Lecture 16: Summary and outlook

Felix Held, Mathematical Sciences

MSA220/MVE440 Statistical Learning for Big Data

24th May 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

The big topics

1. Statistical Learning
2. Supervised learning
   - Classification
   - Regression
3. Unsupervised learning
   - Clustering
4. Data representations and dimension reduction
5. Large scale methods

The big data paradigms

- **Small to medium sized data**
  - "Good old stats"
  - Typical methods: k-nearest neighbour (kNN), linear and quadratic discriminant analysis (LDA and QDA), Gaussian mixture models, ...
- **High-dimensional data**
  - big-p paradigm
  - Typical methods: Feature selection, penalized regression and classification (Lasso, ridge regression, shrunken centroids, ...), low-rank approximations (SVD, NMF), ...
  - **Curse of dimensionality**
- **Large scale data**
  - big-n paradigm (sometimes in combination with big-p)
  - Typical methods: Random forests (with its big-n extensions), subspace clustering, low-rank approximations (randomized SVD), ...

Statistical Learning

What is Statistical Learning?

Learn **a model** from **data** by minimizing **expected prediction error** determined by a **loss function**.

- **Model:** Find a model that is suitable for the data
- **Data:** Data with known outcomes is needed
- **Expected prediction error:** Focus on quality of prediction (predictive modelling)
- **Loss function:** Quantifies the discrepancy between observed data and predictions

Statistical Learning for Regression

- Theoretically best regression function for squared error loss

$$\widehat{f}(\mathbf{x}) = \mathbb{E}_{p(y|\mathbf{x})}[y]$$

- Approximate (1) or make model-assumptions (2)
  1. k-nearest neighbour regression

$$\mathbb{E}_{p(y|\mathbf{x})}[y] \approx \frac{1}{k} \sum_{\mathbf{x}_{i_l} \in N_k(\mathbf{x})} y_{i_l}$$

  2. linear regression (viewpoint: generalized linear models (GLM))

$$\mathbb{E}_{p(y|\mathbf{x})}[y] \approx \mathbf{x}^T \boldsymbol{\beta}$$

Statistical Learning for Classification

- Theoretically best classification rule for 0-1 loss and K possible classes (**Bayes rule**)

$$\hat{c}(\mathbf{x}) = \underset{1 \leq i \leq K}{\arg\max}\, p(i|\mathbf{x})$$

- Approximate (1) or make model-assumptions (2)
  1. k-nearest neighbour classification

$$p(i|\mathbf{x}) \approx \frac{1}{k} \sum_{\mathbf{x}_l \in N_k(\mathbf{x})} \mathbb{1}(i_l = i)$$

  2. Multi-class logistic regression

$$p(i|\mathbf{x}) = \frac{e^{\mathbf{x}^T \boldsymbol{\beta}^{(i)}}}{1 + \sum_{l=1}^{K-1} e^{\mathbf{x}^T \boldsymbol{\beta}^{(l)}}} \quad \text{and} \quad p(K|\mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{\mathbf{x}^T \boldsymbol{\beta}^{(l)}}}$$

Empirical error rates (I)

- **Training error**

$$R^{tr} = \frac{1}{n} \sum_{l=1}^{n} L(y_l, \widehat{f}(\mathbf{x}_l | \mathcal{T}))$$

where

$$\mathcal{T} = \{(y_l, \mathbf{x}_l) : 1 \leq l \leq n\}$$

- **Test error**

$$R^{te} = \frac{1}{m} \sum_{l=1}^{m} L(\tilde{y}_l, \widehat{f}(\tilde{\mathbf{x}}_l | \mathcal{T}))$$

where $(\tilde{y}_l, \tilde{\mathbf{x}}_l)$ for $1 \leq l \leq m$ are new samples from the same distribution as $\mathcal{T}$, i.e. $p(\mathcal{T})$.

Splitting up the data

- **Holdout method:** If we have a lot of samples, **randomly split** available data into **training set** and **test set**
- c**-fold cross-validation:** If we have few samples
  1. **Randomly split** available data into c equally large subsets, so-called **folds**.
  2. By taking turns, use $c - 1$ folds as the **training set** and the last fold as the **test set**

Approximations of expected prediction error

- Use **test error** for hold-out method, i.e.

$$R^{te} = \frac{1}{m} \sum_{l=1}^{m} L(\tilde{y}_l, \widehat{f}(\tilde{\mathbf{x}}_l | \mathcal{T}))$$

  where $(\tilde{y}_l, \tilde{\mathbf{x}}_l)$ for $1 \leq l \leq m$ are the elements in the test set.

- Use average **test error** for c-fold cross-validation, i.e.

$$R^{cv} = \frac{1}{n} \sum_{j=1}^{c} \sum_{(y_l, \mathbf{x}_l) \in \mathcal{F}_j} L(y_l, \widehat{f}(\mathbf{x}_l | \mathcal{F}_{-j}))$$

  where $\mathcal{F}_j$ is the j-th fold and $\mathcal{F}_{-j}$ is all data except fold j.

Note: For $c = 1$ this is called **leave-one-out cross validation (LOOCV)**

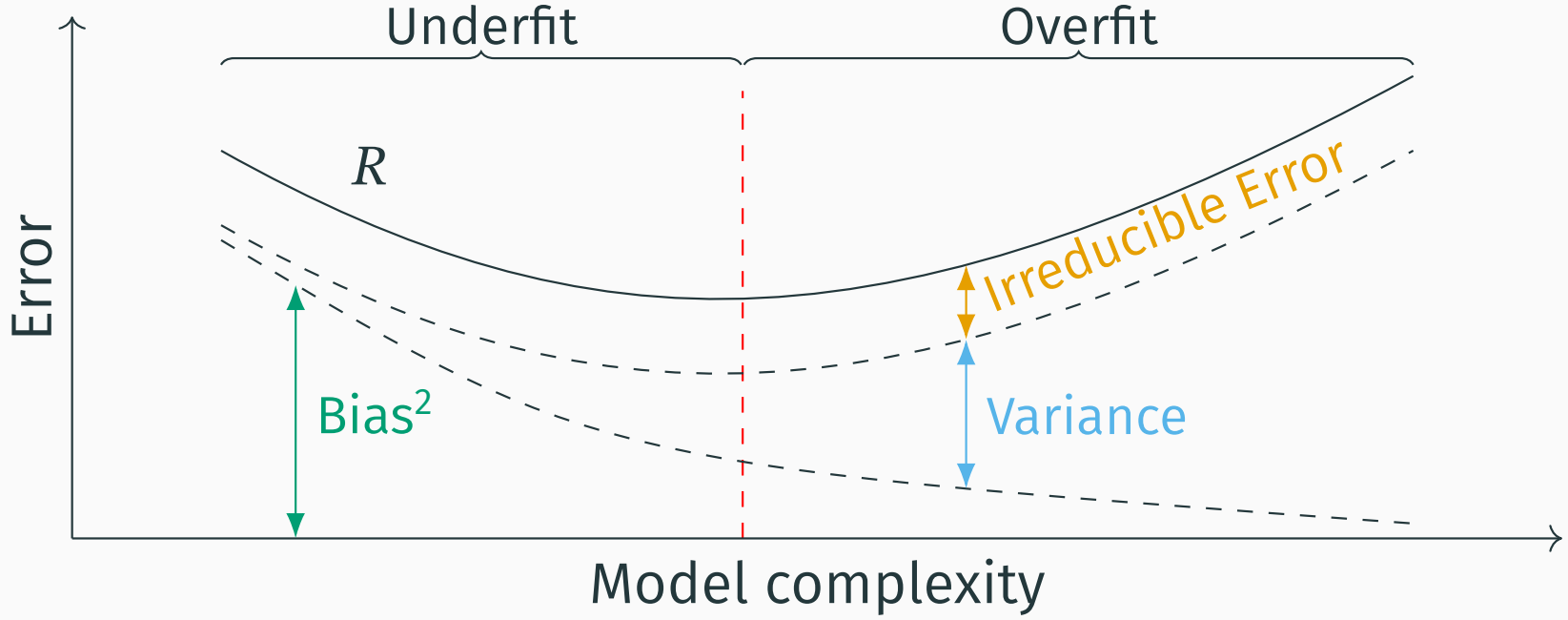
Careful data splitting

- **Note:** For the approximations to be justifiable, test and training sets need to be identically distributed
- **Splitting has to be done randomly**
- If data is unbalanced, then **stratification** is necessary. Examples:
  - Class imbalance
  - Continuous outcome is observed more often in some intervals than others (e.g. high values more often than low values)

Bias-Variance Tradeoff

Bias-Variance Decomposition

$$
\begin{aligned}
R &= \mathbb{E}_{p(\mathcal{T},\mathbf{x},y)}\left[(y-\widehat{f}(\mathbf{x}))^2\right] && \text{Total expected prediction error} \\
&= \sigma^2 && \text{Irreducible Error} \\
&+ \mathbb{E}_{p(\mathbf{x})}\left[\left(f(\mathbf{x})-\mathbb{E}_{p(\mathcal{T})}\left[\widehat{f}(\mathbf{x})\right]\right)^2\right] && \text{Bias}^2 \text{ averaged over } \mathbf{x} \\
&+ \mathbb{E}_{p(\mathbf{x})}\left[\mathrm{Var}_{p(\mathcal{T})}\left[\widehat{f}(\mathbf{x})\right]\right] && \text{Variance of } \widehat{f} \text{ averaged over } \mathbf{x}
\end{aligned}
$$



Classification

Overview

1. k-nearest neighbours (Lecture 1)
2. 0-1 regression (Lecture 2)
   - just an academic example - **do not use in practice**
3. Logistic regression (Lecture 2, both binary and multi-class; Lecture 11 for sparse case)
4. Nearest Centroids (Lecture 2) and shrunken centroids (Lecture 10)
5. Discriminant analysis (Lecture 2)
   - Many variants: linear (LDA), quadratic (QDA), diagonal/Naive Bayes, regularized (RDA; Lecture 5), Fisher's LDA/reduced-rank LDA (Lecture 6), mixture DA (Lecture 8)
6. Classification and Regression trees (CART) (Lecture 4)
7. Random Forests (Lecture 5 & 15)

Multiple angles on the same problem

1. **Bayes rule:** Approximate $p(i|\mathbf{x})$ and choose largest
   - e.g. kNN or logistic regression
2. **Model of the feature space:** Assume models for $p(\mathbf{x}|i)$ and $p(i)$ separately
   - e.g. discriminant analysis
3. **Partitioning methods:** Create explicit partitions of the feature space and assign each a class
   - e.g. CART or Random Forests

Finding the parameters of DA

- Notation: Write $p(i) = \pi_i$ and consider them as unknown parameters
- Given data $(i_l, \mathbf{x}_l)$ the likelihood maximization problem is

$$\arg\max_{\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\pi}} \prod_{l=1}^{n} N(\mathbf{x}_l|\boldsymbol{\mu}_{i_l}, \boldsymbol{\Sigma}_{i_l})\pi_{i_l} \quad \text{subject to} \quad \sum_{i=1}^{K} \pi_i = 1.$$

- Can be solved using a Lagrange multiplier (try it!) and leads to

$$\widehat{\pi}_i = \frac{n_i}{n}, \quad \text{with} \quad n_i = \sum_{l=1}^{n} \mathbb{1}(i_l = i)$$

$$\widehat{\boldsymbol{\mu}}_i = \frac{1}{n_i} \sum_{i_l=i} x_l$$

$$\widehat{\boldsymbol{\Sigma}}_i = \frac{1}{n_i - 1} \sum_{i_l=i} (x_l - \widehat{\boldsymbol{\mu}}_i)(x_l - \widehat{\boldsymbol{\mu}}_i)^T$$

Performing classification in DA

Bayes' rule implies the classification rule

$$c(\mathbf{x}) = \underset{1 \leq i \leq K}{\arg\max}\, N(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\pi_i$$

Note that since $\log$ is strictly increasing this is equivalent to

$$c(\mathbf{x}) = \underset{1 \leq i \leq K}{\arg\max}\, \delta_i(\mathbf{x})$$

where

$$\begin{aligned}\delta_i(\mathbf{x}) &= \log N(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) + \log \pi_i \\ &= \log \pi_i - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \frac{1}{2}\log|\boldsymbol{\Sigma}_i| \quad (+C)\end{aligned}$$

This is a quadratic function in $\mathbf{x}$.

