Lecture 6: Clustering

Felix Held, Mathematical Sciences

MSA220/MVE440 Statistical Learning for Big Data

5th April 2019

CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Projects

- Focus on challenging the algorithms and their assumptions
- Keep your presentations short ($\sim$ 10 min)
- Send in your presentation and code by 10.00 on Friday (all groups)
- There are 30 groups across 3 rooms, i.e.
  - Not every group might get to present (it is not to your disadvantage if you cannot present because there is not enough time)
  - We will group similar topics to allow for better discussion

Importance of standardisation (I)

The overall issue: **Subjectivity vs Objectivity**

(Co-)variance is scale dependent: If we have a sample (size n) of variables x and y, then their empirical covariance is

$$s_{xy} = \frac{1}{n-1}\sum_{l=1}^{n}(x_l - \overline{x})(y_l - \overline{y})$$

If x is scald by a factor c, i.e. $z = c \cdot x$, then

$$\begin{aligned} s_{zy} &= \frac{1}{n-1}\sum_{l=1}^{n}(z_l - \overline{z})(y_l - \overline{y}) \\ &= \frac{1}{n-1}\sum_{l=1}^{n}(c \cdot x_l - c \cdot \overline{x})(y_l - \overline{y}) = c \cdot s_{xy} \end{aligned}$$

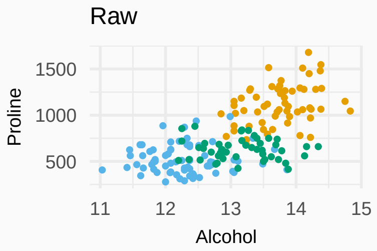
Importance of standardisation (II)

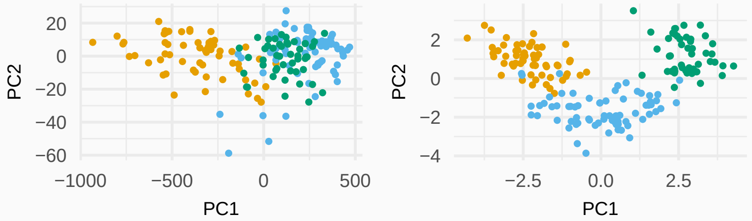
(Co-)variance is scale dependent: $s_{zy} = c \cdot s_{xy}$ where $z = c \cdot x$

- By scaling variables we can therefore make them as large/influential or small/insignificant as we want, which is a very **subjective** process
- By standardising variables we can get of rid of **scaling** and reach an **objective** point-of-view
- **Do we get rid of information?**
  - The **typical range** of a variable is compressed, but if most samples for a variable fall into that range, then it is not very informative after all
  - Real data is not a perfect Gaussian point cloud and therefore there will still be dominating directions after standardisation
  - Outliers will still be outliers

Importance of standardisation (III)

UCI Wine dataset (Three different types of wine with $p = 13$ characteristics)

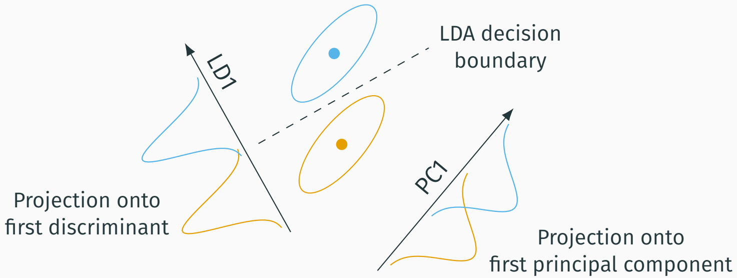




Class-related dimension reduction

Better data projection for classification?

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


LD1
LDA decision boundary
PC1
Projection onto first discriminant
Projection onto first principal component

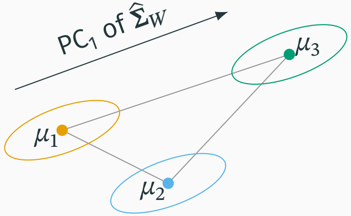
Classification and principal components

In LDA the covariance matrix of the features within each class is $\widehat{\mathbf{\Sigma}}$. Now we will consider the **within-class scatter matrix** $\widehat{\mathbf{\Sigma}}_W = (n-K)\widehat{\mathbf{\Sigma}}$. In addition define

$$\widehat{\mathbf{\Sigma}}_B = \sum_{i=1}^{K} n_i(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})(\boldsymbol{\mu}_i - \overline{\boldsymbol{\mu}})^T, \quad \text{where} \quad \overline{\boldsymbol{\mu}} = \frac{1}{n}\sum_{l=1}^{n} \mathbf{x}_l$$

the **between-class scatter matrix**.

