MSA220 - Statistical Learning for Big Data

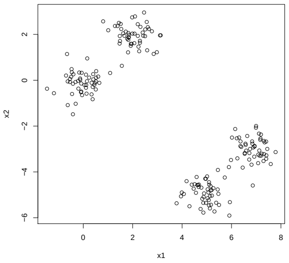
Lecture 13

Rebecka Jörnsten

Mathematical Sciences
University of Gothenburg and Chalmers University of Technology

CLUSTERING

Explorative analysis - finding groups in data.
This is a more difficult task than classification since the goal is rather subjective - what is group?



Are there 2 or 4 clusters?

Clustering

What defines a group is up to you to choose, e.g. by defining an object-object similarity measure or distance.
The most commonly used distance measure is euclidean distance.
However, other distances may be more appropriate to use for some data sets, e.g. matching-metrics for categorical data or correlation-based similarities for curve data or when relative differences between features are more interesting than absolute levels.

Clustering

Main traditional approaches

1. Partitioning methods: kmeans, PAM
2. bottom-up methods: hierarchical clustering
3. Model-based methods/density based methods

How to generate a partition

kmeans is a very popular method and has been around for a long time. It is very simple and fast.

1. Pick K observations at random to be the cluster representatives or *centroids*, $\mu_k, k = 1, \cdots, K$.
2. Allocate observations i to the cluster whose centroid it is closest to
$$C(i) = \arg\min_k d(x_i, \mu_k),$$
where $d(x_i, \mu_k)$ is the distance between observation location x_i and centroid μ_k.
3. Update the centroids as
$$\mu_k = \sum_{C(i)=k} x_i / N_k, \quad N_k = \sum_{i=1}^{n} 1\{C(i) = k\}$$
4. Iterate until convergence (usually very fast).

Note, you may have to run the algorithm a couple of times to ensure you have converged to a local optimum due to poor choice of initial centroids.

Hierarchical clustering

Hierarchical clustering is very popular since it is simple, intuitive and comes with a nice graphical display. Like PAM, it takes pairwise distances as input which makes it rather flexible.
In contrast to PAM and kmeans it constructs clusters "bottom-up", i.e. building clusters by joining observations together as opposed to splitting the data into groups ("top-down").

Hierarchical clustering

1. Start with all the observations as their own clusters, $g_1, g_2, \cdots, g_n$, each cluster of size 1.
2. Join the pair of clusters g_i and g_j that are the closest together
3. Keep on joining clusters pairs until all observations are in one big clusters of size n.

Hierarchical clustering

Step 2 involves some subjective choices:
what is close? that is, what kind of distance metric do you want to use?
what is meant by clusters being close? that is, how do we combine information about observation pairwise distances into a group-level distance?

Linkage

Cluster-cluster distance is called *linkage*

- **average linkage** is the most commonly used. The distance between clusters g and h is computed as

$$d_{gh} = \sum_{i:C(i)=g, j:C(j)=h} d_{ij} / \sum_{i:C(i)=g, j:C(j)=h} 1$$

  The average similarity between all pairs in the two different clusters is encouraged.

