Clustering

We devoted ourselves until now to the problem of *supervised learning*. The first-rate indicator that this is the case is presence of class labels in the training data.

Clustering (also known as cluster analysis or grouping) is an example of *unsupervised learning* and – however paradoxically it seems – is useful for classifier design.

The main purpose of clustering is verification of the labelled data set. The goal is determination of the number of groups (clusters) that data at our disposal forms at given feature set.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Clustering

What is a cluster?
A set of similar **objects,** (they share some properties).
A set of feature vectors grouped in some neighbourghood.

We concentrate on the second possibility, assuming optimistically, that our data features' are in a metric space. We should pay attention to select metric (distance function) which fits our classifier. Particularly, for Bayes classifier (parametric, with normal distribution) we should use Mahalanobis metric.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Clustering quality assessment

The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.

K. Jain, R. C. Dubes

We use:

Internal evaluation: determination of the value of a certain coefficient, which represents the "quality" of the allocation of points to groups (we should have similar points in a group and dissimilar points should belong to different groups)

External evaluation: when we have information about the partitioning from other sources (from an expert, from other clustering procedures); in latter case, we compute the similarity of clusterings.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Silhouette coefficient

We determine the average distance of the sample and the samples belonging to all groups constituting the result of clustering.
By *a(i)*, we mark the average distance in the cluster to which *i* was assigned.
From the remaining average values (i.e. averaged distances to other clusters), we select the smallest and mark it as *b(i)*.
The silhouette coefficient is calculated as:

$$s(i) = \frac{b(i) - a(i)}{\max\left|a(i), b(i)\right|}$$

This coefficient is interesting because it gives information about each sample: values < 0 suggest that the sample is not allocated properly; ~ 0 that the sample lies at the border between clusters; for values > 0, you can hope that the point is assigned to the proper cluster.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Clustering similarity

Clustering similarity measure is necessary if we are using methods depending on some randomly selected values. If for example, for different starting parameters we obtain similar clusters, then we can expect that this results from existence of such clusters in data. The method to compute similarity of two clusterings A and B:

1. Let's take all sample pairs - N(N-1)/2.
2. Each pair is assigned to one of 2 classes:
   0 - both elements are in the same cluster
   1 - elements are in different clusters

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Clustering similarity

3. We compute 4 numbers, depending on classes to which belong pairs in both clusterings:

A\B	Class 0	Class 1
Class 0	a	b
Class 1	c	d

4. Similarity measure: $S(A,B) = \frac{a+d}{a+b+c+d}$

   (Rand's index)

Using the same values we can compute also:

Jaccard's index $S_{Jaccard}(A,B) = \frac{a}{a+b+c}$,

Folwkes-Mallows index $S_{Folwkes-Mallows}(A,B) = \frac{a}{\sqrt{(a+b)(a+c)}}$

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Rand's index – implemetation

It's easy to write poor implementation of Rand's index computation (a loop over all pairs of samples imposes itself, but it's loosy solution: how many pairs of samples do we have for a data set containing 10^6 samples?)
To implement Rand's index effectively we'll use *contingency table*, describing cluster membership of samples in **both** clusterings.

	G_{B1}	G_{B2}	…	G_{Bm}	Σ
G_{A1}	n_{11}	n_{12}	…	n_{1m}	n_{A1}
G_{A2}	n_{21}	n_{22}	…	n_{2m}	n_{A2}
…	…	…	…	…	…
G_{An}	n_{n1}	n_{n2}	…	n_{nm}	n_{An}
Σ	n_{B1}	n_{B2}	…	n_{Bm}	

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Rand's index – implemetation

Taking into account the first row of contingency table we can compute the count of samples assigned to the same cluster in both clusterings: $n_{A1same} = \sum_{j=1}^{m} pairs(n_{1j})$

Immediately we have also the count of samples, which in A are in the same cluster (namely in G_{A1}), while in B they are in different clusters: $n_{A1diff} = pairs(n_{A1}) - n_{A1same}$

Thus: $a = \sum_{i=1}^{n} n_{Aisame}$ and $b = \sum_{i=1}^{n} n_{Aidiff}$

Working with columns allows us to computed *a* and *c* values. Because the total number of pairs is known we can immediately compute *d*.