Note: The principal component directions do not take class-labels into account. Classification after projection on these directions can by problematic.



Fisher's Problem

Recall: The variance of the data projected on a direction given by $\mathbf{r}$ can be calculated as $S(\mathbf{r}) = \mathbf{r}^T \widehat{\mathbf{\Sigma}}_W \mathbf{r}$.

In analogy, the variance between class centres along $\mathbf{r}$ is calculated as $\mathbf{r}^T \widehat{\mathbf{\Sigma}}_B \mathbf{r}$.

The goal is to **maximize** variance between class centres while simultaneously **minimizing** variance within each class.

Optimization goal: Maximize over $\mathbf{r}$

$$J(\mathbf{r}) = \frac{\mathbf{r}^T \widehat{\mathbf{\Sigma}}_B \mathbf{r}}{\mathbf{r}^T \widehat{\mathbf{\Sigma}}_W \mathbf{r}} \quad \text{subject to} \quad ||\mathbf{r}|| = 1$$

which is a more general form of a **Rayleigh Quotient** and is called **Fisher's problem**.

Solving Fisher's Problem

Note: There are maximum $K - 1$ solutions $\mathbf{r}_j$ to Fisher's problem (because $\widehat{\boldsymbol{\Sigma}}_B$ has rank $\leq K - 1$).

Computation of solutions:

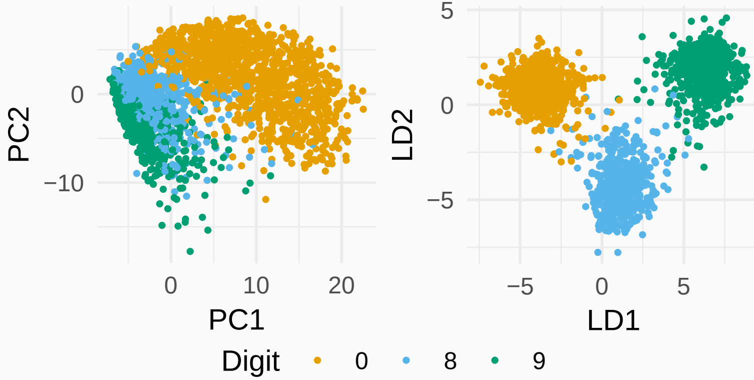
1. Compute the **eigen-decomposition** (the matrix is real and symmetric)
$$\widehat{\boldsymbol{\Sigma}}_W^{-1/2}\widehat{\boldsymbol{\Sigma}}_B\widehat{\boldsymbol{\Sigma}}_W^{-1/2} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$
where $\mathbf{V} \in \mathbb{R}^{p\times p}$ orthogonal and $\mathbf{D} \in \mathbb{R}^{p\times p}$ diagonal.
2. Set $\mathbf{R} = \widehat{\boldsymbol{\Sigma}}_W^{-1/2}\mathbf{V}$. The columns of $\mathbf{R}$ solve Fisher's problem (as with PCA the j-th solution maximizes Fisher's problem on the orthogonal complement of the first $j - 1$ solutions)

Discriminant Variables and Reduced-rank LDA

- The vectors $\mathbf{r}_j$ determined by solving Fisher's problem can be used like PCA, but are **aware of class labels** and give the **optimal separation of projected class centroids**
- Projecting the data onto the j-th solution gives the **j-th discriminant variable** $\mathbf{r}_j^T\mathbf{x}$
- Using only the $m < K - 1$ first is called **reduced-rank LDA**

Reduced-rank LDA: Example

- Consider digits 0, 8 and 9 in the MNIST digit dataset.
- Compare PCA and discriminant variable projections onto the first two components.
- For technical reasons features constant within at least one class had to be excluded before running LDA.