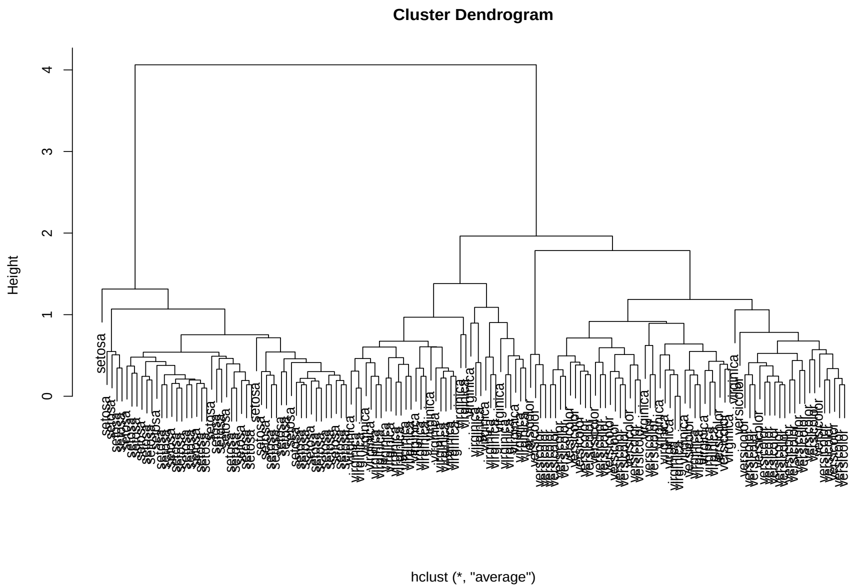
Dendrogram

Hierarchical clustering is graphically summarized with the *dendrogram*. This depicts the iterative procedure of joining clusters.

The dendrogram looks a bit like a CART tree but the meaning is different. You read the dendrogram from the bottom-up, this is how the clusters are formed. The length of the branches in the dendrogram represents the cluster-cluster distances. A really long branch indicates that the within-cluster distances were increased a lot by joining the cluster at the bottom of the branch with the other cluster at the top of the branch.

The dendrogram can therefore suggest how many clusters you should form from your data. Long branches can be cut to form distinct group that have small within-cluster distance and is well separated from the rest of the observations.

Hierarchical clustering



Hierarchical clustering

The average linkage identifies two groups of irises (long branches in the dendrogram): setosa and the versicolor/virginica. The latter group is very mixed up.

Modelbased clustering

So far we have looked at nonparametric cluster methods - clusters are defined through a distance metric and a construction algorithm/criterion. We have noted that clustering is a difficult problem because these choices are subjective.
Parametric, or modelbased clustering, takes clustering into a familiar statistical modeling framework where we can say something about the goodness-of-fit of clusters. It is a statistical problem that can be objectively analyzed BUT of course relies on a modeling assumption that is a subjective choice nonetheless.

Modelbased clustering

The multivariate normal assumption sounds a bit like discriminant analysis. The difference here is that we don't know the class label! We have already looked at a similar problem when we talked about the mixture discriminant analysis method where the class labels were known but the component-labels within each class was not. There we solved the problem with the EM-algorithm, and that is what we do here as well and I will give you a bit more detailed info as well.

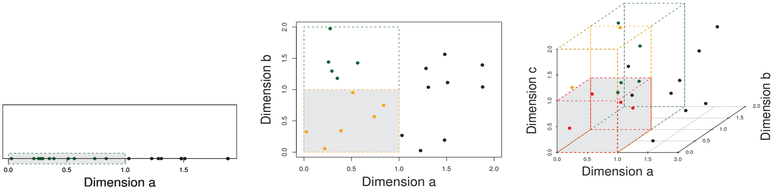
EM-ITERATIONS

- If we knew the labels, we could estimate the parameters of each cluster easily just like we did in discriminant analysis.
- If we knew the model parameters, we could allocate observations to each cluster using the posterior probability approach, just like in discriminant analysis.

This iterative process is the EM approach to model fitting and is a method used to solve complex estimation problems that would be easy to solve with some additional information (as done in each step or the iterative procedure).

High-dimensional clustering

What goes wrong when the data is high-dimensional?

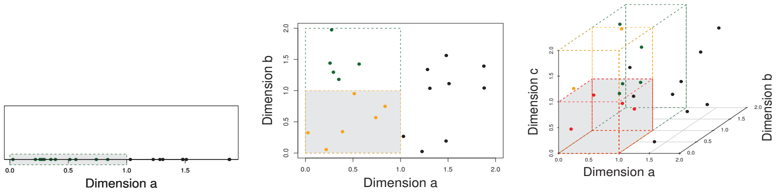


(a) 11 Objects in One Unit Bin (b) 6 Objects in One Unit Bin (c) 4 Objects in One Unit Bin

Figure 1: The *curse of dimensionality*. Data in only one dimension is relatively tightly packed. Adding a dimension stretches the points across that dimension, pushing them further apart. Additional dimensions spreads the data even further making high dimensional data extremely sparse.