Different levels of complexity

- This method is called **Quadratic Discriminant Analysis (QDA)**
- **Problem:** Many parameters that grow quickly with dimension
  - $K - 1$ for all π_i
  - $p \cdot K$ for all $\boldsymbol{\mu}_i$
  - $p(p+1)/2 \cdot K$ for all $\boldsymbol{\Sigma}_i$ (most costly)
- **Solution:** Replace covariance matrices $\boldsymbol{\Sigma}_i$ by a pooled estimate

$$\widehat{\boldsymbol{\Sigma}} = \sum_{i=1}^{K} \widehat{\boldsymbol{\Sigma}}_i \frac{n_i - 1}{n - K} = \frac{1}{n - K} \sum_{i=1}^{K} \sum_{i_l = i} (x_l - \widehat{\boldsymbol{\mu}}_i)(x_l - \widehat{\boldsymbol{\mu}}_i)^T$$

- **Simpler correlation and variance structure:** All classes are assumed to have the same correlation structure between features

Performing classification in the simplified case

As before, consider

$$c(\mathbf{x}) = \underset{1 \leq i \leq K}{\arg\max}\, \delta_i(\mathbf{x})$$

where

$$\delta_i(\mathbf{x}) = \log \pi_i + \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i - \frac{1}{2} \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i \quad (+C)$$

This is a linear function in $\mathbf{x}$. The method is therefore called **Linear Discriminant Analysis (LDA)**.

Even more simplifications

Other simplifications of the correlation structure are possible

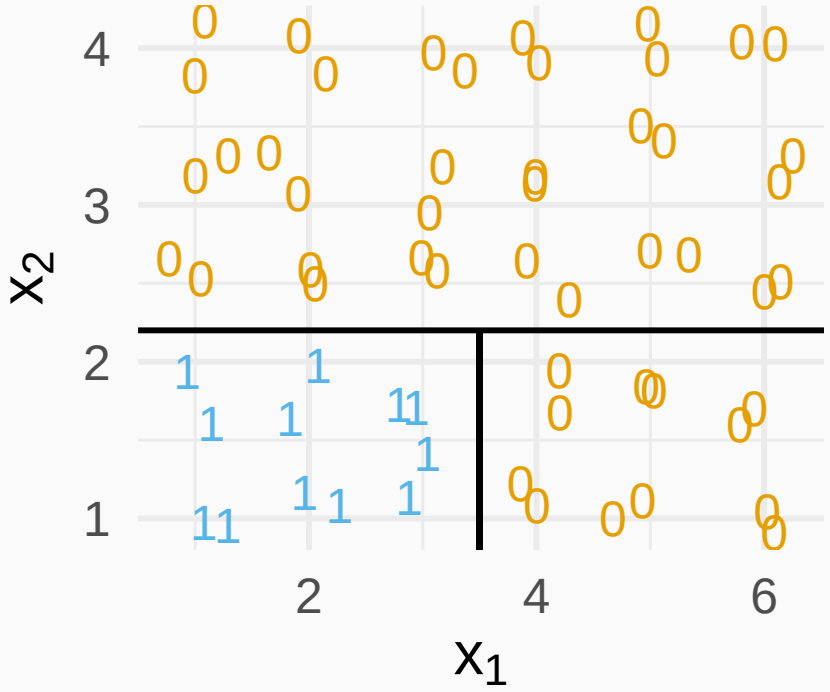
- Ignore all correlations between features but allow different variances, i.e. $\mathbf{\Sigma}_i = \mathbf{\Lambda}_i$ for a diagonal matrix $\mathbf{\Lambda}_i$ (**Diagonal QDA** or **Naive Bayes' Classifier**)
- Ignore all correlations and make feature variances equal, i.e. $\mathbf{\Sigma}_i = \mathbf{\Lambda}$ for a diagonal matrix $\mathbf{\Lambda}$ (**Diagonal LDA**)
- Ignore correlations and variances, i.e. $\mathbf{\Sigma}_i = \sigma^2 \mathbf{I}_{p \times p}$ (**Nearest Centroids adjusted for class frequencies** π_i)

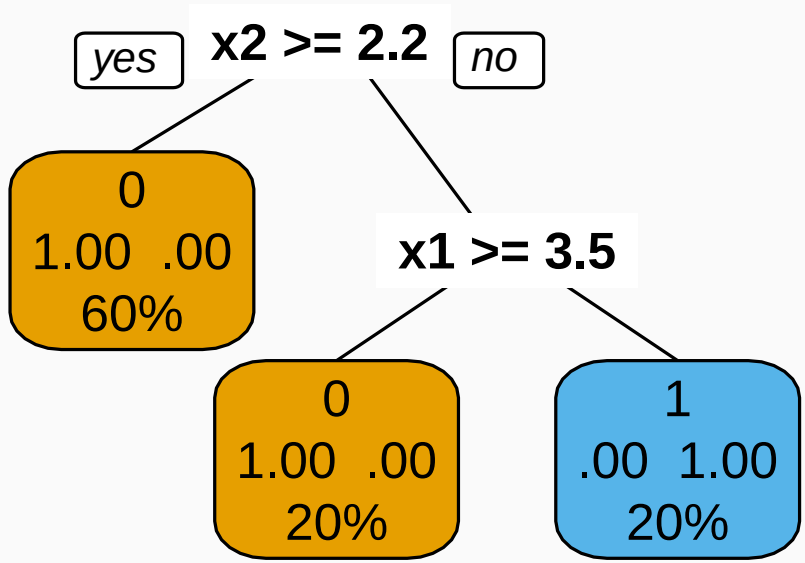
Classification and Regression Trees (CART)

- **Complexity of partitioning:**

  Arbitrary Partition > Rectangular Partition > Partition from a sequence of binary splits

- Classification and Regression Trees create a **sequence of binary axis-parallel splits** in order to **reduce variability** of values/classes in each region





Bootstrap aggregation (bagging)

1. Given a training sample $(y_l, \mathbf{x}_l)$ or $(i_l, \mathbf{x}_l)$, we want to fit a predictive model $\widehat{f}(\mathbf{x})$
2. For $b = 1, \ldots, B$, form bootstrap samples of the training data and fit the model, resulting in $\widehat{f}_b(\mathbf{x})$
3. Define

$$\widehat{f}_{\text{bag}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}_b(\mathbf{x})$$

   where $\widehat{f}_b(\mathbf{x})$ is a continuous value for a regression problem or a vector of class probabilities for a classification problem

Majority vote can be used for classification problems instead of averaging

Computational procedure

1. Given a training sample with p features, do for $b = 1, \ldots, B$
   - 1.1 Draw a bootstrap sample of size n from training data (with replacement)
   - 1.2 Grow a tree T_b until each node reaches minimal node size $n_{\min}$
     - 1.2.1 Randomly select m variables from the p available
     - 1.2.2 Find best splitting variable among these m
     - 1.2.3 Split the node
2. For a new $\mathbf{x}$ predict

$$\text{Regression:} \quad \widehat{f}_{rb}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x})$$

$$\text{Classification:} \quad \text{Majority vote at } \mathbf{x} \text{ across trees}$$

Note: Step 1.2.1 leads to less correlation between trees built on bootstrapped data.

Regression and feature selection

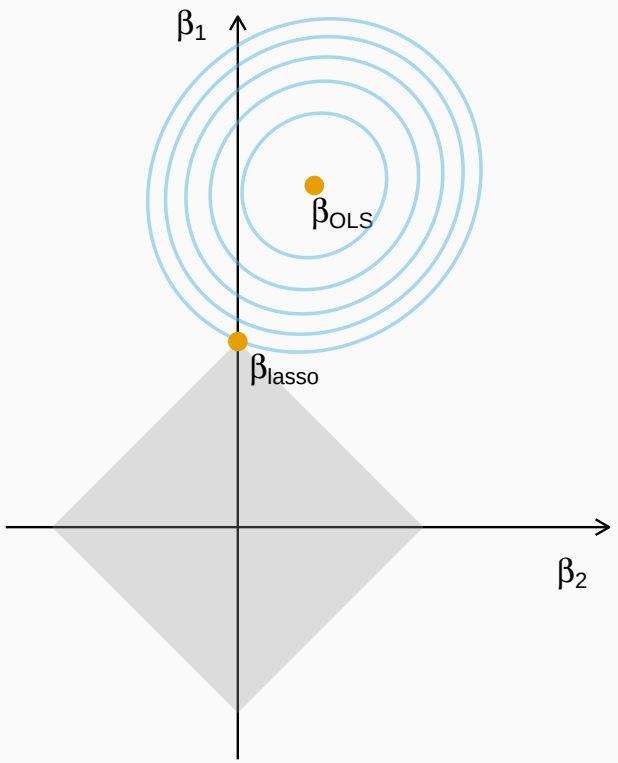
Overview

1. k-nearest neighbours (Lecture 1)
2. Linear regression (Lecture 1)
3. Filtering (Lecture 9)
   - F-score, mutual information, RF variable importance, ...
4. Wrapping (Lecture 9)
   - Forward-, backward-, and best subset selection
5. Penalized regression and variable selection
   - Ridge regression and lasso (Lecture 9), group lasso and elastic net (Lecture 10), adaptive lasso and SCAD (Lecture 11)
6. Computation of the lasso (Lecture 10)
7. Applications of penalisation
   - For Discriminant Analysis: Shrunken centroids (Lecture 10)
   - For GLM: sparse multi-class logistic regression (Lecture 11)
   - For networks: Graphical lasso (Lecture 14)

