Class 6: Dimension reduction (part 2) and clustering

Andrew Parnell
andrew.parnell@mu.ie

PRESS RECORD

https://andrewcparnell.github.io/intermediate_ML

Learning outcomes

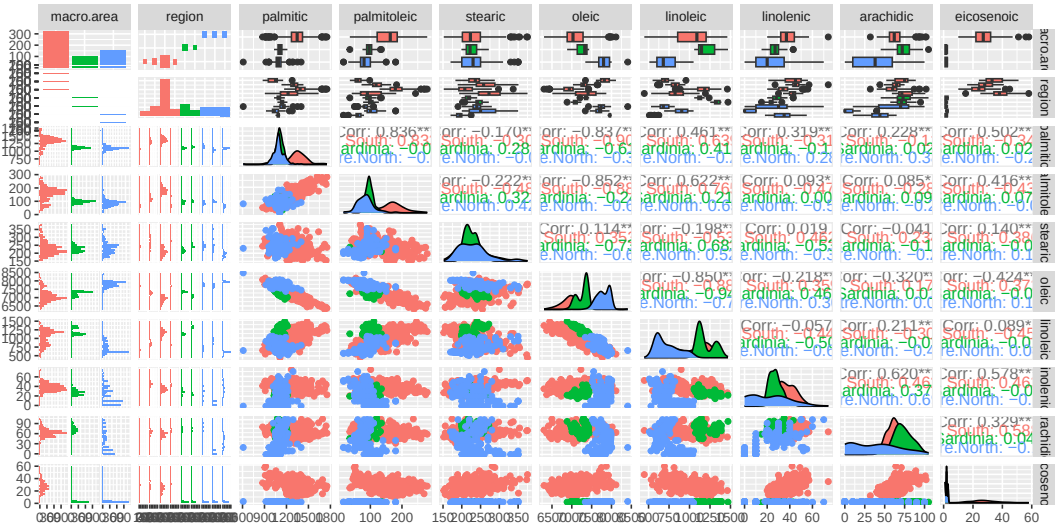
- Learn about 2 more state of the art dimension reduction techniques: Uniform Manifold Approximation and Projection (UMAP), and t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Learn about 2 state of the art clustering techniques model-based clustering (MBC), and spectral clustering
- For each, learn how it works, how to fit it, and see an example
- Be able to change the example code to fit your particular data sets

A new data set

```
library(pdfCluster)
data(oliveoil)
str(oliveoil)
```

```
## 'data.frame':    572 obs. of  10 variables:
##  $ macro.area : Factor w/ 3 levels "South","Sardinia",..: 1 1 1 1 1 1 1
##  $ region     : Factor w/ 9 levels "Apulia.north",..: 1 1 1 1 1 1 1 1 1
##  $ palmitic   : int  1075 1088 911 966 1051 911 922 1100 1082 1037 ...
##  $ palmitoleic: int  75 73 54 57 67 49 66 61 60 55 ...
##  $ stearic    : int  226 224 246 240 259 268 264 235 239 213 ...
##  $ oleic      : int  7823 7709 8113 7952 7771 7924 7990 7728 7745 7944
##  $ linoleic   : int  672 781 549 619 672 678 618 734 709 633 ...
##  $ linolenic  : int  36 31 31 50 50 51 49 39 46 26 ...
##  $ arachidic  : int  60 61 63 78 80 70 56 64 83 52 ...
##  $ eicosenoic : int  29 29 29 35 46 44 29 35 33 30 ...
```

And in pictures



Technique number 1: UMAP

- A nonlinear dimension reduction technique that approximates the manifold structure of the data
- Aims to preserve both local and global properties, unlike other techniques that focus primarily on one aspect
- Computationally efficient and applicable to various domains, including visualization, clustering, and general non-linear dimension reduction
- Very commonly used in genetics

How does UMAP work?