High-dimensional clustering

The notion of similar and dissimilar brakes down - everyone is far apart in high-dimensional space!
Clustering is all about distances - and the concept of relative distance brakes down.



(a) 11 Objects in One Unit Bin (b) 6 Objects in One Unit Bin (c) 4 Objects in One Unit Bin

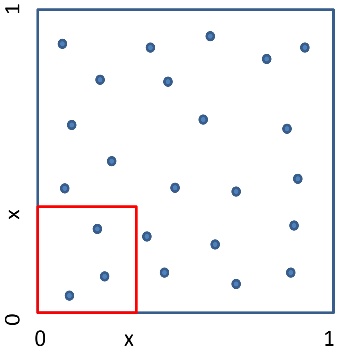
Figure 1: The *curse of dimensionality*. Data in only one dimension is relatively tightly packed. Adding a dimension stretches the points across that dimension, pushing them further apart. Additional dimensions spreads the data even further making high dimensional data extremely sparse.

Dimension reduction

When you have many features, perhaps even more features than samples, some clustering and classification methods don't work.

- Example: LDA doesn't work when $p > n$ since you can't take the inverse of the within-class covariance
  - You can "fix" these either by reducing the dimensionality of the problem before training your classifiers
  - or you can fix the numerical problem directly, e.g. using the inverse of $\Sigma + \lambda\Lambda$ as in penalized discriminant analysis
- Curse of dimensionality. Many methods are based on some notion of distance and obtaining local estimates of class probabilities or densities. In higher dimensions the concept of "closeness" breaks down - everything is far apart..

Curse of dimensionality



Let's say data are uniformly distributed in p-dimensional space in a hypercube $[0, 1]^p$ ($p = 2$ illustrated above). The cube $[0, x]^p$ captures r% of the data. Turning this around, let's ask, as a function of p, what x has to be to capture r % of the data.

Curse of dimensionality

Since we're working with the uniform distribution, and assuming feature independence we have

$$r = P(X_1 \leq x, \cdots, X_p \leq x) = x^p, \quad x = r^{1/p}$$

	$r = 1\%$	$r = 10\%$
$p = 2$	$x = .1$	$x = .32$
$p = 3$	$x = .215$	$x = .46$
$p = 10$	$x = .32$	$x = .56$
$p = 100$	$x = .63$	$x = .79$

The table indicates that for large p to capture a quite small percentage of probability mass you actually have to look almost in the entire space! So nothing is close! If nothing is close, what does "local" mean when you try to estimate local class posterior probabilities.

Curse of dimensionality

Another way of saying this: If there are n points in 1D space, the density is roughly $1/n$. To get that same density in p-dimensional space you would need n^p data points! This number grows fast with p....

Dimension reduction

This is actually a very complicated task.
In some sense, we would need to know the answer (how the method would perform) in order to know how best to reduce the dimensionality. Otherwise we run the risk of removing features in a way that is very suboptimal for the method we wish to apply to the data for classification or clustering.
Dimension reduction for classification is "easier" than for clustering since at least we have something we can compute and measure for each feature, e.g. how much the feature varies between different classes (ANOVA or Between-to-within variance filtering).

Filtering

Pre-processing the data to remove some features prior to classication is sometimes called *filtering*. Examples of filtering procedures

- Between-to-Within filtering
- Statistical testing, multiple testing, ranking
- Principal component analysis

FILTERING

Filtering feature for clustering is more difficult. Without seeing the whole, how can you know if a feature is "helpful" for clustering or not?