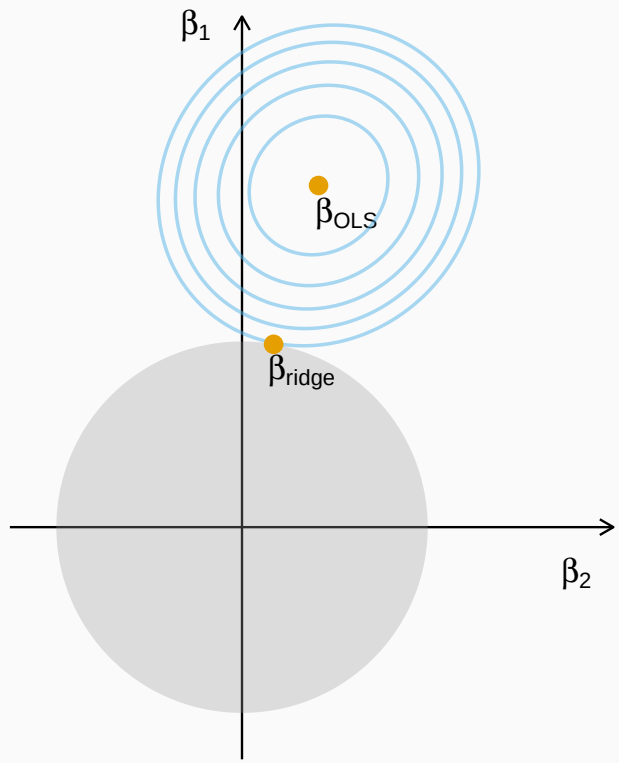
Intuition for the penalties

The least squares RSS is minimized for $\boldsymbol{\beta}_{\mathrm{OLS}}$. If a constraint is added ($||\boldsymbol{\beta}||_q^q \leq t$) then the RSS is minimized by the closest $\boldsymbol{\beta}$ possible that fulfills the constraint.

Lasso



Ridge

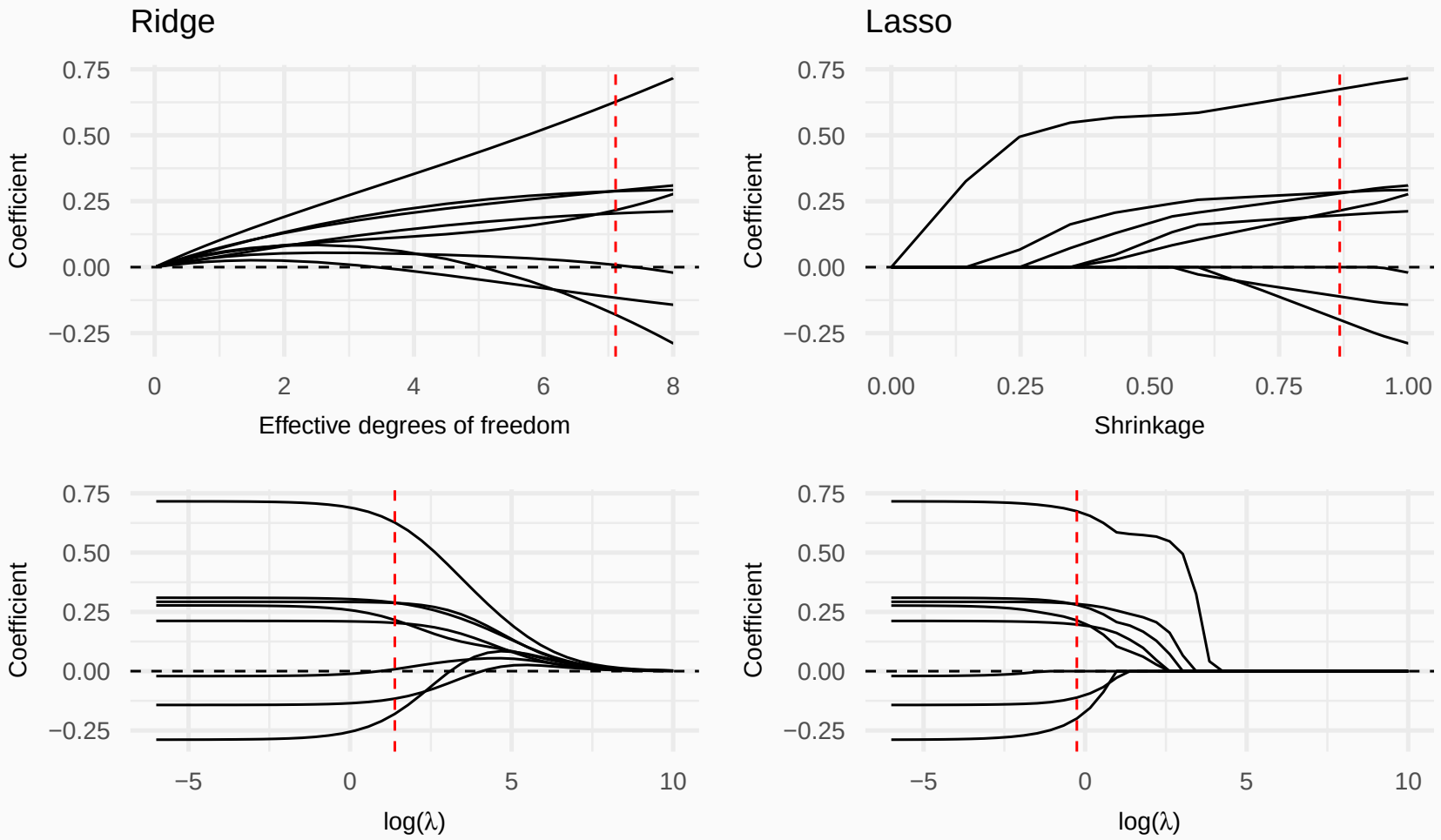


The blue lines are the contour lines for the RSS.

A regularisation path

Prostate cancer dataset $(n = 67, p = 8)$

Red dashed lines indicate the λ selected by cross-validation



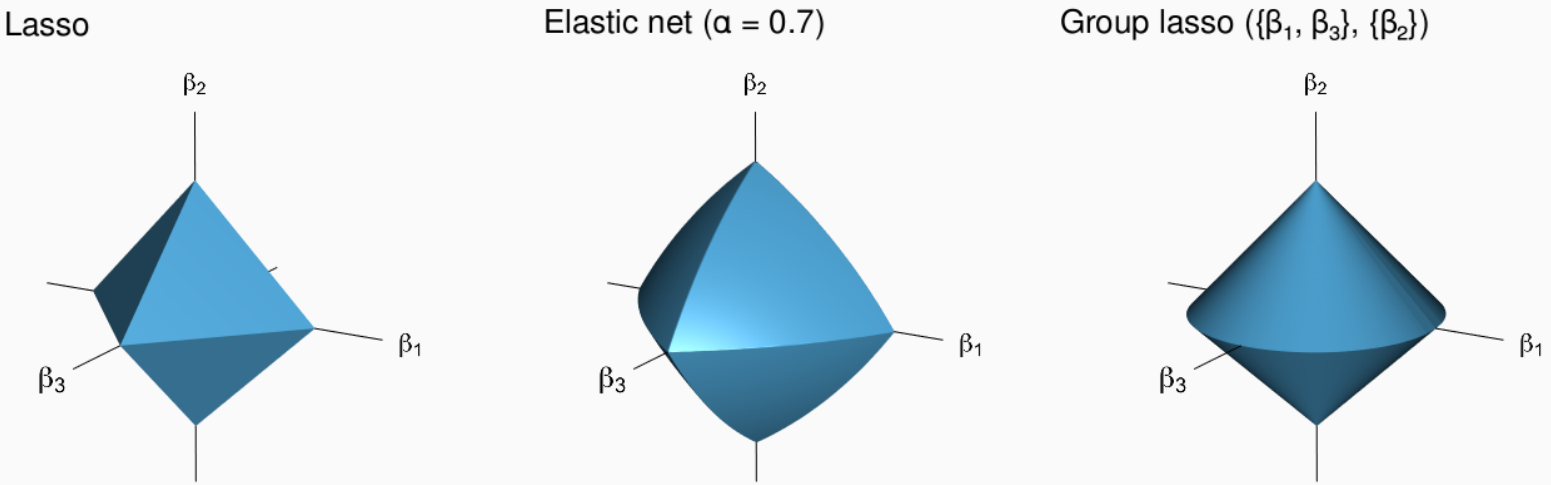
Potential caveats of the lasso

- **Sparsity of the true model:**
  - The lasso only works if the data is generated from a sparse process.
  - However, a dense process with many variables and not enough data or high correlation between predictors can be unidentifiable either way
- **Correlations:** Many non-relevant variables correlated with relevant variables can lead to the selection of the wrong model, even for large n
- **Irrepresentable condition:** Split $\mathbf{X}$ such that $\mathbf{X}_1$ contains all **relevant variables** and $\mathbf{X}_2$ contains all **irrelevant variables**. If