Approximately:

1. Neighbors: UMAP starts by looking at each data point and identifying its closest neighbors, forming a network of connections like a web.
2. Building a Map: It then constructs a simplified map that represents the connections in a lower-dimensional space (like going from 3D to 2D). The goal is to make the distances between points on this new map as similar as possible to the distances in the original high-dimensional space.
3. Refining the Map: Through an iterative process, UMAP fine-tunes the map, adjusting and shifting points to ensure that close neighbors in the original space remain close in the new space, and far-away neighbors stay far away.
4. Result: The final result is a simpler representation of the data that maintains much of the original structure. This allows us to visually understand complex relationships that might have been hard to see in the original high-dimensional space.

UMAP in a bit more detail

- The original paper McInnes et al 2018 is very mathematical so only read if you really want to get into the detail
- One of the core components of UMAP is the use of embeddings, just like we have met in Transformers and autoencoders
- It minimises a loss function based on the distance metric to compare the higher and lower dimensional projections of the data to make them match as close as possible
- There is still a PCA included in the background!

Using UMAP

- Very straightforward to run in R

```
library(umap)
umap_result <- umap(my_data)
```

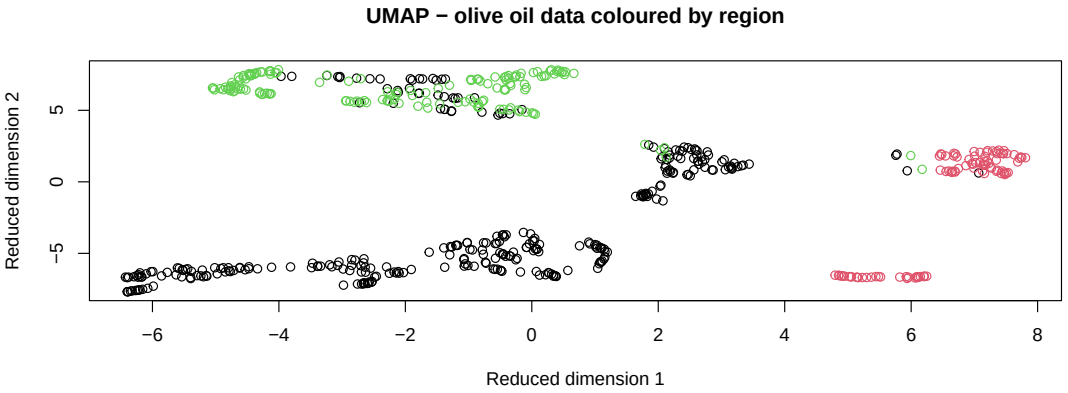
- The defaults are pretty good and you can change the settings with:

```
umap_result <- umap(my_data, n_components = 3)
```

. . . or provide a full list of things you want to change (neighbour structures, distance metrics, etc)

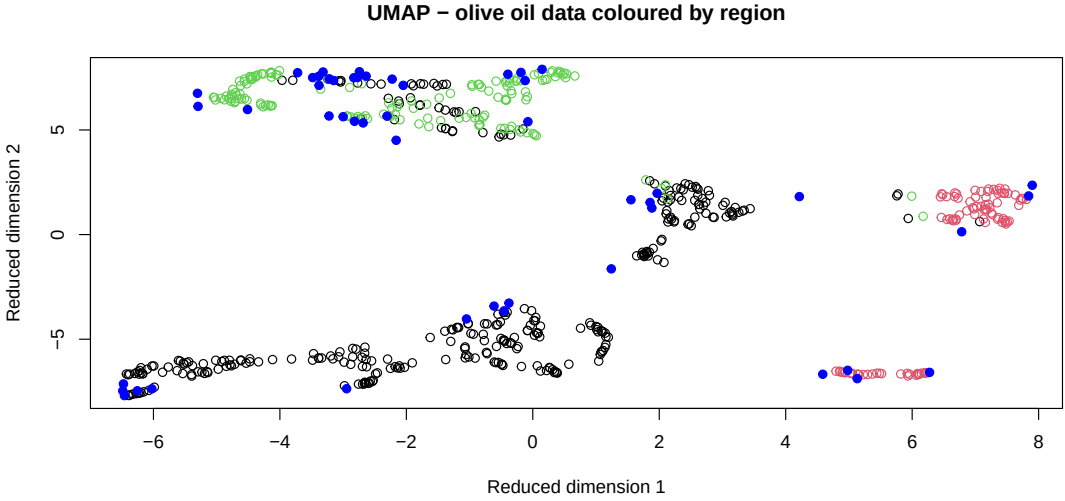
UMAP output

```
umap_result <- umap(olive1)
plot(umap_result$layout, col = oliveoil$macro.area,
     xlab = 'Reduced dimension 1', ylab = 'Reduced dimension 2',
     main = 'UMAP - olive oil data coloured by region')
```