- Max-variance filtering selects the features that varies the most across observations. The idea is that such features have variance *because* it's along these directions that clusters are placed and therefore observations are spread out.
- Principal component analysis, which essentially is a more structured Max-variance filtering.
- Max-variance is not necessarily helping with cluster detection. You can also look for multi-modality in the distribution for each feature (e.g. via QQplots or mixture modeling on one feature at a time).

High-dimensional clustering

What to do?

- Feature selection
- Feature transformation

Feature selection

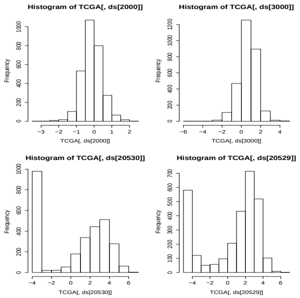
- This is a much easier task for high-dimensional classification
- For example, run ANOVA on each feature and choose the most significant features to train the classifier on
- How can we screen for clustering strength when we don't know the clusters?

Feature selection

- Take the most variable features.
- The idea is that large spread is due to cluster separation
- Caution: this is a bad idea if features are measured at different scales!

Feature selection

- An alternative is to think that a cluster feature should have a clear multi-modal distribution where each "hump" in the distribution corresponds to a cluster
- Screen features by testing for unimodality (Hartigan's dip-test).
- Keep features with the largest test statistic against unimodality



Feature transformation

- We can also transform the data - projecting onto a lower-dimensional space
- We want to ensure we retain as much information as possible
- PCA to the rescue!
- Keep principal components corresponding to the largest eigenvalues

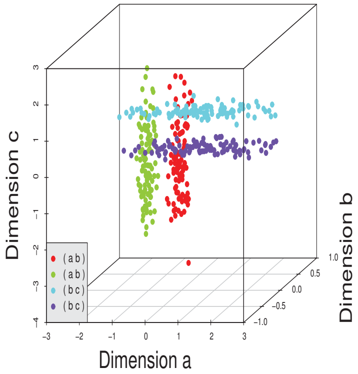
Dimension reduction and clustering

- Careful! Check how sensitive the results are to your screening
- both the type of screening and
- how aggressively you screen

Subspace clustering

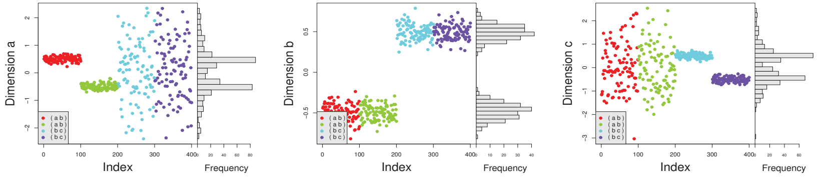
- Another method for dealing with high-dimensional data
- Assume each cluster only "lives" in a subspace (subset) of dimensions
- If we knew which subspace we could adjust how we compute distances and circumvent the COD (curse of dimensionality)

Subspace clustering



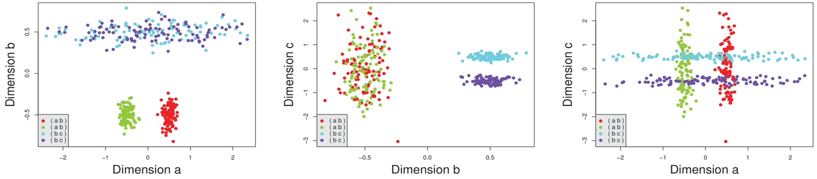
- Here are some nice figures from sigkdd review paper (see class home page) of Parsons, Hague and Liu
- 4 clusters that live in different subspaces

Subspace clustering



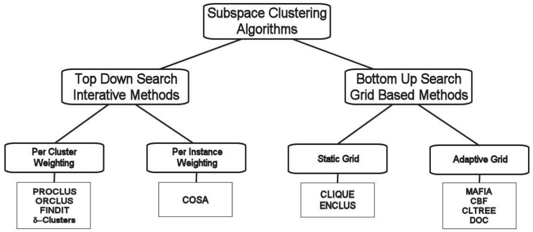
(a) Dimension a (b) Dimension b (c) Dimension c

Figure 3: Sample data plotted in one dimension, with histogram. While some clustering can be seen, points from multiple clusters are grouped together in each of the three dimensions.