What is computational and memory complexity of this solution?

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Clustering Methods

1. Methods minimizing squared error.
2. Hierarchical clustering
   - top-down,
   - bottom-up.
3. Graph-theoretical clustering.

Jain, A.K., Murty M.N., and Flynn P.J. (1999): *Data Clustering: A Review*, ACM Computing Surveys, Vol 31, No. 3, 264-323.
http://www.cs.rutgers.edu/~mlittman/courses/lightai03/jain99data.pdf

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Minimizing squared error

We have N points, that must be partitioning into *g* clusters.

Cluster *k* contains M_k points and has centroid (or sample mean) x_k: $\bar{x}_k = \frac{1}{M_k} \sum_{l=1}^{M_k} x_{kl}$

Squarred error of cluster *k*: $\varepsilon_k^2 = \sum_{l=1}^{M_k} (x_{kl} - \bar{x}_k)^T (x_{kl} - \bar{x}_k)$

We minimize overall error of clustering into *g* clusters: $E_g^2 = \sum_{k=1}^{g} \varepsilon_k^2$

Exhaustive search is not feasible, the best that can be achieved is the local E^2 minimum.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

k-means Algorithm

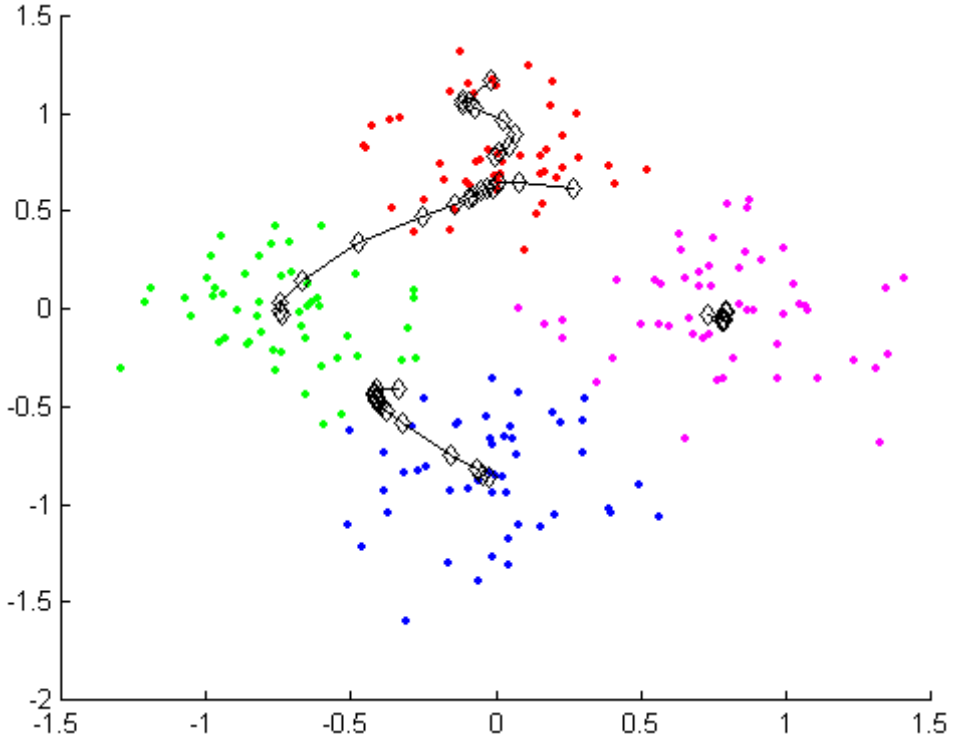
1. Choose initial values for the means $\bar{x}_k$
2. Assign the N points to the closest means.
3. Recompute means $\bar{x}_k$ in function of the points assigned to them.
4. If any mean changed value, return to 2; otherwise stop.