Projecting new values

- The `umap` package comes with a `predict` function
- If we have new data that arise we can see where they lie in the projected space



Technique number 2: *t*-SNE

- t-Distributed Stochastic Neighbor Embedding (t-SNE) is a popular technique to perform dimension reduction
- It's very widely used though perhaps has fallen out of favour compared to UMAP in recent years (particularly in genetics)
- The Rtsne package allows to fit these models pretty quickly though it does get slower for large data sets
- It has one key parameter called 'complexity' which controls much of how the algorithm works

How does it work?

1. Neighborhoods: t-SNE starts by figuring out the close friends (or neighbors) of each data point
2. Creating a new map. It now creates a lower dimensional projection of all the data points but puts them in random positions
3. Adjusting positions: It then moves data points around in the lower dimensional space to try to keep the same pattern. Just like UMAP - it's looking for the low dimensional space to match the original data
4. Perplexity. This key parameter controls how closely you bring data point together. The t-distribution underlying t-SNE has long details so can push data points further apart than other dimension reduction techniques
5. Final map: it returns the optimised low dimensional projection subject to the perplexity parameter in d-dimensional space (you choose d)

How does it work in a bit more detail?

1. In the high-dimensional space, t-SNE computes pairwise similarities using conditional probabilities, where $p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}$. The parameter perplexity, related to σ_i, balances attention between local and global aspects, guiding how many effective neighbors each point has.

2. It next defines a similar probability distribution in the low-dimensional space using a Student's t-distribution: $q_{ij} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1+||y_k - y_l||^2)^{-1}}$. The algorithm minimizes the Kullback-Leibler divergence between the high-dimensional and low-dimensional distributions, $D_{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$, using gradient descent.

3. The final result is a 2D (or higher) map that preserves the structure of the data. Points close in the high-dimensional space remain close in the low-dimensional space, providing an intuitive visualization that reflects original relationships and clusters.

Using t-SNE in R

- Again just one command

```
library(Rtsne)
Rtsne(data, dims = 2, perplexity = perplexity_value)
```

- The dims and perplexity values are optional (defaults 2 and 30)
- Outputs a matrix Y of dimension $(N, dims)$ which contains the projection

Results

```r
# Load necessary libraries
library(pdfCluster)
library(Rtsne)
library(fpc) # For cluster.stats
# Load the olive oil data
data(oliveoil)

# Transform the data first since it is proportions
# See help(oliveoil) and help(pdfCluster)
olive1 <- 1 + oliveoil[, 3:10]
margin <- apply(data.matrix(olive1),1,sum)
olive1 <- olive1/margin

tnse_model <- Rtsne(olive1)

plot(tnse_model$Y,
     col = oliveoil$macro.area,
```

Notes on t-SNE

- You can see in the previous plot that it actually separates out two groups that should be together (though recall that it doesn't get the labels when learning the plot)
- Another disadvantage is that, because of the way it's fitted, t-SNE doesn't have a predict function. You either need to re-run the algorithm with the test data included
- t-SNE doesn't have the theoretical background that UMAP does, it's a bit more of an ad-hoc approach

Comparison of techniques

- UMAP vs t-SNE: UMAP preserves more of the global structure than t-SNE, while also being faster and more scalable
- UMAP vs PCA: Unlike linear methods like PCA, UMAP captures complex nonlinear relationships in the data
- Flexibility and Applications: All of these can be further used for clustering, feature selection, anomaly detection, etc as in Class 5