$$|(\mathbf{X}_2^T \mathbf{X}_1)(\mathbf{X}_1^T \mathbf{X}_1)^{-1}| < 1 - \eta$$

for some $\eta > 0$ then the lasso is (almost) guaranteed to pick the true model

Elnet and group lasso



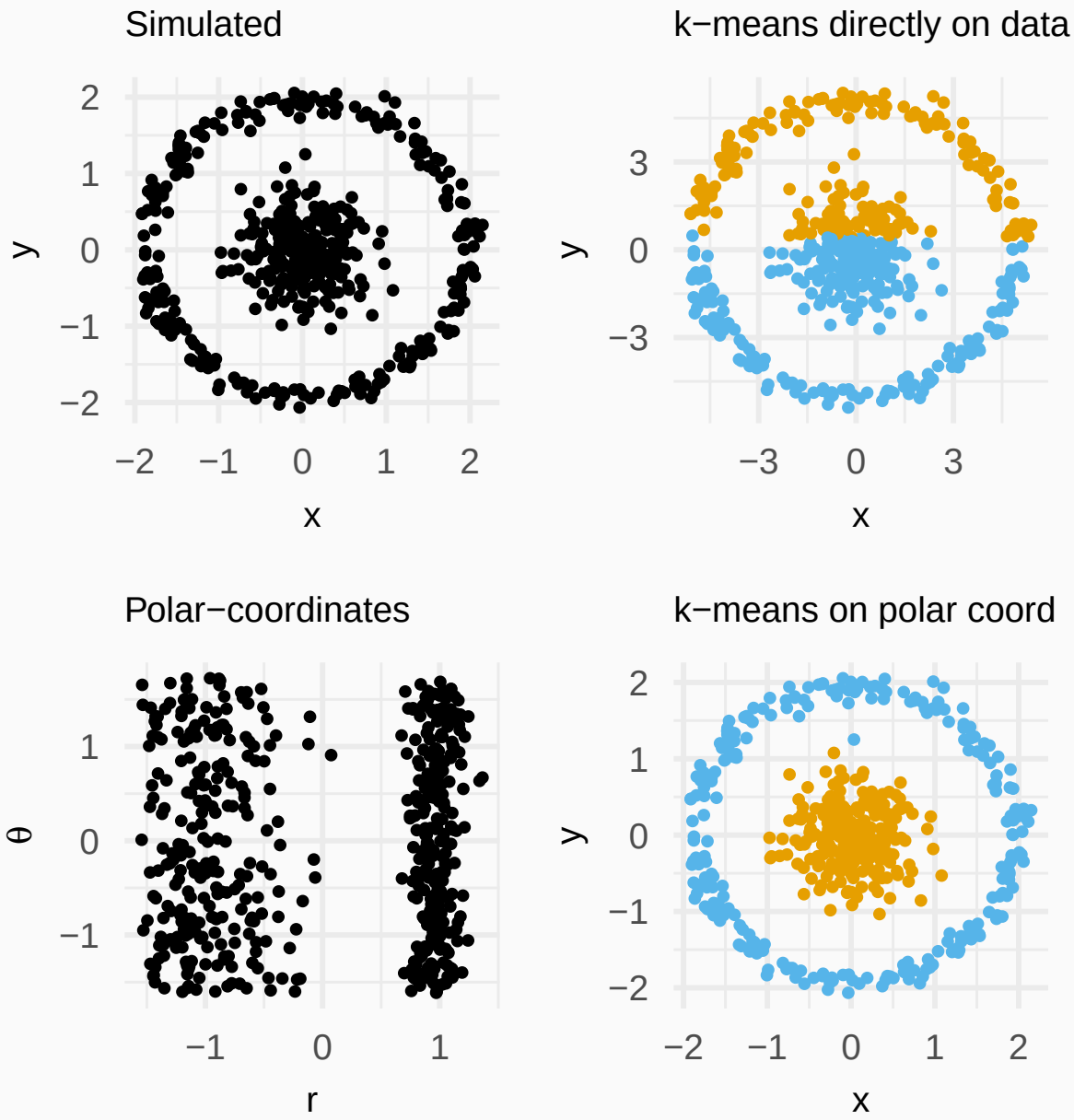
- The lasso sets variables exactly to zero either on a corner (all but one) or along an edge (fewer).
- The elastic net similarly sets variables exactly to zero on a corner or along an edge. In addition, the curved edges encourage coefficients to be closer together.
- The group lasso has actual information about groups of variables. It encourages whole groups to be zero simultaneously. Within a group, it encourages the coefficients to be as similar as possible.

Clustering

Overview

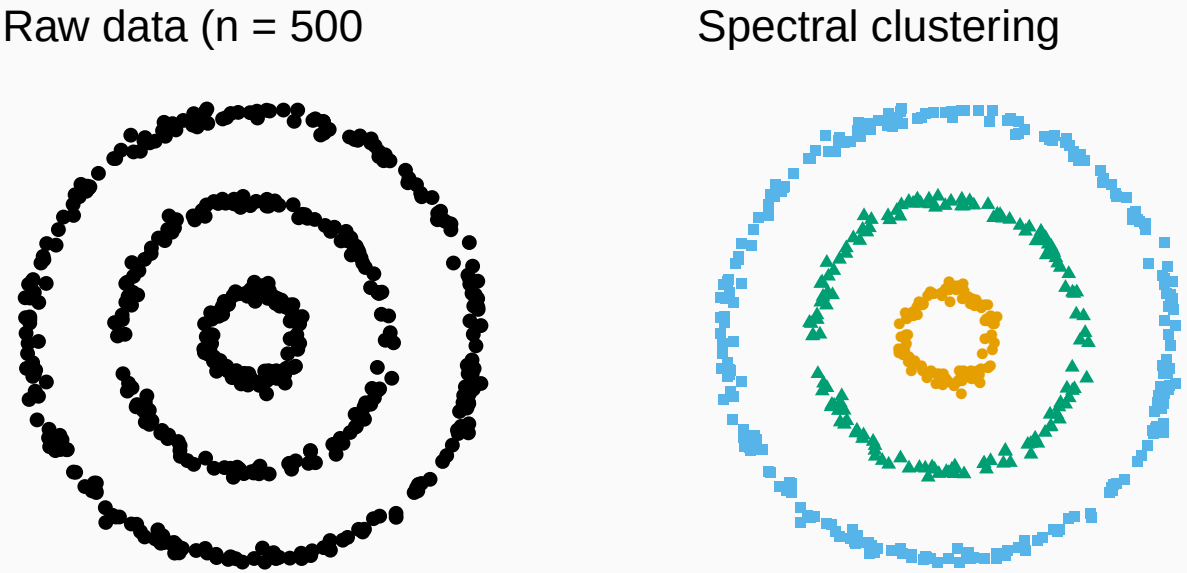
1. Combinatorial Clustering (Lecture 6)
2. k-means (Lecture 6)
3. Partion around medoids (PAM)/k-medoids (Lecture 7)
4. Cluster count selection (Lecture 7 & 8)
   - Ellbow heuristic, Silhouette Width, Cluster Strength Prediction, Bayesian Information Criterion (BIC) for GMM
5. Hiearchical clustering (Lecture 7)
6. Gaussian Hierarchical Models (GMM; Lecture 8)
7. DBSCAN (Lecture 8)
8. Non-negative matrix factorisation (NMF; Lecture 11)
9. Subspace clustering (Lecture 14)
   - In particular, CLIQUE and ProClus
10. Spectral clustering (Lecture 14)

k-means and the assumption of spherical geometry



Spectral Clustering

1. Determine the weighted adjacency matrix $\mathbf{W}$ and the graph Laplacian $\mathbf{L}$
2. Find the K smallest eigenvalues of $\mathbf{L}$ that are near zero and well separated from the others
3. Find the corresponding eigenvectors $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_K) \in \mathbb{R}^{n \times K}$ and use k-means on the rows of $\mathbf{U}$ to determine cluster membership



Hierarchical Clustering

Procedural idea:

1. **Initialization:** Let each observation $\mathbf{x}_l$ be in its own cluster g_l^0 for $l = 1, \dots, n$
2. **Joining:** In step i, join the two clusters g_l^{i-1} and g_m^{i-1} that are closest to each other resulting in $n - i$ clusters
3. After $n - 1$ steps all observations are in one big cluster

Subjective choices:

- How do we measure distance between observations?
- What is **closeness** for clusters?

Linkage

Cluster-cluster distance is called **linkage**

Distance between clusters g and h

1. **Average Linkage:**

$$d(g,h) = \frac{1}{|g| \cdot |h|} \sum_{\substack{\mathbf{x}_l \in g \\ \mathbf{x}_m \in h}} \mathbf{D}_{l,m}$$

2. **Single Linkage**

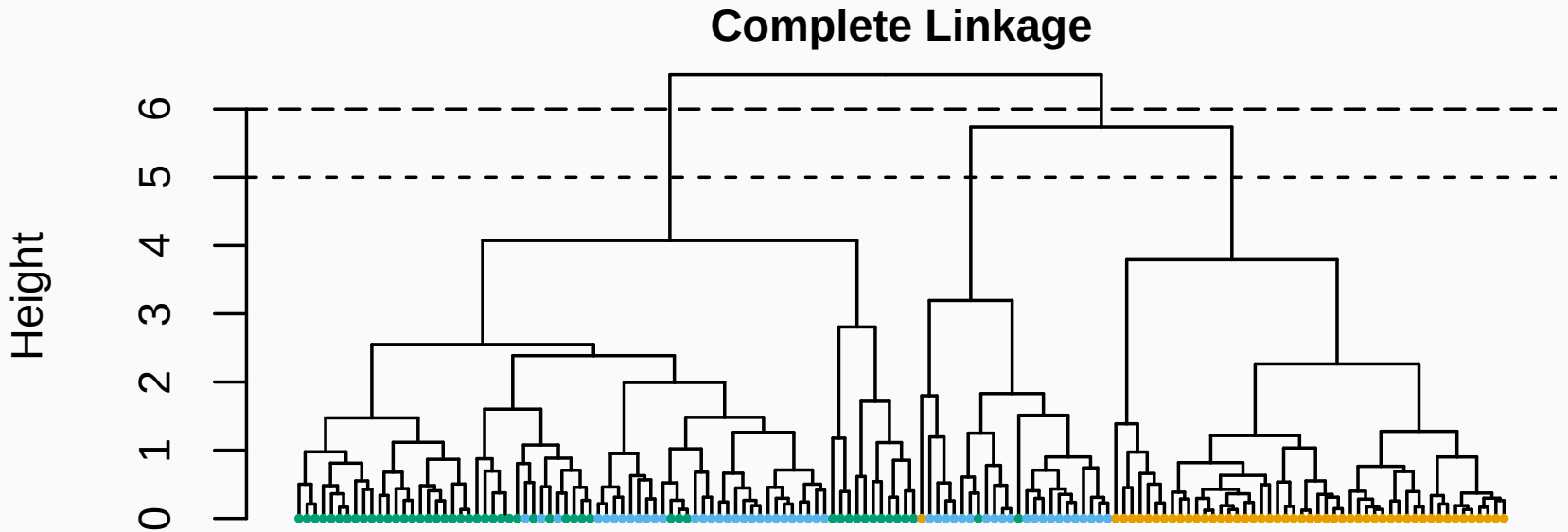
$$d(g,h) = \min_{\substack{\mathbf{x}_l \in g \\ \mathbf{x}_m \in h}} \mathbf{D}_{l,m}$$

3. **Complete Linkage**

$$d(g,h) = \max_{\substack{\mathbf{x}_l \in g \\ \mathbf{x}_m \in h}} \mathbf{D}_{l,m}$$

Dendrograms

Hierarchical clustering applied to **iris dataset**



- Leaf colours represent iris type: setosa, versicolor and virginica
- **Height** is the distance between clusters
- The tree can be **cut** at a certain height to achieve a final clustering. Long branches mean large increase in within cluster scatter at join

Remember QDA

In Quadratic Discriminant Analysis (QDA) we assumed

$$p(\mathbf{x}|i) = N(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \quad \text{and} \quad p(i) = \pi_i$$

This is known as a **Gaussian Mixture Model (GMM)** for $\mathbf{x}$ where

$$p(\mathbf{x}) = \sum_{i=1}^{K} p(i)p(\mathbf{x}|i) = \sum_{i=1}^{K} \pi_i N(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

QDA used that the classes i_l and feature vectors $\mathbf{x}_l$ of the observations were known to calculate π_i, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$.