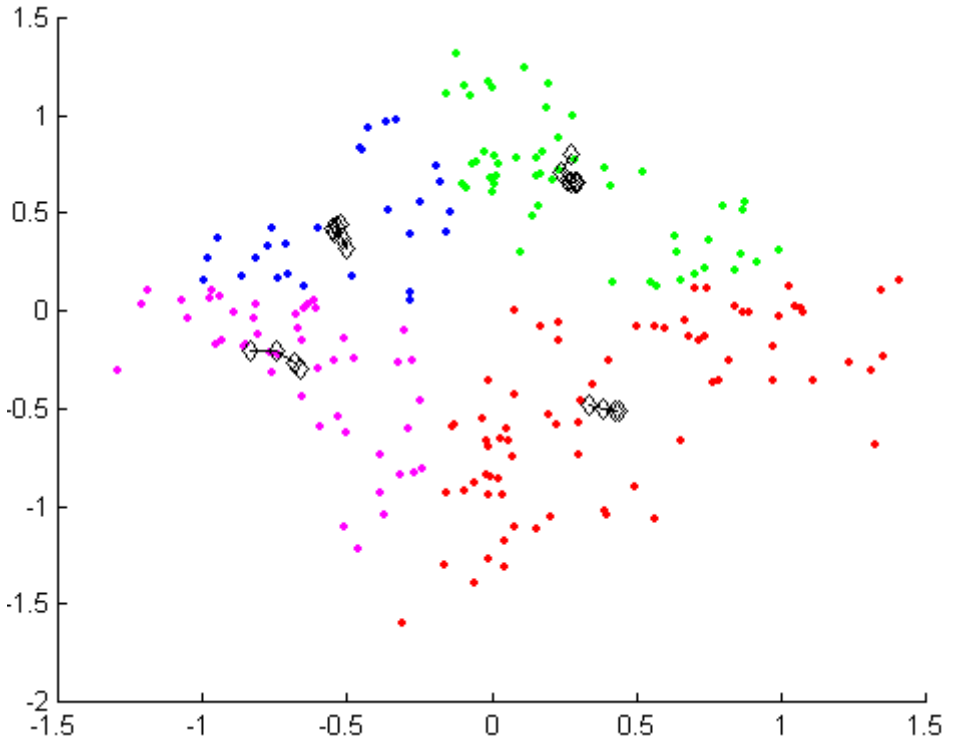
Two obvious problems:

Wrong guess of initial mean values $\bar{x}_k$ leads to wrong clustering ("bad" local minimum). We should repeat clustering several times using different initial mean values and compare resulting clusterings.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