Clustering

- Rather than reducing the dimensionality of the observations, group similar observations together in clusters
- Lots of methods: we just cover 2: model-based clustering (MBC) and spectral clustering
- As before, we cover how each one works and how to implement it in R
- There are some other fancy new clustering methods (e.g. Deep Embedded Clustering) that we don't cover

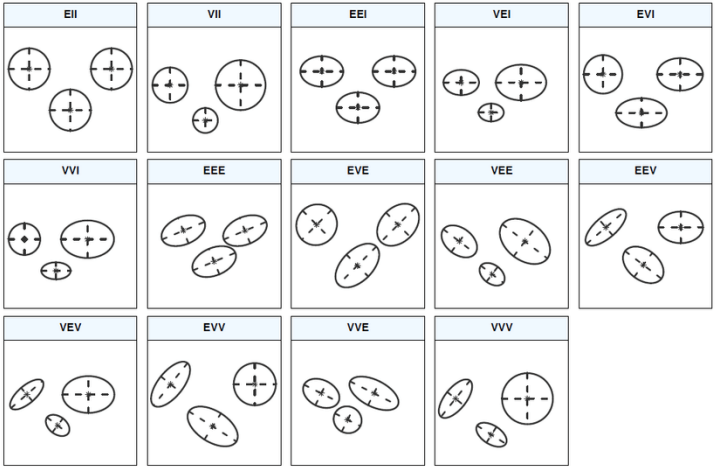
Model-Based Clustering - Introduction

- Model-based clustering assumes that data is generated from a mixture of several statistical distributions, usually normal distributions.
- The most popular package is `Mclust` which is very fast and flexible
- The most difficult problem is usually choosing the number of clusters

How does it work? (normal distribution version)

- MBC assumes that each observation comes from a multivariate normal distribution, potentially in high dimensions
- Using a given number of clusters, it uses an algorithm called Expectation-Maximisation (EM) to allocate each observation to its closes normal distribution, while estimate the means and variance matrices of each one
- The user also has a choice as to how the normal distributions behave in terms of their shape, known as `modelNames` in `mclust`
- Because it is a statistical model, each observation has a probability of being in a certain cluster, so we have full access to the uncertainty in the model

Pictures of different types of `mclust` models



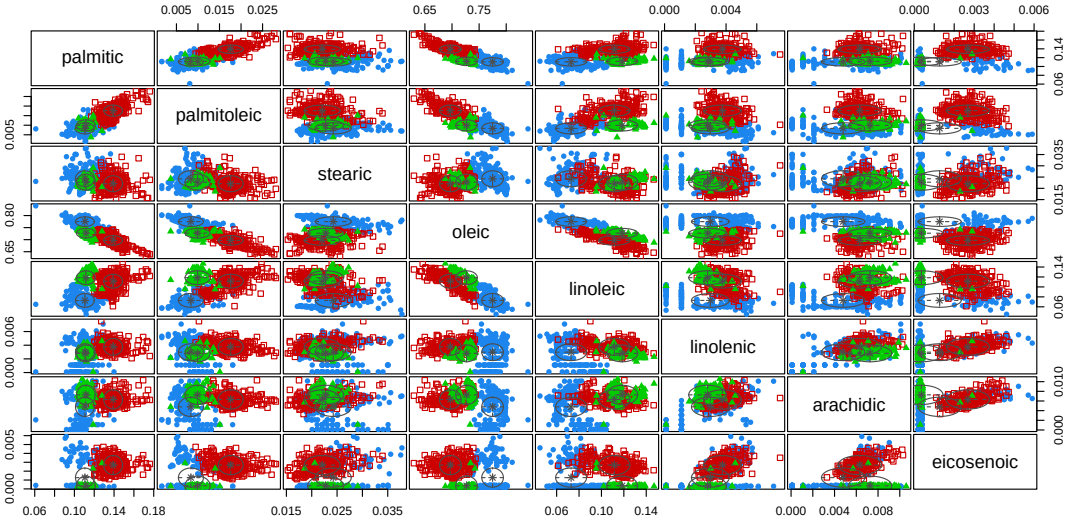
From https://www.researchgate.net/figure/Cluster-shapes-of-the-14-Gaussian-models-implemented-in-Mclust-package-in-case-of_fig2_327423776

How does it work in more detail?