Cross-validation and dimension reduction

Caution when using a dimension reduction technique like PCA or reduced-rank LDA, together with cross-validation:

- PCA is a **class-unrelated** technique for dimension reduction
- Whereas LDA is a **class-related** technique for dimension reduction
- Any transformation done to all samples **before application of cross validation** has to be class-**un**related. Otherwise the projected data contains information about the test data even in its training data
- **However:** To avoid potential confusion, best to perform all data preparation on the training data alone and then apply the same transformations to the test data

Clustering

Classification without classes

In **classification** the main idea was to determine

$$p(i|\mathbf{x}) \quad \text{or} \quad p(\mathbf{x}, i) = p(\mathbf{x}|i)p(i)$$

through model approximations (LDA, logistic regression), rules/partitioning (CART, random forests) or directly from data (kNN).

What if we do not have any classes? Clustering

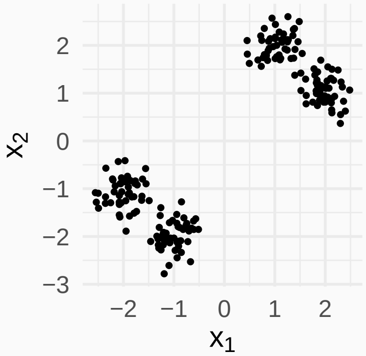
Goals

- Find groups in data
- Summarize high-dimensional data
- Data exploration

Clustering

Clustering is a harder problem than classification

- What is a cluster?
- How many clusters are there?
- How do we find them? Can they have shape?



We need to able to measure **dissimilarity** between features to determine which samples/objects are close together or far apart.

Note: In clustering *classes* are often called **labels** and *features* are **attributes**

Dissimilarity measures

A **dissimilarity measure** for features x_1, x_2 is a function such that

$$d(x_1, x_2) \geq 0 \quad \text{and} \quad d(x_1, x_2) = d(x_2, x_1)$$

Dissimilarity across all features can be defined as

$$D(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{p} d_j(x_1^{(j)}, x_2^{(j)})$$

Typical examples

- For quantitative features: ℓ_1 or ℓ_2 norm, correlation between whole feature vectors, …
- For categorical variables: Loss matrix $\mathbf{L} \in \mathbb{R}^{K \times K}$ such that $\mathbf{L}_{rs} = \mathbf{L}_{sr}$, $\mathbf{L}_{rr} = 0$ and $\mathbf{L}_{rs} \geq 0$. Then $d(x_1, x_2) = \mathbf{L}_{x_1 x_2}$

Challenges in Clustering

Two main challenges

1. How many clusters are there?
2. Given a number of clusters, how do we find them?

Focus on Challenge 2 first.

Idea: Partition the observations into K groups/clusters so that **pairwise dissimilarities within groups** are **smaller than between groups.**

Note: A partition of the observations is called a **clustering rule** $C(\mathbf{x}) = i$

Combinatorial Clustering (I)

Similar to Fisher's problem we are looking at **point scatter**.

Total amount of dissimilarity across all observations

$$T = \underbrace{\sum_{l=1}^{n} \sum_{m<l} D(\mathbf{x}_l, \mathbf{x}_m)}_{\text{Total point scatter}}$$

$$= \sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \left(\sum_{\substack{m<l \\ C(\mathbf{x}_m)=i}} D(\mathbf{x}_l, \mathbf{x}_m) + \sum_{\substack{m<l \\ C(\mathbf{x}_m)\neq i}} D(\mathbf{x}_l, \mathbf{x}_m) \right)$$

$$= \underbrace{\sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \sum_{\substack{m<l \\ C(\mathbf{x}_m)=i}} D(\mathbf{x}_l, \mathbf{x}_m)}_{\substack{=:W(C) \\ \text{Within cluster point scatter}}} + \underbrace{\sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \sum_{\substack{m<l \\ C(\mathbf{x}_m)\neq i}} D(\mathbf{x}_l, \mathbf{x}_m)}_{\substack{=:B(C) \\ \text{Between cluster point scatter}}}$$

Combinatorial Clustering (II)

Note that T does not depend on the clustering. Therefore

$$W(C) = T - B(C)$$

and **minimizing within cluster point scatter** is equivalent to **maximizing between cluster point scatter**.