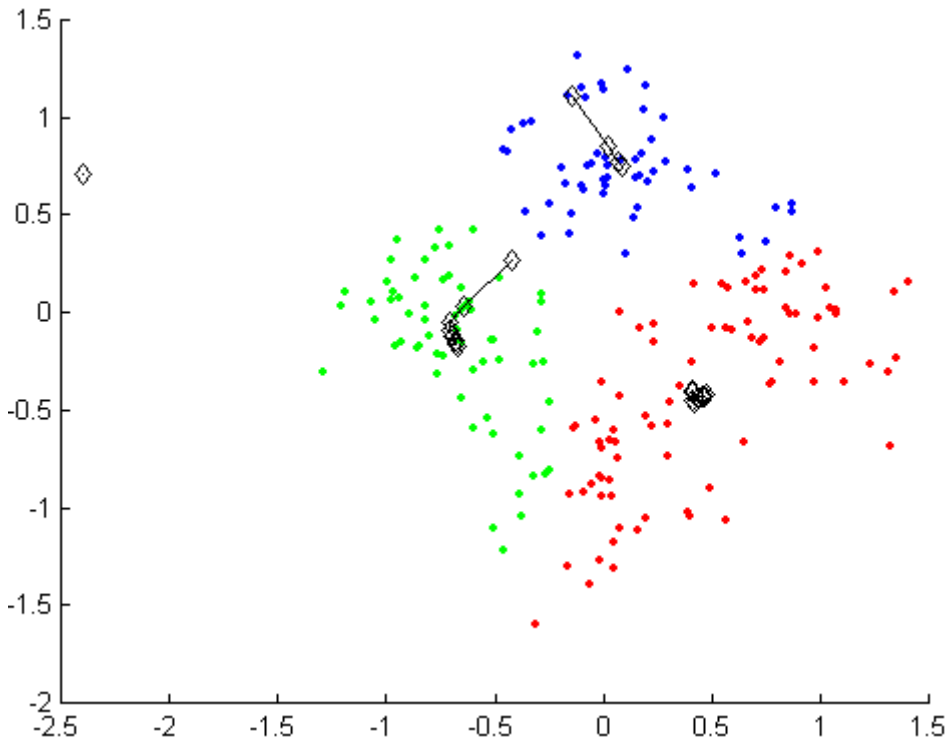
k-means Algorithm

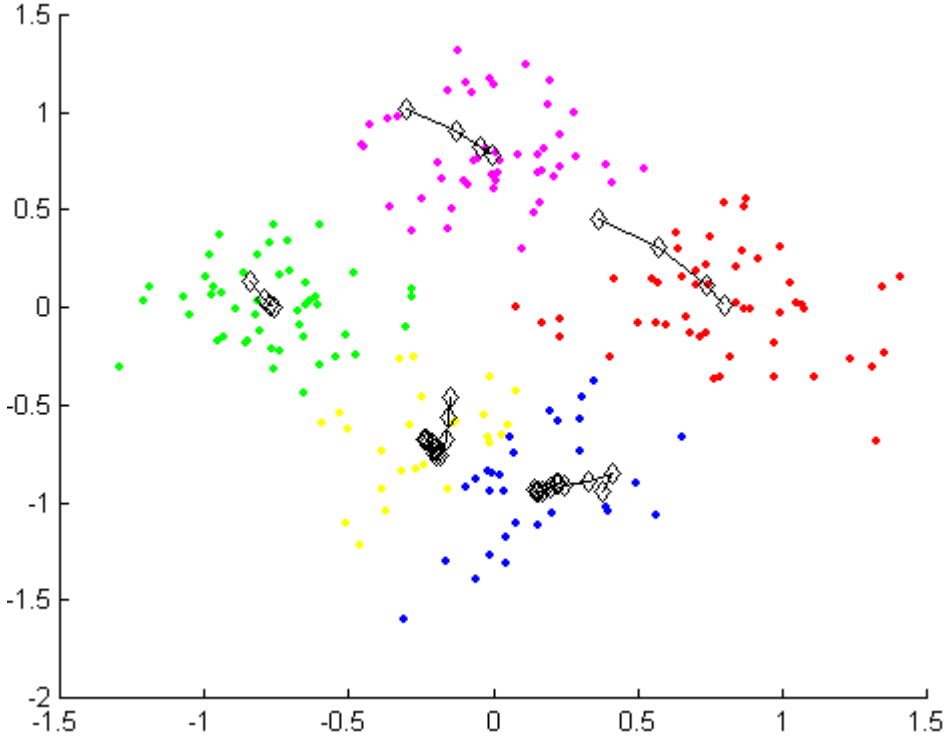




Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

k-means Algorithm





Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Fuzzy k-means

1. Choose initial values for the means $\bar{x}_k$
2. Compute membership degree of x_j to cluster *k*, using attenuated distance: $a(x_j, \bar{x}_k) = e^{-d(x_j, \bar{x}_k)^2}$

   membership degree: $u(x_j, k) = \frac{a(x_j, \bar{x}_k)}{\sum_{i=1}^{g} a(x_j, \bar{x}_i)}$
3. Compute new **fuzzy** means values $\bar{x}_k$ from the points assigned to the cluster: $\bar{x}_k = \frac{\sum_{j=1}^{M_k} a(x_j, k)^2 x_j}{\sum_{j=1}^{M_k} a(x_j, k)^2}$
4. If any mean changed value, return to 2; otherwise stop.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