What if we only know the features $\mathbf{x}_l$?

Expectation-Maximization for GMMs

Finding the MLE for parameters θ in GMMs results in an iterative process called **Expectation-Maximization (EM)**

1. Initialize θ
2. **E-Step:** Update

$$\eta_{li} = \frac{\pi_i N(\mathbf{x}_l | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{j=1}^{K} \pi_j N(\mathbf{x}_l | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

3. **M-Step:** Update

$$\boldsymbol{\mu}_i = \frac{\sum_{l=1}^{n} \eta_{li} \mathbf{x}_l}{\sum_{l=1}^{n} \eta_{li}} \qquad \pi_i = \frac{\sum_{l=1}^{n} \eta_{li}}{n}$$

$$\boldsymbol{\Sigma}_i = \frac{1}{\sum_{l=1}^{n} \eta_{li}} \sum_{l=1}^{n} \eta_{li} (\mathbf{x}_l - \boldsymbol{\mu}_i)(\mathbf{x}_l - \boldsymbol{\mu}_i)^T$$

4. Repeat steps 2 and 3 until convergence

Density-based clusters

A **cluster** C is a set of points in D s.th.

Density-based clusters

A **cluster** C is a set of points in D s.th.

1. If $p \in C$ and q is density-reachable from p then $q \in C$ (**maximality**)

Density-based clusters

A **cluster** C is a set of points in D s.th.

1. If $p \in C$ and q is density-reachable from p then $q \in C$ (**maximality**)
2. For all $p, q \in C$: p and q are density-connected (**connectivity**)

Density-based clusters

A **cluster** C is a set of points in D s.th.

1. If $p \in C$ and q is density-reachable from p then $q \in C$ (**maximality**)
2. For all $p, q \in C$: p and q are density-connected (**connectivity**)

This leads to **three types of points**

1. **Core points:** Part of a cluster and at least $n_{\min}$ points in neighbourhood
2. **Border points:** Part of a cluster but not core points
3. **Noise:** Not part of any cluster

Density-based clusters

A **cluster** C is a set of points in D s.th.

1. If $p \in C$ and q is density-reachable from p then $q \in C$ (**maximality**)
2. For all $p, q \in C$: p and q are density-connected (**connectivity**)

This leads to **three types of points**

1. **Core points:** Part of a cluster and at least $n_{\min}$ points in neighbourhood
2. **Border points:** Part of a cluster but not core points
3. **Noise:** Not part of any cluster

Note: Border points can have **non-unique cluster assignments**

DBSCAN algorithm

Computational procedure:

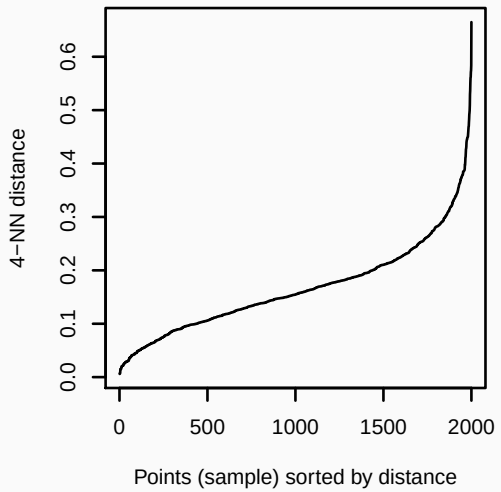
1. Go through each point p in the dataset D
2. If it has already been processed take the next one
3. Else determine its ε-neighbourhood. If less than $n_{\min}$ points in neighbourhood, label as noise. Otherwise, start a new cluster.
4. Find all points that are density-reachable from p and add them to the cluster.

Dependence on $n_{\min}$

- Controls how easy it is to connect components in a cluster
  - Too small and most points are core points, creating many small clusters
  - Too large and few points are core points, leading to many noise labelled observations
- A cluster has by definition at least $n_{\min}$ points
- Choice of $n_{\min}$ is very dataset dependent
- Tricky in high-dimensional data (**curse of dimensionality**, everything is far apart)

Dependence on ε

- Controls how much of the data will be clustered
  - Too small and small gaps in clusters cannot be bridged, leading to isolated islands in the data
  - Too large and everything is connected
- Choice of ε is also dataset dependent but there is a **decision tool**
  - Determine distance to the k nearest neighbours for each point in the dataset
  - Inside clusters, increasing k should not lead to a large increase of d
  - The optimal ε is supposed to be roughly at the knee



Dimension reduction and data representation

Overview

1. Principal Component Analysis (PCA; Lecture 5)
2. Singular Value Decomposition (SVD; Lecture 5, 11 & 15)
3. Factor Analysis (Lecture 11)
4. Non-negative matrix factorization (NMF; Lecture 11 & 12)
5. Kernel-PCA (kPCA; Lecture 12)
6. Multi-dimensional scaling (MDS; Lecture 13)
   - Classical scaling and metric MDS for general distance matrices
7. Isomap (Lecture 13)
8. t-distributed Stochastic Neighbour Embedding (tSNE; Lecture 13)
9. Laplacian Eigenmaps (Lecture 14)

Principal Component Analysis (PCA)

Computational Procedure:

1. **Centre** and **standardize** the columns of the data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$
2. Calculate the **empirical covariance matrix** $\widehat{\mathbf{\Sigma}} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$
3. Determine the **eigenvalues** λ_j and corresponding orthonormal **eigenvectors** $\mathbf{r}_j$ of $\widehat{\mathbf{\Sigma}}$ for $j = 1, \dots, p$ and order them such that

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$$

4. The vectors $\mathbf{r}_j$ give the direction of the **principal components (PC)** $\mathbf{r}_j^T\mathbf{x}$ and the eigenvalues λ_j are the **variances along the PC directions**

Note: Set $\mathbf{R} = (\mathbf{r}_1, \dots, \mathbf{r}_p)$ and $\mathbf{D} = \mathrm{diag}(\lambda_1, \dots, \lambda_p)$ then

$$\widehat{\mathbf{\Sigma}} = \mathbf{RDR}^T \quad \text{and} \quad \mathbf{R}^T\mathbf{R} = \mathbf{RR}^T = \mathbf{I}_p$$

PCA and Dimension Reduction

Recall: For a matrix $\mathbf{A} \in \mathbb{R}^{k \times k}$ with eigenvalues $\lambda_1, \dots, \lambda_k$ it holds that

$$\operatorname{tr}(\mathbf{A}) = \sum_{j=1}^{k} \lambda_j$$

For the empirical covariance matrix $\widehat{\mathbf{\Sigma}}$ and the variance of the j-th feature $\operatorname{Var}[x_j]$

$$\operatorname{tr}(\widehat{\mathbf{\Sigma}}) = \sum_{j=1}^{p} \operatorname{Var}[x_j] = \sum_{j=1}^{p} \lambda_j$$

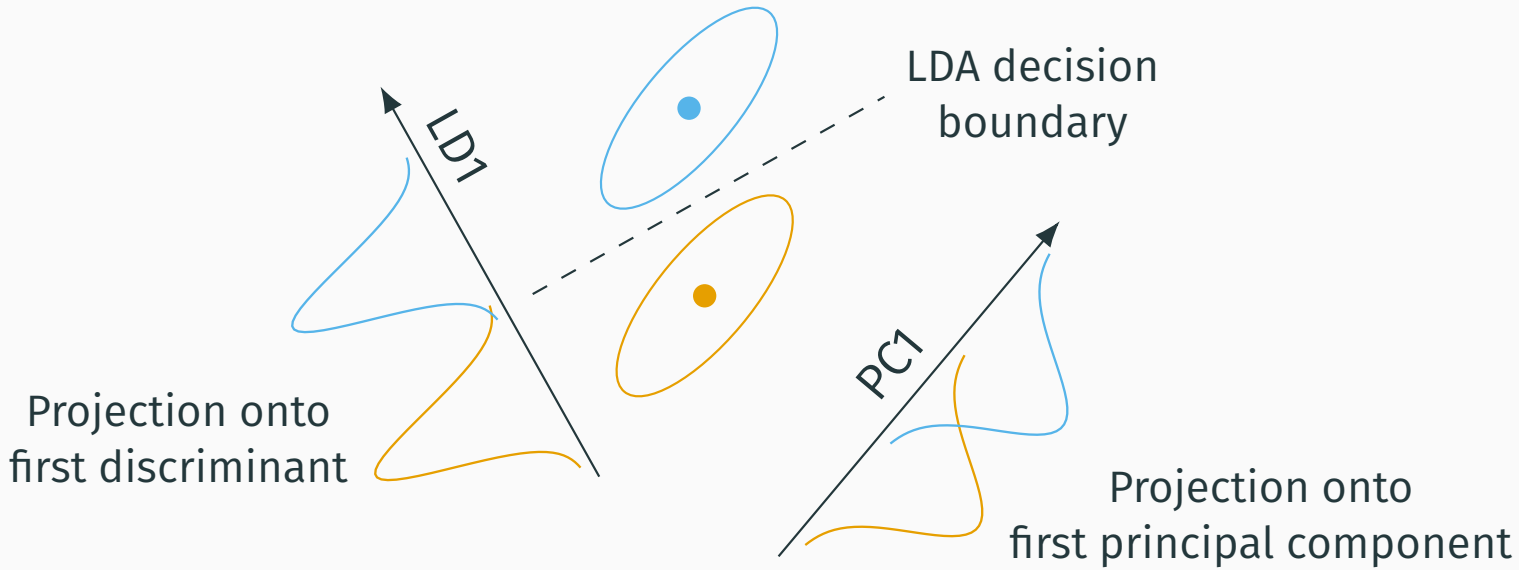
is called the **total variation**.

Using only the first $m < p$ principal components leads to

$$\frac{\lambda_1 + \cdots + \lambda_m}{\lambda_1 + \cdots + \lambda_p} \cdot 100\% \quad \text{of } \textbf{explained variance}$$

Better data projection for classification?

Idea: Find directions along which projections result in minimal within-class scatter and maximal between-class separation.