As in the case of decision trees/CART looking at all possible partitions and finding the global minimum of $W(C)$ is too computational expensive.

Use **greedy algorithms** to find local minima.

Approximations to Combinatorical Clustering (I)

Consider the **special case** $D(\mathbf{x}_l, \mathbf{x}_m) = \|\mathbf{x}_l - \mathbf{x}_m\|^2$ then

$$W(C) = \sum_{i=1}^{K} \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \sum_{\substack{m<l \\ C(\mathbf{x}_m)=i}} \|\mathbf{x}_l - \mathbf{x}_m\|^2$$

$$= \sum_{i=1}^{K} N_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \|\mathbf{x}_l - \mathbf{m}_i\|^2$$

where

$$N_i = \sum_{l=1}^{n} \mathbb{1}(C(\mathbf{x}_l) = i) \quad \text{and} \quad \mathbf{m}_i = \frac{1}{N_i} \sum_{C(\mathbf{x}_l)=i} \mathbf{x}_l$$

Approximations to Combinatorical Clustering (II)

The goal now is to solve

$$\arg\min_{C} \sum_{i=1}^{K} N_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \|\mathbf{x}_l - \mathbf{m}_i(C)\|^2$$

which still requires to visit all possible partitions.

Observation: For a fixed clustering rule C it holds that

$$\mathbf{m}_i(C) = \arg\min_{\mathbf{m}} \sum_{C(\mathbf{x}_l)=i} \|\mathbf{x}_l - \mathbf{m}\|^2$$

Approximative solution: Consider the larger problem

$$\arg\min_{\substack{C \\ m_i \text{ for } 1\leq i\leq K}} \sum_{i=1}^{K} N_i \sum_{\substack{l=1 \\ C(\mathbf{x}_l)=i}}^{n} \|\mathbf{x}_l - \mathbf{m}_i\|^2$$

k-means

This approximation can be solved iteratively for the clustering C and the cluster centres. This is called the **k-means** algorithm.

Computational procedure:

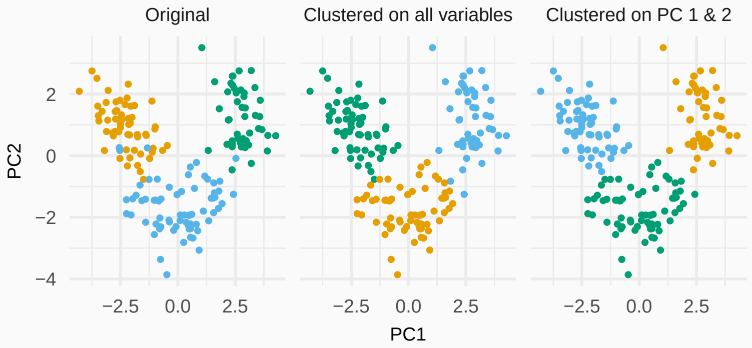
1. **Initialize:** Randomly choose K observations as cluster centres $\mathbf{m}_i$ and set $J_{\max}$
2. For steps $j = 1, \dots, J_{\max}$
    2.1 **Cluster allocation:** $C(\mathbf{x}_l) = \underset{1 \le i \le K}{\arg\min} \|\mathbf{x} - \mathbf{m}_i\|^2$
    2.2 **Cluster centre update:** $\mathbf{m}_i = \frac{1}{N_i} \sum_{C(\mathbf{x}_l) = i} \mathbf{x}_l$
    2.3 Stop if clustering C did not change

Notes on k-means

- **Dependence on initial selection:** Run repeatedly to
- Since k-means uses the ℓ_2 norm it has all the typical problems (**sensitive to outliers and noise**)
- **Clusters tend to be circular:** k-means looks in a circular fashion around each cluster centre and assigns an observation to the closest centre
- **Always finds K clusters** (not unique to k-means)

Using k-means on the wine dataset

UCI Wine dataset: $K = 3$ classes. Let's see if k-means recovers the classes given only the features/attributes.



Note: k-means (and all clustering algorithms) are very sensitive to certain geometries

Take-home message

- Standardisation is important to remove subjective scaling from data
- Reduced-rank LDA can lead to an optimal dimension reduction with regards to class separation
- Clustering is a more challenging problem than classification and needs to answer two questions:
  - How many clusters?
  - What is a cluster?