1. Creates initial values of the cluster membership probabilities and the means and variances of the normal distributions. The model will run faster with better initial values but these can be hard to choose
2. Runs the EM algorithm to refine all of these values. The E-step calculates the expectation of the model given the current parameter estimates. The M-step then updates the parameter estimates by minimising the loss function
3. Once converged, the model provides the best estimates of these values

Fitting MBC in R on the olive oil data

```
mbc <- Mclust(olive1, G = 3, modelNames = "EEI")
plot(mbc, what = 'classification')
```

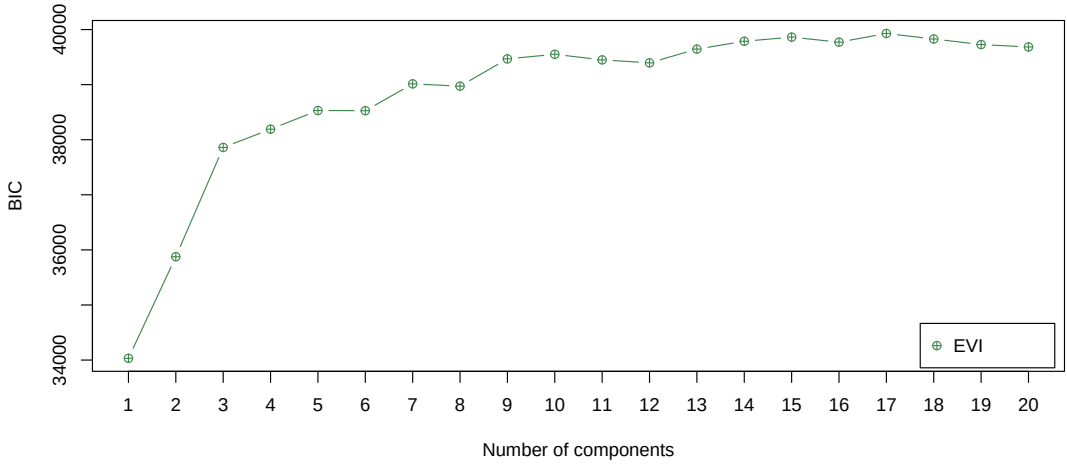


Choosing the number of clusters

- There are lots of methods for comparing clustering solutions (see e.g. associated R scripts using silhouette scores)
- `mclust` uses a specific score called the Bayesian Information Criterion (BIC) which measures comparatively how well the model performs
- Usually we choose a variety of covariance types and a variety of cluster values and choose the one with the (highest) BIC

Comparing clusters

```
all_mods <- Mclust(olive1, G = 1:20, modelNames = "EVI")
plot(all_mods, what = 'BIC')
```



Final notes on Mclust

- `mclust` focuses on normal distributions but you can have clusters of other distributions too
- There are also packages that extend `mclust` to deal with missing data, time series data, etc
- Choosing the number of clusters remains a problem: a more complex method called 'Dirichlet Processes' aims to solve this problem (not covered)

Spectral Clustering - Introduction

- Lots of R packages perform spectral clustering (SPC), we will use the `kernlab` package but others may be better
- It's quite a simple algorithm and can produce some cool results on interesting data sets
- It's especially good at irregularly shaped clusters - e.g. not circles or ovals