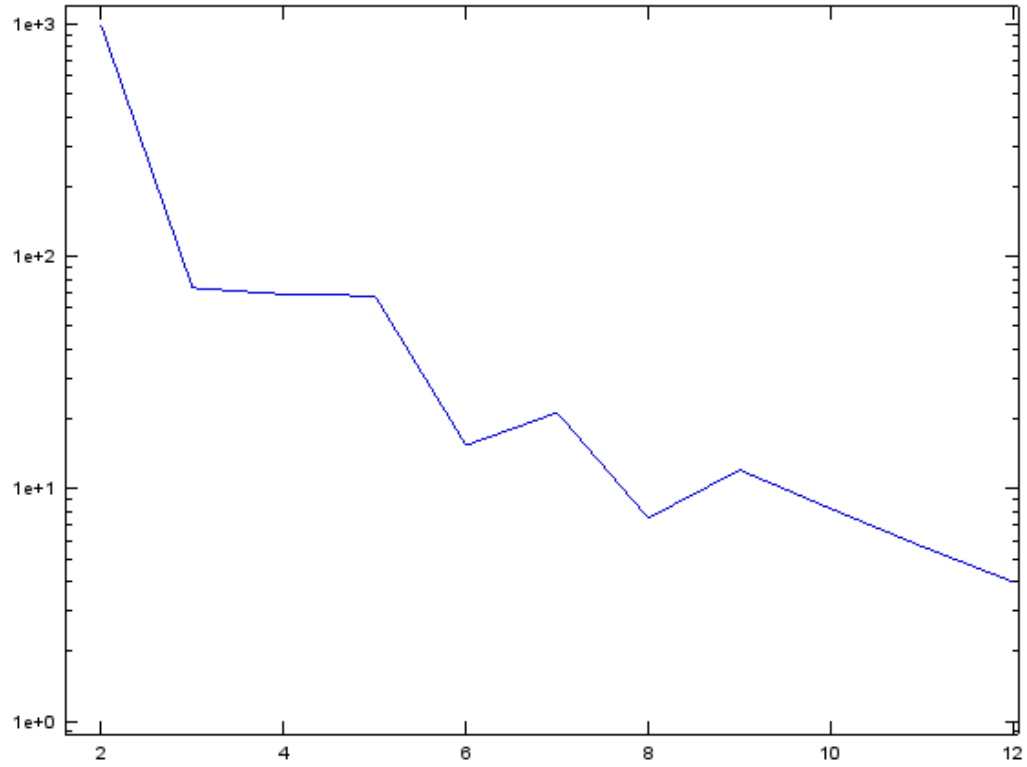
Some questions & remarks

1. Where to find initial means values?
2. Clustering result depends on initial values.
   Often we take several clustering iterations with different sets of initial values and compare clusters obtained.
3. Empty clusters (generally smaller than some predefined treshold).
   Their presence may suggest wrongly placed samples (outliers), but it is sometimes the result of improper number of clusters.
4. Result depends on metric used.
5. Again: how are we supposed to learn clusters number g?

As in the PCA case we can use a diagram of „explained” data variance. We prepare such a diagram for subsequent *g*, values in the hope we will be able to see „true” number of clusters on it.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Error function diagram



Error values on the diagram are averaged!

(a few errors for just one point on the diagram: 5.2 5.5 **0.34** **32.3** 4.2 5.9 1.0 5.5)

How many groups are there in the data set, for which clustering error is shown on the diagram?

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Hierarchical methods

A *top-down (divisive)* method begins with all cases in one cluster. This cluster is gradually broken down into smaller and smaller clusters, according to some division criterion.
Cluster subdivision is stopped after reaching *g* clusters.

A *bottom-up (agglomerative)* method start with (usually) single member clusters. **Closest** clusters are gradually fused until one (or *g*) large cluster is formed.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Bottom-up Algorithm