(a) Dims a & b (b) Dims b & c (c) Dims a & c

Subspace clustering


Subspace Clustering Algorithms
Top Down Search Interative Methods
Bottom Up Search Grid Based Methods
Per Cluster Weighting
Per Instance Weighting
Static Grid
Adaptive Grid
PROCLUS ORCLUS FINDIT δ–Clusters
COSA
CLIQUE ENCLUS
MAFIA CBF CLTREE DOC

- In subspace clustering there are also two main approaches
- Bottom-up/Grid-based
- Top-down search

SUBSPACE CLUSTERING

An example of a bottom-up method is CLIQUE

- Generate a grid in high-dimensional space by dividing each dimension into say 10 equal length intervals
- Each high-dimensional rectangle now contains a set of observations
- We search for a connected set of dense rectangles in a subspace

Subspace clustering

CLIQUE

- Empty or near empty rectangles are removed, the density threshold tau is a tuning parameter
- For each set of two dimensions we check if there are two neighboring dense units in these two dimensions and then they are saved as a cluster.
- This is repeated for all sets of three, four, five,. . . dimensions. After every step adjacent clusters are replaced by a joint cluster.

Subspace clustering

Top-down methods work along these lines (there are many methods so check out the posted paper). Here I outline the PROCLUS method

- Start with a larger than desired set of randomly selected "medoids" that are far apart in the data
- We're now going to iteratively update a k-medoid clustering
- We select k medoids at random.
- We check for each medoid if it any good: i.e. is it at a center of densely clustered observations:
- We check in which subdimension the cluster lives by looking at within-cluster distances as a function of subspace dimension. We assign observations to the subspace medoid and if very few observations are allocated to it we remove this medoid and choose another observation at random as a new seed.

Subspace clustering

Subspace clustering outputs clusters and their subspace dimension

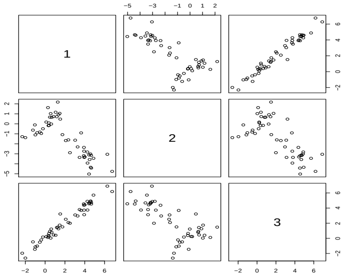
- A way to deal with complex structures and high-dimensions
- Can also be interesting to interpret clusters in terms of their subspaces
- Ongoing research and applications to video, images, genomics,...

Feature selection

Once we move to a modelbased clustering procedure we can use BIC to select features as well. An elegant approach to this, which is an extension of Mclust, is the following (implemented in the clustvarsel() package.
Consider the following; maybe not all features are relevant for clustering, either directly or indirectly.

Feature selection

Here's an example where $x1$ and $x2$ are relevant for clustering (having means $0, 0$ and $4, -3$ for the two clusters respectively, and correlation 0.6 and -0.6 between features 1 and 2 in cluster 1 and 2 respectively). Feature $x3$ is related to the clustering indirectly as $x3 = x1 + e$, $e \sim N(0, .5)$.



We can of course also have features that are completely unrelated to the clustering, i.e. not differ in mean for the different clusters and not correlated with any feature that is mean-shifted between clusters.

ClustVarSel

ClustVarSel is a procedure that decomposes the likelihood as follows. Let Z be the cluster labels for the data set:

$$p(x_c, x_{no-c} \mid Z) = p(x_c \mid Z)p(x_{no-c} \mid Z, x_c) = p(x_c \mid Z)p(x_{no-c} \mid x_c)$$

The x-variables are thus partitioned into a set x_c that is dependent on the clustering label, i.e. the multivariate normal distribution for the x_c variables have mean and possibly covariance parameters that are cluster specific. The variable set x_{no-c} are conditionally (on x_c) independent of the cluster labels. That is, if we know x_c then $x_{no-c} \mid x_c$ distributions are not cluster specific.