Non-negative matrix factorization (NMF)

A **non-negative matrix factorisation** of $\mathbf{X}$ with rank q solves

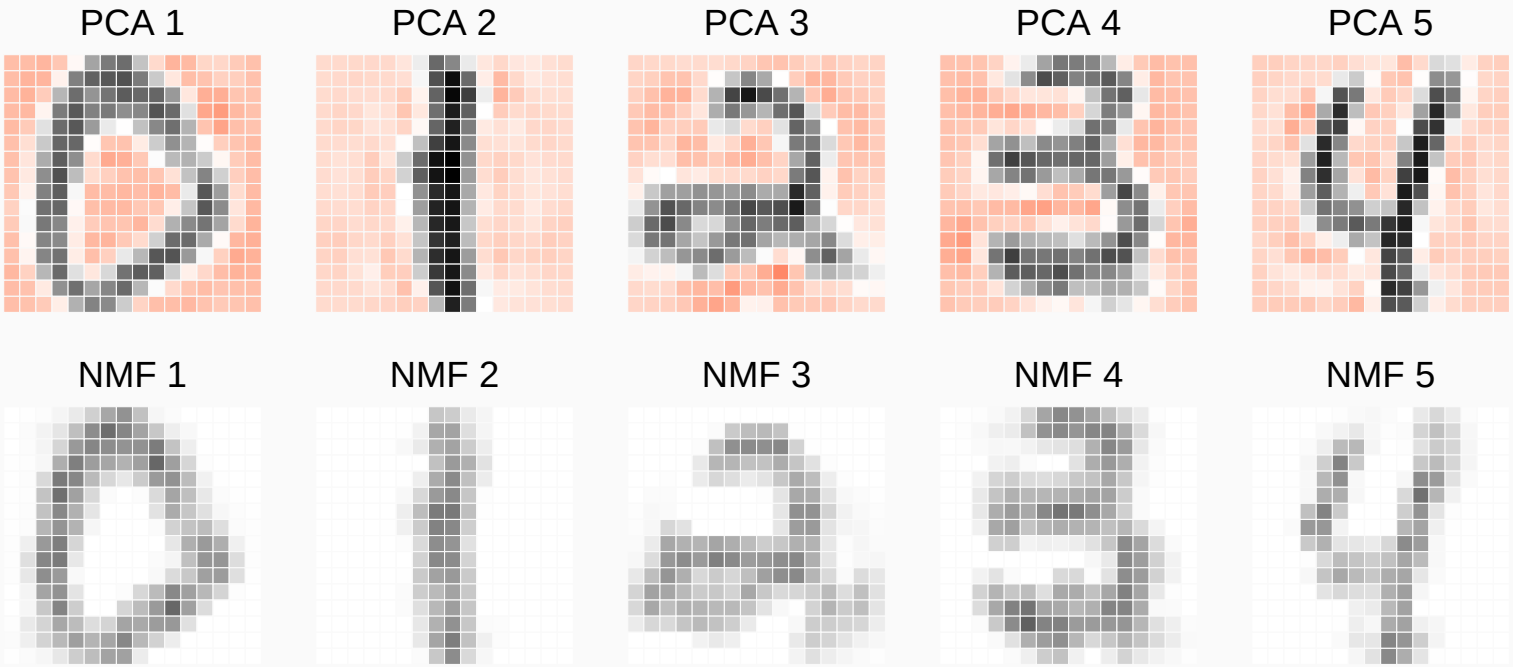
$$\underset{\mathbf{W}\in\mathbb{R}^{p\times q},\mathbf{H}\in\mathbb{R}^{q\times n}}{\arg\min} \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2 \quad \text{such that} \quad \mathbf{W}\geq 0, \mathbf{H}\geq 0$$

- **Sum of positive layers:** $\mathbf{X}\approx\sum_{j=1}^{q}\mathbf{W}_{\cdot j}\mathbf{H}_{\cdot j}^T$
- Non-negativity constraint leads to **sparsity in basis** (in $\mathbf{W}$) and **coefficients** (in $\mathbf{H}$) [example on next slides]
- NP-hard problem, i.e. no general algorithm exists

SVD vs NMF – Example: Reconstruction

MNIST-derived zip code digits ($n = 1000$, $p = 256$)

100 samples are drawn randomly from each class to keep the problem balanced.

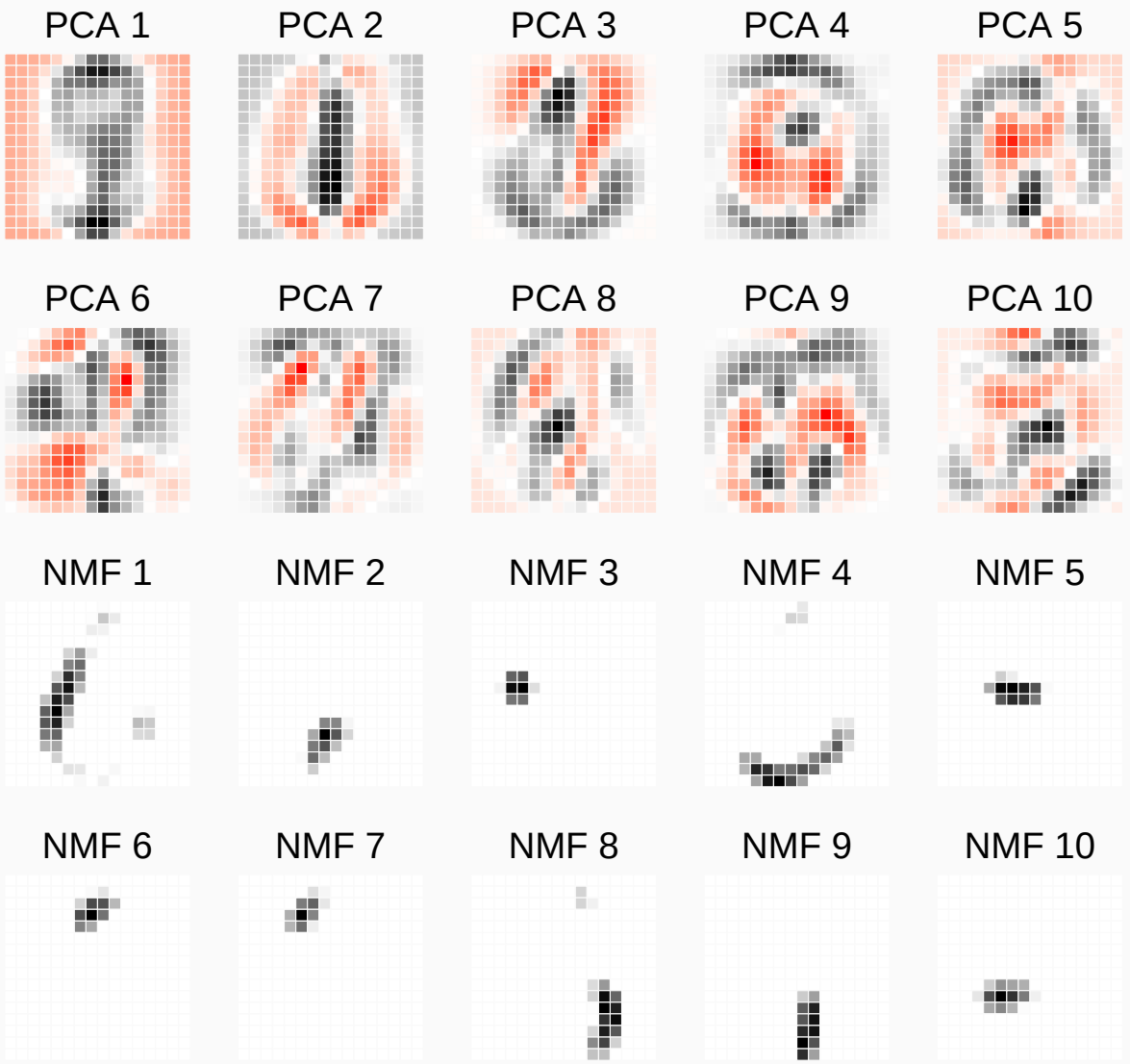


Red-ish colours are for negative values, white is around zero and dark stands for positive values

SVD vs NMF – Example: Basis Components

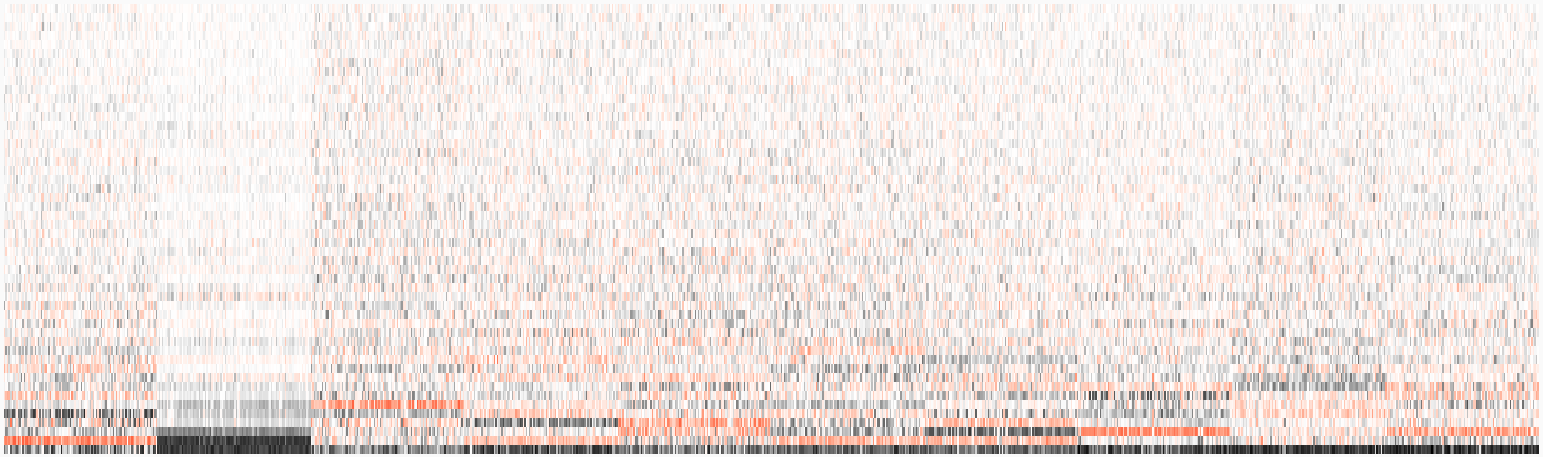
Large difference between SVD/PCA and NMF basis components

NMF captures **sparse characteristic parts** while PCA components capture more global features.