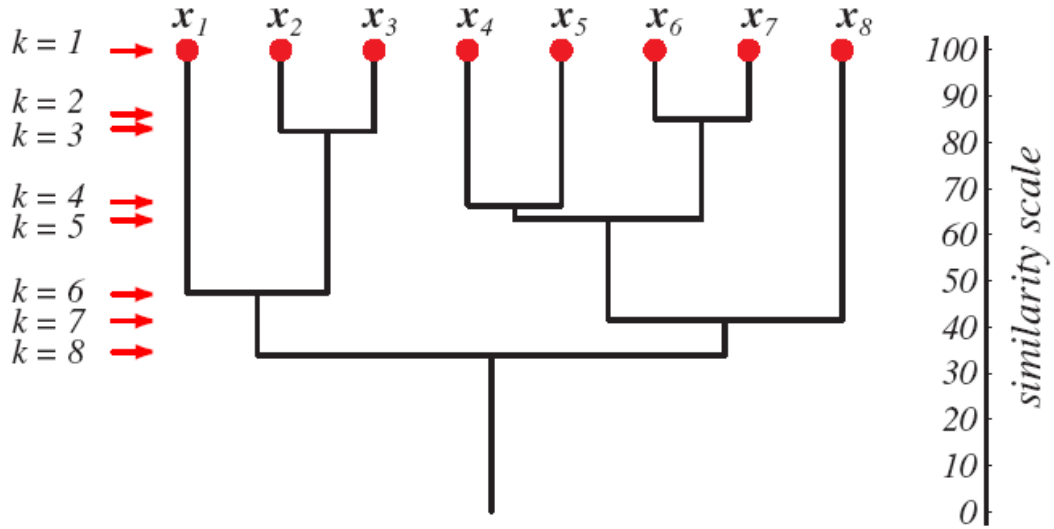
1. Create one-element clusters.
2. Compute distances between clusters - dissimilarity matrix,
3. Find in dissimilarity matrix two closest clusters.
4. Join the two found clusters.
5. If the number of clusters is greater than given minimum return to 2; otherwise stop.

The natural representation of hierarchical clustering is the tree showing how the samples (and clusters) are grouped, called *dendrogram*. Its analysis can show existence of „natural" clusters in data (if we have the similarity scale shown in our dendrogram). Yet another visualization – Venn diagrams – reveals the hierarchical structure but not the quantitative distances between clusters.

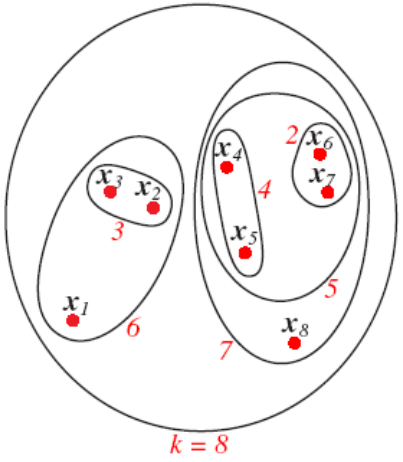
Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Bottom-up Algorithm

Dendrogram:



Venn diagram:



Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

How to compute cluster distance

1. Average linkage:
   For each point in one cluster we compute its distance to each point in the other cluster. Average distance is the clusters dissimilarity measure.
   We compute distance between clusters' means (centroids).
2. Complete linkage (maximum):
   The greatest distance between points of both clusters.
3. Single linkage (minimum):
   The smallest distance between points of both clusters.
4. Variation distance:
   We treat both clusters as a one and compute variation of such a new cluster. Variation (after the fuse) is the measure of dissimilarity of clusters.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Graph-theoretical methods

We construct a graph with nodes in sample points. Every graph edge has assigned weight (usually the distance between nodes). Overall weight of the graph should be minimized.
After constructing the graph, longest edges are removed to split points into clusters.

Kruskal's Algorithm:

1. Construct the forest (each tree consists of one node).
2. The edges are placed in a priority queue.
3. Until n-1 edges have been added:
    4. Get the cheapest edge from the queue.
    5. If it form a cycle, reject it,
    6. Else add it to the forest. (Adding it to the forest will join two trees together).

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Minimum spanning tree

We have MST containing *n* points. We add point *x* and construct MST containing *n+1* points. The whole procedure is performed in two steps:

1. Find x's nearest neighbours
2. Reconstruct MST

Finding nearest neighbours of x:

1. Find NN of x - c - in set of points V
2. Add c to output set {p}
3. For each point v from the set V
   if d(v,x) >= d(v,c)
   remove v from the set V
4. If V is not empty return to 1.

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds

Minimum spanning tree

Reconstruction:

1. Find the set of NN of x - {p}
2. Find the point nearest to x in {p} - z, and delete it from {p}
3. Add edge x - z
4. For each point p in {p}
    5. Find the longest edge in the path from *x* to *p* - E.
    6. If d(x,p) <= E
       delete edge E
       add edge x - p

Preparing of these materials was supported by a grant from Liechtenstein, Iceland and Norway through EEA Grants and Norway Grants and co-financed by domestic funds