ClustVarSel

If x_{no-c} has a distribution that cannot be simplified to remove cluster-specific distribution parameters by conditioning on x_c, then x_{no-c} is directly related to the clustering.
The ClustVarSel procedure searches for variables to add in either the c (cluster related) or $no-c$ (not cluster related) set. The partitioning that is optimal is determined via the BIC.

ClustVarSel

- If x_1 is in the cluster relevant set already, consider adding x_2
- Fit the model where both x_1, x_2 are in the set c.
- Fit the model where x_1 is in the set c and x_2 is not. This is done via a mixture model fit with x_1 and a regression model for $x_2 \mid x_1$.
- Compute BIC for the two alternatives and pick the alternative that has the smallest BIC.
- Considering adding or removing variables from the set c until no move can be accepted (no smaller BIC alternative).

The search can be done forward (where no variable is in set c to start with) or backward (where all are in the set c initially).

ClustVarSel

Running clustvarsel on the simulation from figure above:

```
'clustvarsel' model object:

Stepwise (forward) greedy search:
  Var.proposed  BIC        BIC diff.    TypeStep  Decision
1 2            -211.3307  5.255851      Add        Accepted
2 1            -377.4739  38.873054     Add        Accepted
3 3            -377.4739  -14.769282    Add        Rejected
4 1            -377.4739  38.873083     Remove     Rejected

Selected subset: 2, 1 }
```

Clustvarsel picks variables 2 and 1 to be cluster related (correctly) and does not add variable 3 (also correct decision).

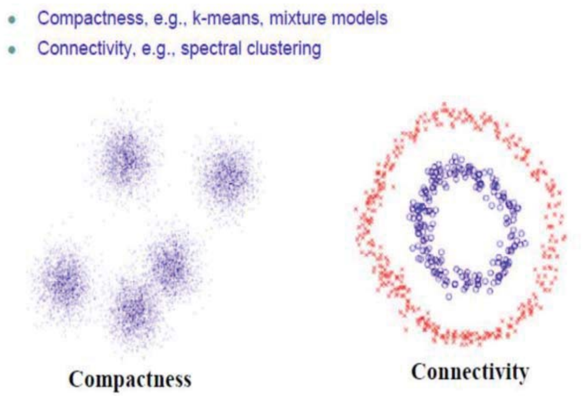
High-dimensional Classification and Clustering

- Reduce the number of parameters in the mixture model
- Assume classes/clusters live in a lower dimensional space (intrinsic number of dimensions)
- How? Generalize QDA/Mixture model to only utilize the leading PC components of the class/cluster-specific Σ_k

High-dimensional Classification and Clustering

- Assume Q_k are the leading d_k components of the $p \times p$-dimensional Σ_k
- Assume the corresponding leading eigenvalues are $a_{jk}, j = 1, \cdots, d_k$ and the remaining eigenvalues are small and equal b_k
- Think of the $p - d_k$ dimensions corresponding to the small eigenvalues as noise
- Estimate parameters under these restrictions - save a lot of parameters!
- Choose class/cluster-specific complexity (d_k) via BIC
- R-package HDclassif

Spectral clustering



- Compactness, e.g., k-means, mixture models
- Connectivity, e.g., spectral clustering

https://charlesmartin14.wordpress.com/2012/10/09/spectral-clustering/

- Most clustering methods are geared at finding dense, compact regions
- What if clusters are more complex than that?

Spectral clustering

Similarity graphs

- A similarity measure between observational pairs can be illustrated with a graph
- The length of an edge between objects inversely proportional to the similarity
- If we threshold similarities that are small we get a graph where only some observations are connected
- Graph-partitioning: which edges should we cut to form good clusters? Clearly the ones with low similarity.