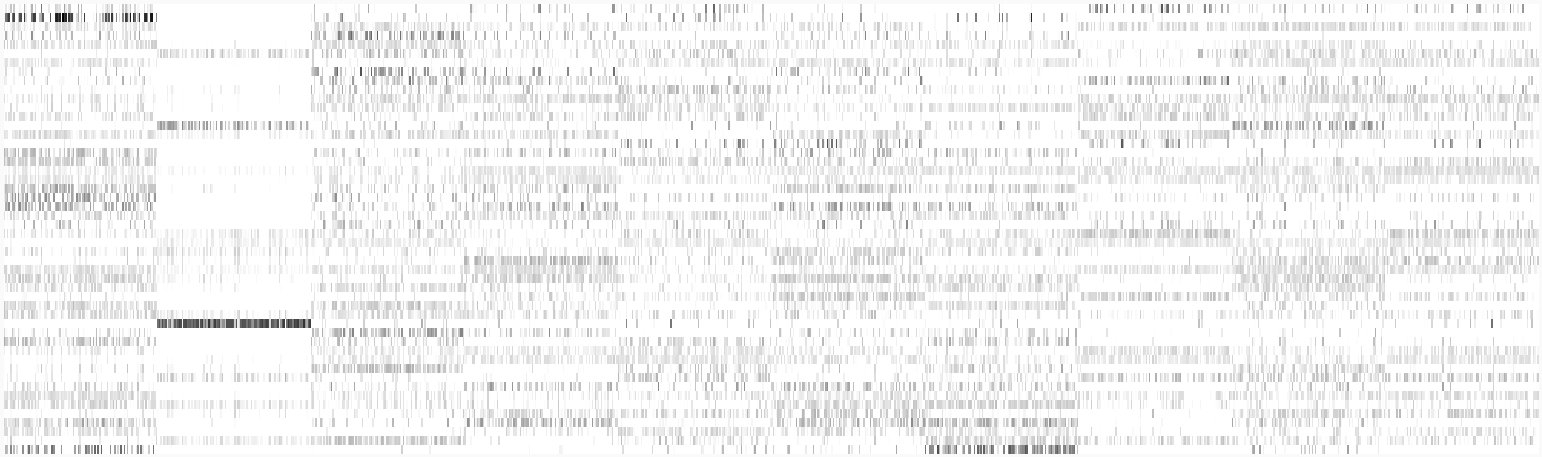


SVD vs NMF – Example: Coefficients

SVD coefficients



NMF coefficients



Note the additional **sparsity** in the NMF coefficients.

t-distributed Stochastic Neighbour Embedding (tSNE)

t-distributed stochastic neighbour embedding (tSNE) follows a similar strategy as Isomap, in the sense that it **measures distances locally**.

Idea: Measure distance of feature $\mathbf{x}_l$ to another feature $\mathbf{x}_i$ proportional to the likelihood of $\mathbf{x}_i$ under a Gaussian distribution centred at $\mathbf{x}_l$ with an isotropic covariance matrix.

Computation of tSNE

For feature vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$, set

$$p_{i|l} = \frac{\exp(-\|\mathbf{x}_l - \mathbf{x}_i\|_2^2/(2\sigma_l^2))}{\sum_{k \neq l} \exp(-\|\mathbf{x}_l - \mathbf{x}_k\|_2^2/(2\sigma_l^2))} \quad \text{and} \quad p_{il} = \frac{p_{i|l} + p_{l|i}}{2n},\ p_{ll} = 0$$

The variances σ_i^2 are chosen such that the **perplexity** (here: **approximate number of close neighbours**) of each marginal distribution (the $p_{i|l}$ for fixed l) is constant.

In the lower-dimensional embedding distance between $\mathbf{y}_1, \dots, \mathbf{y}_n$ is measured with a **t-distribution with one degree of freedom** or **Cauchy distribution**

$$q_{il} = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_l\|_2^2\right)^{-1}}{\sum_{k \neq j} \left(1 + \|\mathbf{y}_k - \mathbf{y}_j\|_2^2\right)^{-1}} \quad \text{and} \quad q_{ll} = 0$$

To determine the $\mathbf{y}_l$ the KL divergence between the distributions $P = (p_{il})_{il}$ and $Q = (q_{il})_{il}$ is minimized with gradient descent

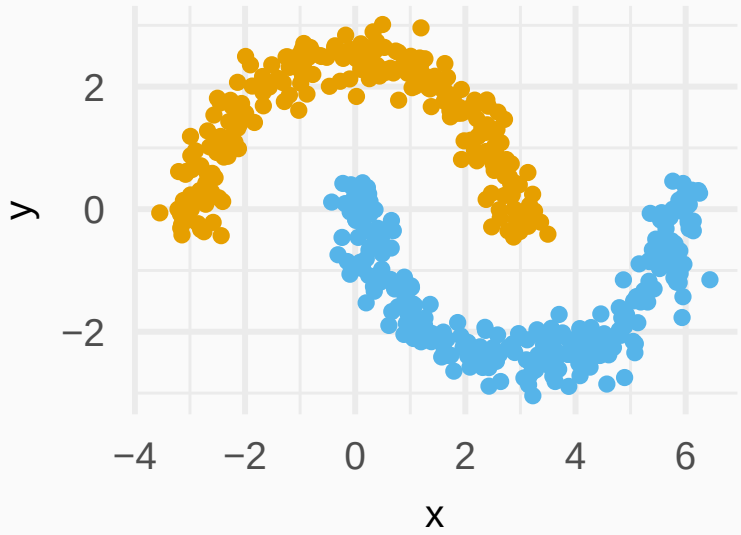
$$\mathrm{KL}(P||Q) = \sum_{i \neq l} p_{il} \log \frac{p_{il}}{q_{il}}$$

Caveats of tSNE

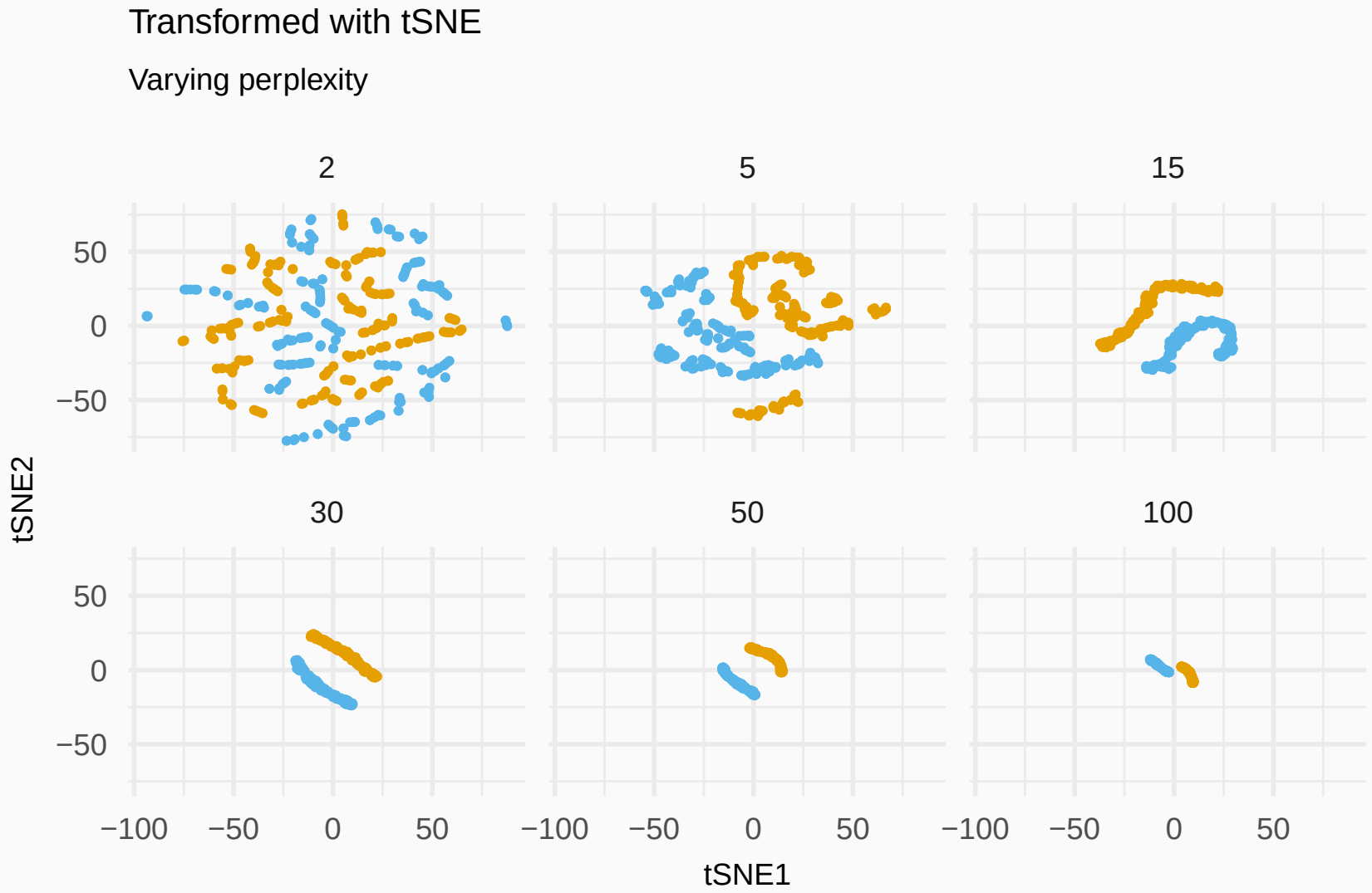
tSNE is a powerful method but comes with some difficulties as well

- Convergence to local minimum (i.e. repeated runs can give different results)
- Perplexity is hard to tune (as with any tuning parameter)

