12(1):61–86, 2002.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562, 2001.

M. S. Lewicki and B. A. Olshausen. Inferring sparse, overcomplete image codes using an efficient coding framework. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 815–821. MIT Press, Cambridge, MA, 1998.

M. S. Lewicki and T. J. Sejnowski. Learning nonlinear overcomplete representations for efficient coding. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 556–562. MIT Press, Cambridge, MA, 1998.

R.H. Lilien, H. Farid, and B. R. Donald. Probabilistic disease classification of expression-dependent proteomic data from mass spectrometry of human serum. *Computational Biology*, 10(6), 2003.

P. Lio, J. L. Thorne, N. Goldman, and D. T. Jones. Passml: Combining evolutionary inference and protein secondary structure prediction. *Bioinformatics*, 14:726–73, 1999.

H. Lütkepohl. *Handbook of Matrices*. John Wiley & Sons, 1996.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. LeCam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE ACM T. Comput. Bi.*, 1(2):24–45, 2004.

S. C. Madeira and A. L. Oliveira. A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series. *Algorithm. Mol. Biol.*, 4:8+, 2009.

S. C. Madeira, M. C. Teixeira, I. Sa-Correia, and A. L. Oliveira. Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm. *IEEE ACM T. Comput. Bi.*, 7(1):153–165, 2010.

E. Moulines, J.-F. Cardoso, and E. Gassiat. Maximum-likelihood for blind separation and deconvolution of noisy signals using mixture models. In *Proceedings of the IEEE International Conference non Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 3617–3620, 1997.

F. Mulier and V. Cherkassky. Self-organization as an iterative kernel smoothing process. *Neural Computation*, 7(6):1165–1177, 1995.

T. M. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pac. Symp. Biocomputing*, pages 77–88, 2003.

R. M. Neal and P. Dayan. Factor analysis using delta-rule wake-sleep learning. *Neural Computation*, 9(8):1781–1803, 1997.

K. Obermayer, G. G. Blasdel, and K. Schulten. A statistical mechanical analysis of self-organization and pattern formation during the development of visual maps. *Physical Review A*, 45:7568–7589, 1992.

E. Oja. A simplified neuron model as principal component analyser. *Journal of Mathematical Biology*, 15:267–273, 1982.

E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1(1):61–68, 1989.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

P. Paatero and U. Tapper. Least squares formulation of robust non-negative factor analysis. *Chemometr. Intell. Lab.*, 37:23–35, 1997.

J. Palmer, D. Wipf, K. Kreutz-Delgado, and B. Rao. Variational EM algorithms for non-Gaussian latent variable models. In *Advances in Neural Information Processing Systems 18*, pages 1059–1066, 2006.

J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *Journal of Molecular Biology*, 284(4):1201–1210, 1998.

B. A. Pearlmutter and L. C. Parra. Maximum likelihood blind source separation: A context-sensitive generalization of ICA. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 613–619. MIT Press, Cambridge MA, 1997.

K. Pearson. Contribution to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society A*, 185:71–110, 1894.

M. P. Perrone and L. N. Cooper. Coulomb potential learning. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 272–275, Cambridge, MA, 1995. The MIT Press.

A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

C. R. Rao. *Linear Statistical Inference and its Applications*. Wiley, New York, 1973.

D. J. Reiss, N. S. Baliga, and R. Bonneau. Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC Bioinformatics*, 2(7):280–302, 2006.

H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps. An Introduction*. Addison-Wesley, Reading, MA, 1992.

H. Ritter, K. Obermayer, K. Schulten, and J. Rubner. Self-organizing maps and adaptive filters. In E. Domani, J. L. van Hemmen, and K. Schulten, editors, *Physics of Neural Networks*, pages 281–306. Springer-Verlag, New York, 1991.

E. T. Rolls and M. J. Tovee. Sparseness of the neuronal representation of stimuli in the primate temporal visual cortex. *Journal of Neurophysiology*, 73(2):713–726, 1995.

A. Rosenwald, G. Wright, W. C. Chan, J. M. Connors, E. Campo, R. I. Fisher, R. D. Gascoyne, H. K. Muller-Hermelink, E. B. Smeland, J. M. Giltnane, E. M. Hurt, H. Zhao, L. Averett, L. Yang, W. H. Wilson, E. S. Jaffe, R. Simon, R. D. Klausner, J. Powell, P. L. Duffey, D. L. Longo, T. C. Greiner, D. D. Weisenburger, W. G. Sanger, B. J. Dave, J. C. Lynch, J. Vose, J. O. Armitage, E. Montserrat, A. L'opez-Guillermo, T. M. Grogan, T. P. Miller, M. LeBlanc, G. Ott, S. Kvaloy, J. Delabie, H. Holte, P. Krajci, T. Stokke, and L. M. Staudt. The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma. *New Engl. J. Med.*, 346:1937–1947, 2002.

J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4 (6):863–879, 1992.

B. Schölkopf, S. Mika, A. J. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction via approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, pages 147–152, Berlin, 1998. Springer Verlag.

B. Schölkopf and A. J. Smola. *Learning with kernels – Support Vector Machines, Reglarization, Optimization, and Beyond*. MIT Press, Cambridge, 2002.

J. Schultz, R. R. Copley, T. Doerks, C. P. Ponting, and P. Bork. A web-based tool for the study of genetically mobile domains. *Nucleic Acids Res.*, 28:231–23, 2000.

Q. Sheng, Y. Moreau, and B. De Moor. Biclustering micrarray data by Gibbs sampling. *Bioinformatics*, 19(Suppl. 2):ii196–ii205, 2003.

K. Sjölander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *CABIOS*, 12(4):327–345, 1996.

A. I. Su, M. P. Cooke, K. A.Ching, Y. Hakak, J. R. Walker, T. Wiltshire, A. P. Orth, R. G. Vega, L. M. Sapinoso, A. Moqrich, A. Patapoutian, G. M. Hampton, P. G. Schultz, and J. B. Hogenesch. Large-scale analysis of the human and mouse transcriptomes. *P. Natl. Acad. Sci. USA*, 99(7):4465–4470, 2002.

W. Talloen, D.-A. Clevert, S. Hochreiter, D. Amaratunga, L. Bijnens, S. Kass, and H. Göhlmann. I/NI-calls for the exclusion of non-informative genes: a highly effective filtering tool for microarray data. *Bioinformatics*, 23(21):2897–2902, 2007.

W. Talloen, S. Hochreiter, L. Bijnens, A. Kasim, Z. Shkedy, and D. Amaratunga. Filtering data from high-throughput experiments based on measurement reliability. *Proc. Natl. Acad. Sci. USA*, 107(46):173–174, 2010.

A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(Suppl. 1):S136–S144, 2002.

C. Tang, L. Zhang, I. Zhang, and M. Ramanathan. Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In *Pro. 2nd IEEE Int. Symp. on Bioinformatics and Bioengineering*, pages 41–48, 2001.

R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown. Clustering methods for the analysis of DNA microarray data. Technical report, Dept. of Health Research and Policy, Dept. of Genetics and Dept. of Biochemestry, Stanford University, 1999.

W. S. Torgerson. *Theory and Methods of Scaling*. Wiley, New York, 1958.

T. Van den Bulcke. *Robust algorithms for inferring regulatory networks based on gene expression measurements and biological prior information*. PhD thesis, Katholieke Universiteit Leuven, 2009.

L. vanderMaaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

L. J. van't Veer, H. Dai, M. J. van de Vijver, Y. D. He, A. A. Hart, M. Mao, H. L. Peterse, K. van der Kooy, M. J. Marton, A. T. Witteveen, G. J. Schreiber, R. M. Kerkhoven, C. Roberts, P. S. Linsley, R. Bernards, and S. H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–536, 2002.

H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proc. 2002 ACM SIGMOD Int. Conf. on Management of Data*, pages 394–405, 2002.

G. Yang and S. Amari. Adaptive online learning algorithms for blind source separation: Maximum entropy and minimum mutual information. *Neural Computation*, 9(7):1457–1482, 1997.

H. H. Yang, S. Amari, and A. Cichocki. Information back-propagation for blind separation of sources from non-linear mixtures. In *Proceedings of the International Conference on Neural Networks*, pages 2141–2146. Houston, USA, 1996.

J. Yang, H. Wang, W. Wang, and P. S. Yu. An improved biclustering method for analyzing gene expression profiles. *Int. J. Artif. Intell. T.*, 14(5):771–790, 2005.

S. Young and M. P. Yamane. Sparse population coding of faces in the inferotemporal cortex. *Science*, 256:1327–1331, 1992.

Y. Zhao and C. G. Atkeson. Implementing projection pursuit learning. *IEEE Transactions on Neural Networks*, 7(2):362–373, 1996.

M. Zibulevsky and B. A. Pearlmutter. Blind source separation by sparse decomposition. *Neural Computation*, 13(4):863–882, 2001.