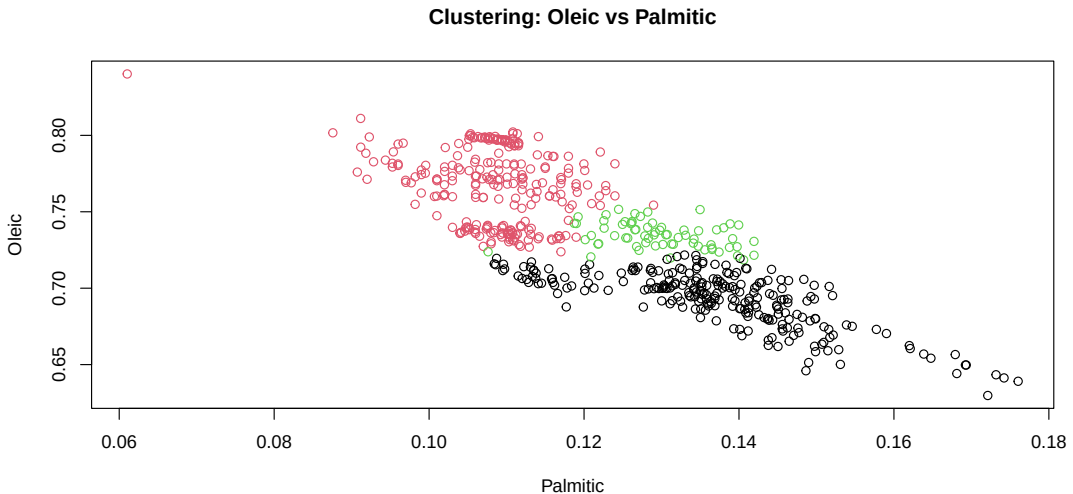
SPC - how does it work?

1. Just like most other dimension reduction and clustering techniques, it starts with a similarity (or distance) matrix W of the observations
2. It then builds a **Laplacian Matrix** to capture the matrix's structure. A common one is the un-normalised Laplacian, given by $L = D - W$, where D is the degree matrix (a diagonal matrix where the diagonal elements are the sum of the corresponding row in the similarity matrix)
3. It then does PCA on the matrix L and selects a number of the components according to the number of clusters
4. It then performs a standard cluster analysis (e.g. k-means) in the new space of principal components

Using SPC

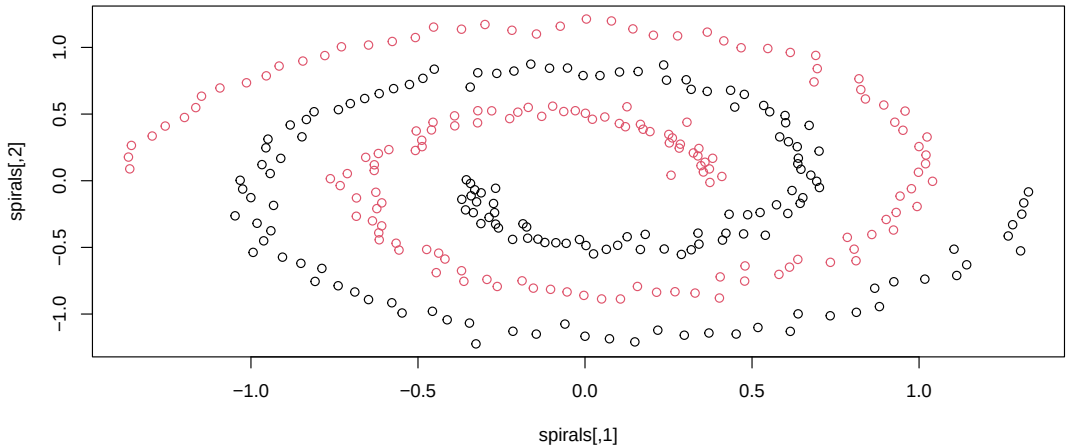
Again, just a few lines in R

```
clustering <- specc(as.matrix(olive1), centers = 3)
```



A cool example: spirals

```
data(spirals)
sc <- specc(spirals, centers=2)
plot(spirals, col=sc)
```



Final notes on clustering and dimension reduction

- You can combine many of these techniques, use e.g. UMAP for dimension reduction and visualisation and `mclust` for clustering data
- A big issue in many of these methods is computation. They can get very slow and a lot of work is being devoted to more computationally efficient methods
- Key issues are cluster shape and number of clusters. Choosing these is hard and not easy to provide general rules
- Lots of R packages now that perform these methods

Summary

- UMAP and *t*-SNE both really cool dimension reduction methods; try both and see which works best for your data
- Model-based clustering very principled and based on sound statistical foundations
- Spectral clustering useful for non-regular cluster shapes