Spectral clustering

Similarity graphs

- $w_i j$ is the adjacency graph edge between object i and j
- $d_i = \sum_j w_{ij}$ is the *degree* of node i
- If we partition the graph into node sets A and B the "cost" of this operation is $\sum_{i \in A, j \in B} w_{ij}$
- A good graph-partitioning minimizes this cost

Spectral clustering

Spectral clustering is a fast a simple method that produces a graph-cut

- Form the adjacency matrix W and the degree matrix D
- Define the Laplacian $L = D - W$
- Fact: For any vector f: $f'Lf = \sum_{i,j}^{n} w_{ij}(f_i - f_j)^2$
- Fact: The eigenvalues of L: $0 = \lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$

Spectral clustering

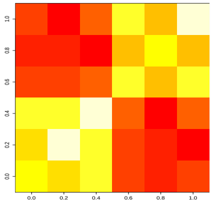
- Turns out: if there are k connected components in your graph (clusters) then there are k zero-eigenvalues of L
- and the corresponding eigenvectors can be used to find the clusters using e.g. kmeans.
- Like isomap + kmeans really.

Spectral clustering

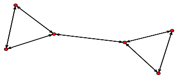
- Special case: one connected component
- Assume f is an eigenvector with value 0
- $0 = f'Lf = \sum w_{ij}(f_i - f_j)^2$ means we have have $f_i = f_j$ for all $w_ij > 0$
- That is, f has to be constant in any part of the graph that is connected!
- Now think about a case with k connected components
- The corresponding eigenvector has to be constant for all objects that are connected!

Spectral clustering

Simple example: block-diagonal similarity matrix



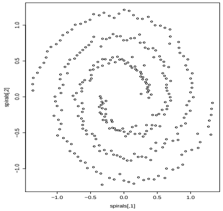
corresponding graph

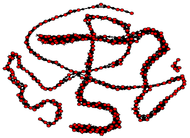


and leading eigenvectors $(0, 0, 0, .57, .57, 57)$ and $(.57, .57, .57, 0, 0, 0)$

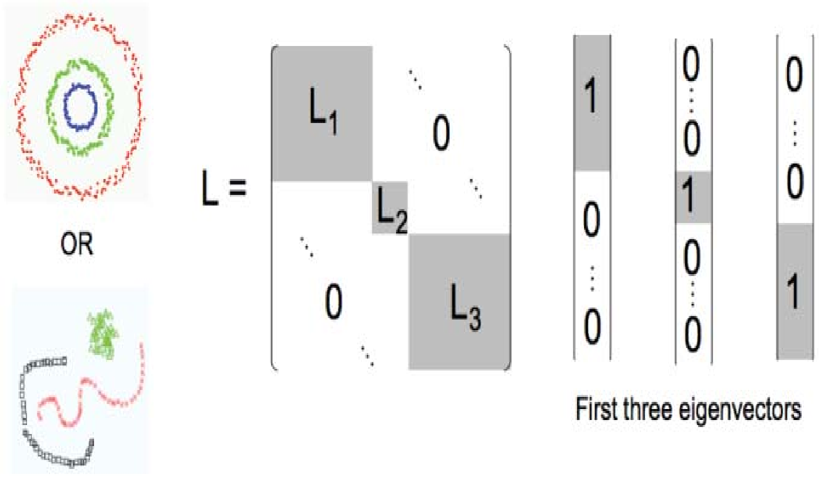
Spectral clustering

More complex example:





Spectral clustering



Spectral clustering

- Compute a similarity matrix S
- From this, compute the graph or adjacency matrix W (e.g. by thresholding the similarities)
- Compute the Laplacian L
- Compute the eigenvalues of L - look for a good separation between small and large values
- Cluster the eigenvectors corresponding to the smallest eigenvalues using kmeans.

Consensus clustering

- Any method that comprises many steps is subject to instability since each step is a source of error
- How many features, how many eigenvalues?
- In addition, many clustering methods are quite sensitive to small data perturbations

Consensus clustering

- If you can do things once, you can do it 100 times!
- Add some randomness to the procedure and run it many times
- Retain clusters that are *stable* across multiple runs!

Consensus clustering

- How add randomness?
- Subset of features
- Subset of features + PCA
- Random projections
- ....

Consensus clustering

- Each run produces a clustering result
- How do we combine these?
- Some methods compare the clusters in terms of overlap
- Other methods use a similar idea to RF clustering: for each pair of objects, count how many times they appear in a cluster together. Use this is a new similarity metric and use e.g. hierarchical clustering to produce a final result.
- I like the latter approach because it gives you a lot of flexibility in which clustering procedures to compare across runs.

Graphical Lasso

- A "hot area" for research is *network modeling*
- Nice visualizations of complex data!
- Related to clustering in the sense that....
- ... observations are represented in a network - neighbors are more similar.
- But also a more complex question - neighbors are close once dependency on other observations taken into account

Graphical Lasso

- Lots of methods for network modeling (Bayesian networks, information theoretic, directed/mechanistic,...)
- Here we will focus on *sparse modeling*
- Assume data comes from a multivariate normal model $N(\mu, \Sigma)$
- The inverse of the covariance matrix Σ, Θ, is called the *precision matrix*

Graphical Lasso

- The inverse of the covariance matrix Σ, Θ, is called the *precision matrix*
- Fact: The precision matrix is non-zero for entry i, j only if the *partial correlation* between i, j is non-zero
- Partial correlation $=$ correlation between i, j once dependency on all other observations accounted for
- $\theta_{i,j} = Cov(X_i, X_j \mid X_k, k \neq i, j)$
- Can compute the partial correlation from residual correlation from regression of i on all other variables and j on all other variables

Graphical Lasso

- In practice, can't compute the inverse $\hat{\Theta}$ of the $p \times p$ $\hat{\Sigma}$ if $p > n$
- Sparse modeling to the rescue
- Maximize the gaussian log-likeihood with penalty $\lambda \sum_{j<i} |\theta_{i,j}|$
- Methods: gradient based glasso, lasso-regression based neighborhood selection.
- Packages glasso and huge

Graphical Lasso

- Does it work?
- Like sparse regression, there are some caveats. Too many highly correlated Xs, we cannot identify the network model.
- Is the data sparse?
- Fixes: randomized lasso. Run glasso many times with random penalties: check how often a graph-link is selected.
- High-dimensional data? First filter. If a set of variables has no correlation with any member of another set exceeding λ, you can run glasso separately on the sets (implemented in huge package).

Graphical Lasso

- Can extend this to group-penalties or fused penalties for network modeling across different data sets
- JGL package

Mini 4

Doodle!

1. Clustering + Dimension reduction
   - Filtering
   - Wrapper/Variable selection
   - Data projections
2. Classification + Dimension reduction
   - Filtering
   - Wrapper/Variable selection
   - Data projections

Mini 4

3. Data representations
    - On different data sets, investigate different data representations
    - Impact on clustering/classification
4. Semi-supervised learning
    - Small data set with labels, massive data set without - how do we make use of this?
    - Check literature for methods (in R there are packages RMixMod and RSSL for example).
    - Either apply to real data or you create a semi-supervised problem by randomly "hiding" labels on real data.
    - How much can methods be boosted from unlabeled part? Can you simulate a case where it might hurt instead? (Assumptions on similarity of labeled vs unlabeled data).

Mini 4

5 Consensus clustering
- Check the literature for a few variants on consensus clustering
- Apply to 2-3 data sets
- Discuss and interpret

Mini 4

6 Clustering and big sample size
   - Some methods scale better with respect to sample size than others (and some with respect to dimension)
   - Check literature for big-n clustering methods
   - Apply to 2-3 data sets and discuss

7 Other methods
   - There are so many methods for clustering/classification. Pick 1-2 from the book or other sources and compare to some of the methods we have discussed.
   - Pros and Cons?
   - Examples: sparse-PLS classifiers, SVMs, NN
   - Make sure you can compare modeling assumptions to some of the methods from class - so not just compare performance w/o insight.