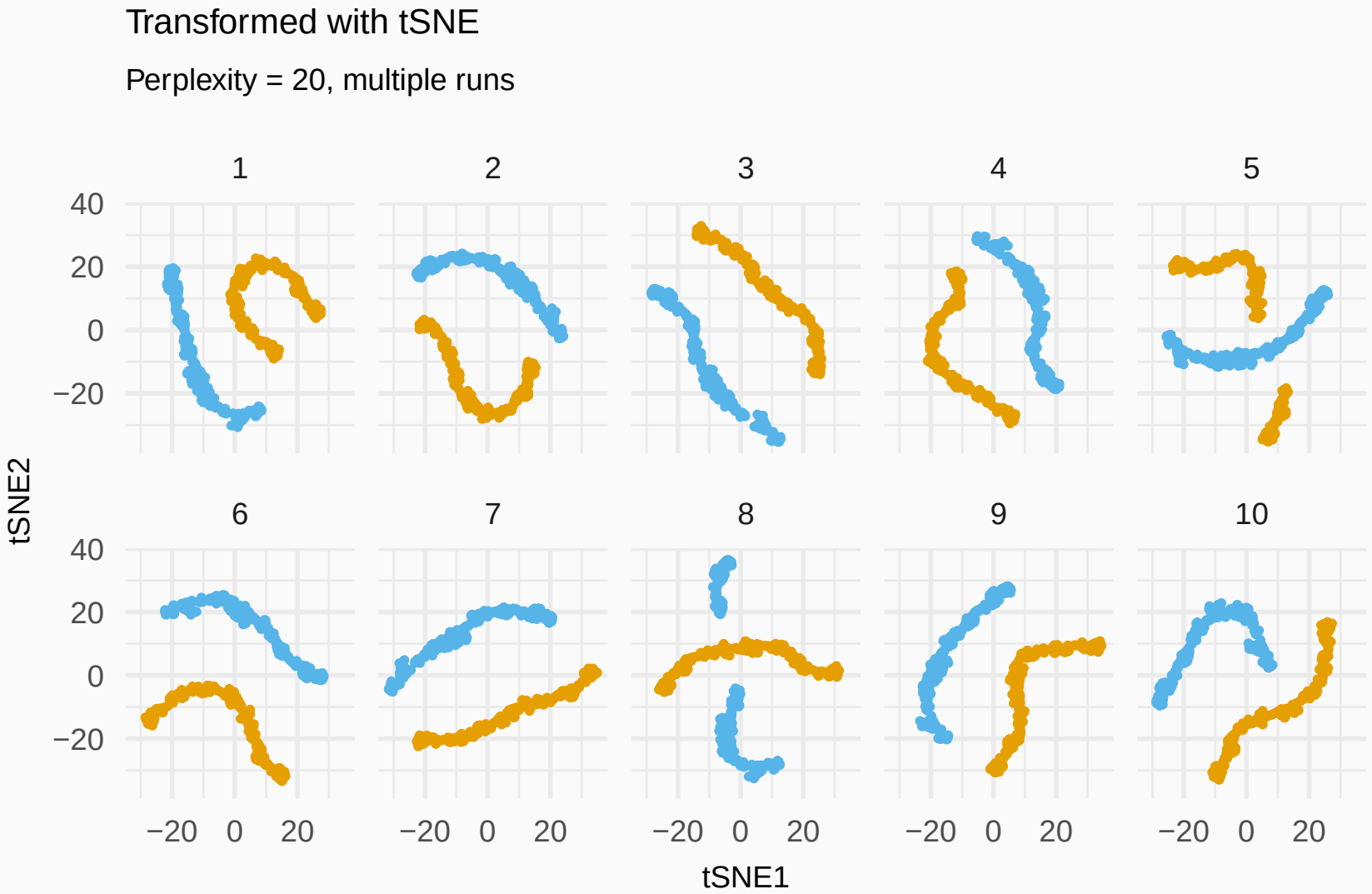
Let's see what tSNE does to our old friend, the moons dataset.



Influence of perplexity on tSNE



tSNE multiple runs



Large scale methods

Overview

- Randomised low-rank SVD (Lecture 15)
- Divide and Conquer (Lecture 15)
- Random Forests with big-n extensions (Lecture 15)
- Leveraging (Lecture 15)

Random projection

There are multiple possibilities how the map f in the **Johnson-Lindenstrauss theorem** can be found.

Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a data matrix and q the target dimension.

- **Gaussian random projection:** Set

$$\Omega_{ij} \sim N\left(0, \frac{1}{q}\right) \quad \text{for} \quad i = 1, \dots, p,\ j = 1, \dots, q$$

- **Sparse random projection:** For a given $s > 0$ set

$$\Omega_{ij} = \sqrt{\frac{s}{q}} \begin{cases} -1 & 1/(2s) \\ 0 & \text{with probability} \quad 1 - 1/s \\ 1 & 1/(2s) \end{cases}$$

  for $i = 1, \dots, p,\ j = 1, \dots, q$ where often $s = 3$ (Achlioptas, 2003) or $s = \sqrt{p}$ (Li et al., 2006)

then $\mathbf{Y} = \mathbf{X}\boldsymbol{\Omega} \in \mathbb{R}^{n \times q}$ is a **random projection** for $\mathbf{X}$.

Randomized low-rank SVD

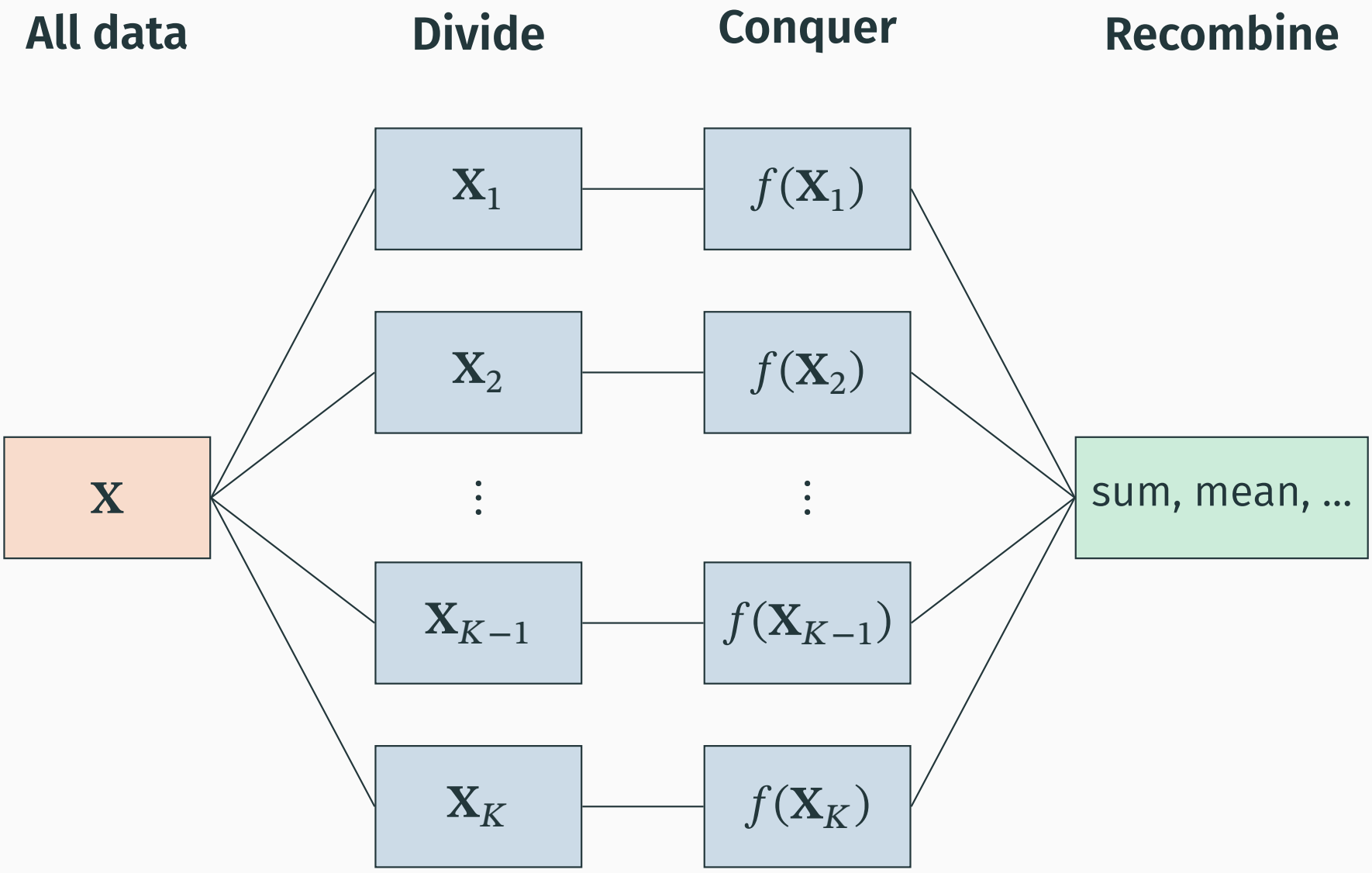
Original goal: Apply SVD in cases where both n and p are large.

Idea: Determine an approximate low-dimensional basis for the range of $\mathbf{X}$ and perform the matrix-factorisation in the low-dimensional space.

- Using a random projection $\mathbf{X} \approx \mathbf{Q}\mathbf{Q}^T\mathbf{X} = \mathbf{Q}\mathbf{T}$
- Note that $\mathbf{T} \in \mathbb{R}^{q \times p}$
- Calculate the SVD of $\underset{q\times p}{\mathbf{T}} = \underset{q\times q}{\mathbf{U}_0} \cdot \underset{q\times q}{\mathbf{D}} \cdot \underset{q\times p}{\mathbf{V}^T}$
- Set $\mathbf{U} = \mathbf{Q}\mathbf{U}_0 \in \mathbb{R}^{n\times q}$, then $\mathbf{X} \approx \mathbf{U}\mathbf{D}\mathbf{V}^T$

The SVD of $\mathbf{X}$ can therefore be found by **random projection** into a q-dimensional subspace of the range of $\mathbf{X}$, performing **SVD in the lower-dimensional subspace** and subsequent **reconstruction** of the vectors into the original space.

Divide and conquer



Random forests for big-n

Instead of the standard RF with normal bootstrapping, multiple strategies can be taken

- **Subsampling (once):** Take a subsample of size m and grow RF from there. Very simple to implement, but difficult to ensure that the subsample is representative.
- **m-out-of-n sampling:** Instead of standard bootstrapping, draw repeatedly m samples and grow a tree on each subsample. Recombine trees in the usual fashion.
- **BLB sampling:** Grow a forest on each subset by repeatedly oversampling to n samples.
- **Divide and Conquer:** Split original data in K parts and grow a random forest on each.

Leverage

Problem: Representativeness

How can we ensure that a subsample is still representative?

We need **additional information** about the samples. Consider the special case of linear regression and $n >> p$.

Recall: For least squares predictions it holds that

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{H}\mathbf{y}$$

with the **hat-matrix** $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$.

Specifically $\hat{y}_i = \sum_{j=1}^{n} H_{ij}y_j$, which means that H_{ii} influences its own fitted values.

Element H_{ii} is called the **leverage** of the observation. Leverage captures if the observation i is close or far from the centre of the data in feature space.

Leveraging

Goal: Subsample the data, but make the **more influential** data points, those with **high leverage**, more likely to be sampled.

Computational approach

- Weight sample i by

$$\pi_i = \frac{H_{ii}}{\sum_{j=1}^{n} H_{jj}}$$

- Draw a weighted subsample of size $m \ll n$
- Use the subsample to solve the regression problem

This procedure is called **Leveraging** (Ma and Sun, 2013).

Outlook

Where to go from here?

- **Support vector machines (SVM):** Chapter 12 in ESL
- **(Gradient) Boosting:** Chapter 10 in ESL
- **Gaussian processes:** Rasmussen and Williams (2006) Gaussian Processes for Machine Learning[1]
- **Streaming/online methods**[2]
- **Neural networks/Deep Learning:** Bishop (2006) Pattern Recognition and Machine Learning
- **Natural Language Processing:** Course[3] by Richard Johansson, CSE, on this topic in the fall
- **Reinforcement Learning:** Sutton and Barto (2015) Reinforcement Learning: An Introduction[4]

[1] http://www.gaussianprocess.org/gpml/chapters/RW.pdf

[2] https://en.wikipedia.org/wiki/Online_machine_learning

[3] https://chalmers.instructure.com/courses/7916

[4] https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf