PASCAL Workshop on

Stability and Resampling Methods for Clustering

Tübingen, July 16-18, 2007

Clustering Stability — a literature review, many questions, and a few ideas for answers

Workshop on Stability and Resampling Methods for Clustering

Ulrike von Luxburg

June 2007

Overview

- Stability for model selection – literature review
- Practice vs. theory – many questions
- Some ideas to solve some of the questions

The principle of stability

Scientific results should be reproducible.

- If two researchers collect similar data by similar methods and apply the same algorithm, the outcomes should be similar.

Ideally, would like to have algorithms which are robust

- ... with respect to the sampling of the data
- ... with respect to the noise in the data
- ... with respect to numerical issues

Consider this as a minimal requirement for any machine learning algorithm.

Stability as a tool for model selection in clustering

Model selection for clustering is difficult in general:

- Don't have ground truth
- Difficult to evaluate clustering results
- Difficult to compare clusterings.

Thus idea: evaluate clusterings indirectly using stability.

- Want that our results are stable.
- Hence, choose parameter for which the result is most stable.
- In practice, this often works

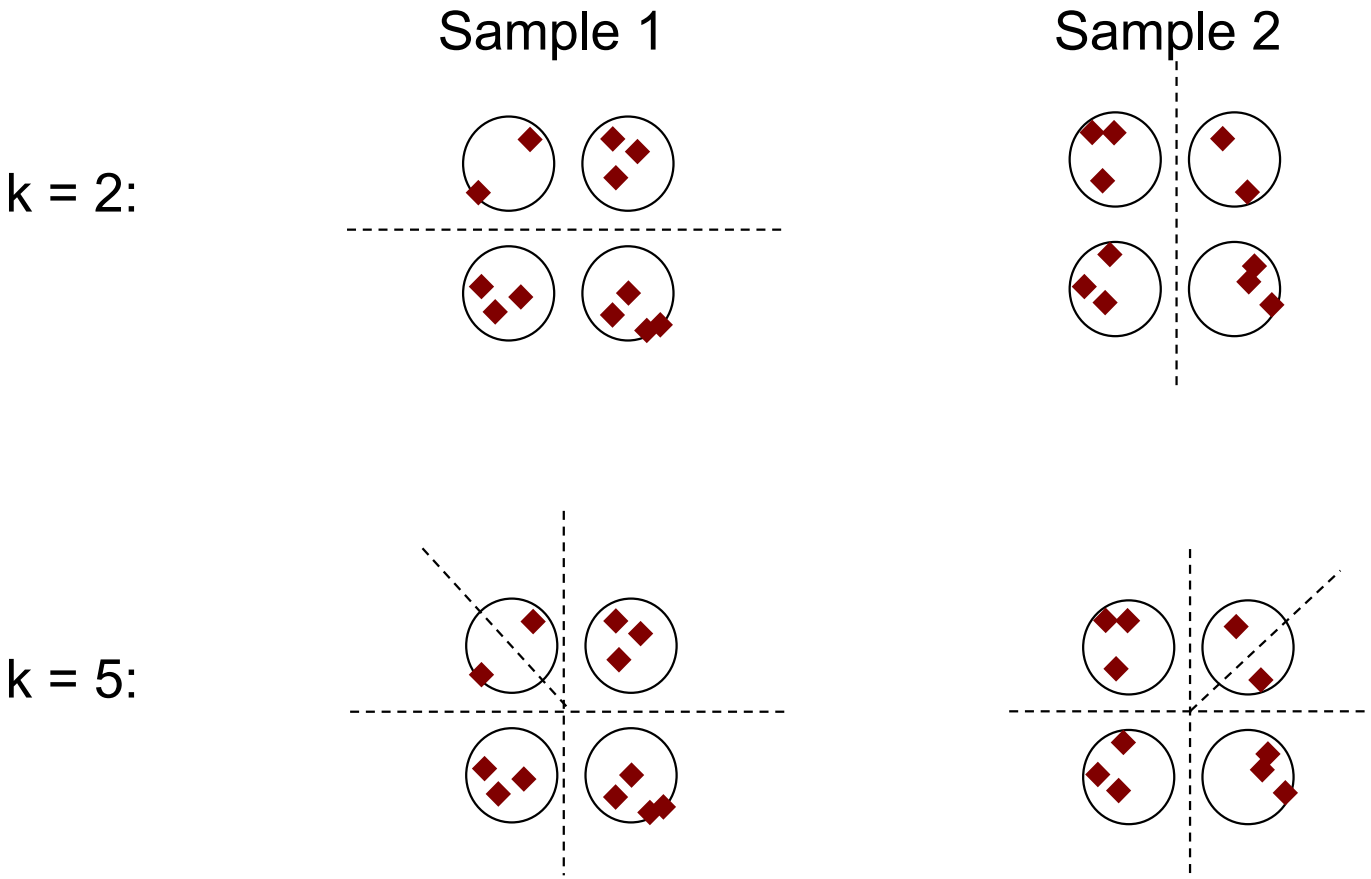
Theory: ???

Stabilty – the general principle

- Given a data set $X_1, ..., X_n$, a clustering algorithm $\mathcal{A}$
- For different values of k (=number of clusters):
  - draw subsamples of the given data
  - cluster them in k clusters using $\mathcal{A}$
- compare the resulting clusterings
  - define some distance between the clusterings
  - compute some notion of "stability" depending on how much the clustering distances vary
- choose the parameter k which gives the "best" stability (where "best" is defined in different ways)

The toy figure in favor of stability

How many clusters?



The different steps involved in stability

Generating artificial data sets

- Draw a subsample of the original data set.

  Levine and Domany (2001), Ben-Hur, Elisseeff, and Guyon (2002), Fridlyand and Dudoit (2001), Lange, Roth, Braun, and Buhmann (2004)
- Use the original data set, but add random noise to the data points Bittner et al. (2000)
- If the original data set is high-dimensional: use different random projections in low-dimensional spaces, and then cluster the low-dimensional data sets Smolkin and Ghosh (2003)
- If we are in a model-based framework, sample data from the model Kerr and Churchill (2001)

Generating artificial data sets (2)

In all cases, there is a trade-off which has to be treated carefully:

- If we change the data too much (subsample is too small; noise is too large), then we might destroy the structure we want to discover by clustering.
- If we change the data too little, then more or less everything will be stable.

How to use the clustering algorithm

- In the stability approach, we usually fix a clustering algorithm and its parameters
- (This is different from standard ensemble methods, where people vary the algorithm rather than the data set)
- But often people use randomized algorithms (e.g., random initialization in K-means); here randomization different for each run of the algorithm

Distances between the clusterings

If clusterings are defined on the same data set: easy.

Count how many pairs of points end up in the same or in different clusters according to both clusterings. Use this to build various distance/similarity scores:

- Rand Index
- Jacard Index
- Hamming distance
- Variation of Information Meila (2003)
- ... many more ...

Distances between the clusterings (2)

To compare clusterings on different data sets: two approaches:

Using a restriction operator:

- compute the joint domain $S = \{X_1, ..., X_n\} \cap \{X'_1, ..., X'_m\}$ of both clusterings
- Restrict both clusterings to $S \rightsquigarrow \mathcal{C}'_1, \mathcal{C}'_2$
- Compute distance between $\mathcal{C}'_1$ and $\mathcal{C}'_2$ (easy as now defined on same domain)
- Note that this only makes sense if the two domains have a reasonable overlap.

Problem: we loose a lot

Distances between the clusterings (3)

Using an extension operator:

- Extend both clusterings from their domain to the domain of the other clustering (or even to the whole underlying space)
- Then compute a distance between the resulting clusterings (easy as now defined on same domain)
- For some algorithms there exist natural extensions:
  - K-means (just assign new points to the closest cluster center)
  - single linkage (assign new points to the same cluster as the closest data point belongs to)
  - spectral clustering (using integral operators)

Distances between the clusterings (4)

- If one only needs to extend to a few new points: greedy heuristic
- Otherwise: use a classifier as extension operator!
  - Classifier should "fit" to clustering.
  - Prototype classifier for K-means
  - Nearest-neighbor classifier for single linkage
  - Often not clear which one to use.

Problem: what bias does classifier introduce to stability?

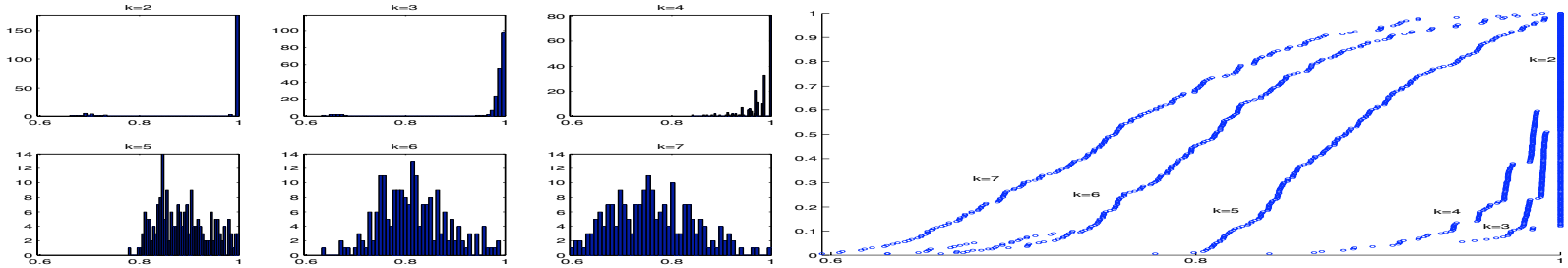
Which clusterings to compare?

- The clustering of the original data set with the clustering of a subsample Levine and Domany (2001)
- Clusterings of overlapping subsamples

  Ben-Hur et al. (2002)
- Clusterings of disjoint subsamples

  Fridlyand and Dudoit (2001), Lange et al. (2004)

End up with an empirical distribution over distances/similarities between those clusterings, for different values of k.



Ben-Hur et al. (2002)

Stability scores

Now we need to define when we say the results are "stable":

- Most people use: stability = mean(distances between clusterings)
- Some people use: stability = area under the cumulative distribution function of the distance scores.

  Ben-Hur et al. (2002), Bertoni and Valentini (2007)
- The empirical distribution of course contains more information, for example the number of modes. But as far as I can see, nobody has used this information.

Normalization

Note: $stability(k)$ scales with k, independently of the structure of the data. Need to normalize!

Normalization using a reference null distribution: Fridlyand and Dudoit (2001), Bertoni and Valentini (2007)

- Repeatedly sample random artificial data sets from some null distribution (e.g the uniform distribution)
  - Uniform distribution on data domain
  - Scramble features of the data points
- Apply the clustering algorithm to the uniform data sets and compute stability scores. Leads to a distribution of scores $stability_{norm,r}$ (where r is an index over the repetitions)

Normalization (2)

Normalization by random labels: Lange et al. (2004)

- For each of the artificial data sets:
- Instead of clustering it, assign random cluster labels.
- Then compute the distances between the random clusterings, and the corresponding stability score. $\rightsquigarrow$ $stability_{norm}(k)$

Selecting K, finally

First approach:

minimize normalized stability score, i.e.

$$K = \text{argmin}\, stability(k)/stability_{norm}(k)$$

Levine and Domany (2001), Ben-Hur et al. (2002), Lange et al. (2004)

Selecting K, finally (2)

Second approach: use some kind of statistical test:

- Compare the actual similarity score to the distribution of random similarity scores.
- For each k, test whether $stability(k)$ it is significantly different of $stability_{norm}(k)$.
  - Using bootstrap test: Fridlyand and Dudoit (2001)
  - Using approximation by a χ^2-test Bertoni and Valentini (2007)
  - Using test based on Bernstein inequality Valentini et al., submitted
- Among the significant k, choose the one which is most significant.

Stability in theory

Negative results on stability

Formalize what we mean by stability:

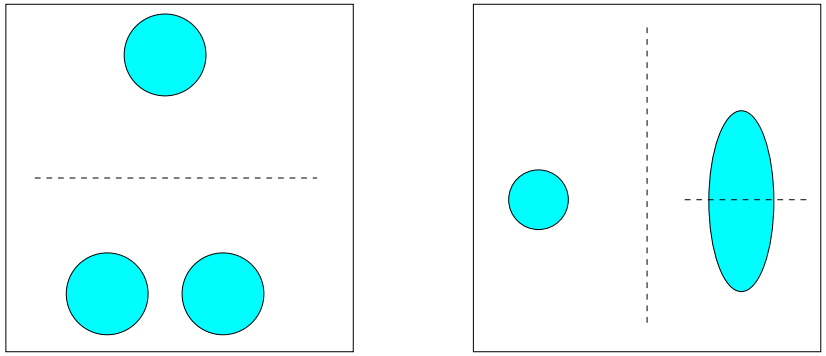
- Consider a clustering algorithm $\mathcal{A}$ which minimizes some empirical cluster quality function Q_{emp}.
- Assume that the algorithm always finds a global minimum of Q_{emp} (no convergence issues).
- Denote by S_n, $\tilde{S}_n$ two independent samples of size n drawn i.i.d. according to probability distribution $\mathbb{P}$.
- Let d be a distance function between clusterings.
- Define stability of algorithm $\mathcal{A}$ with respect to sample size n:

$$stab(\mathcal{A}, n) := \mathbb{E}_{S,\tilde{S}} \; d(\mathcal{A}(S_n), \mathcal{A}(\tilde{S}_n))$$

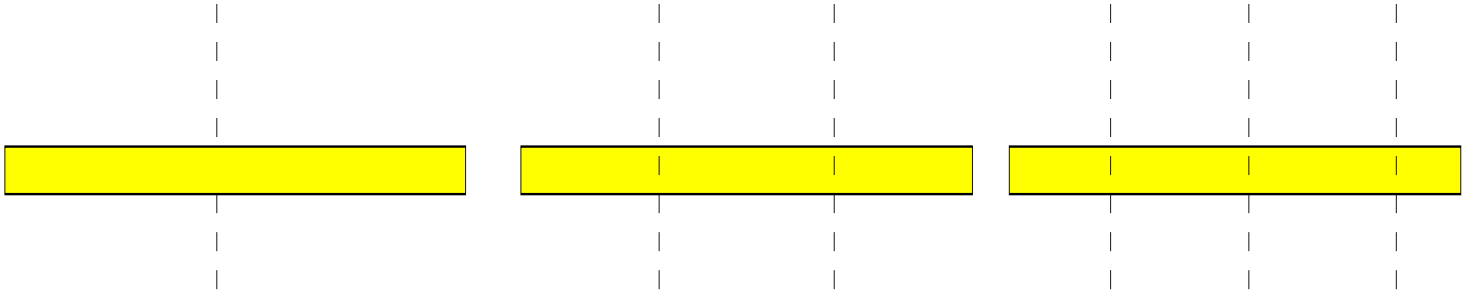
Negative results on stability (2)

The counter-stability toy figures: stability even for "wrong" k

- Non-symmetric distribution: stability even for "wrong" k



- Uniform distribution on $[0, 1]$: k-means is stable for all k

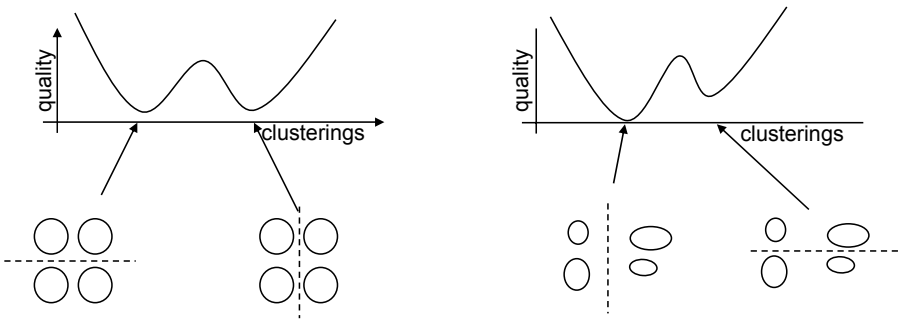


For large n we often have stability for every k!!!

Negative results on stability (3)

Theorem Ben-David, von Luxburg, and Pál (2006), Ben-David, Pál, and Simon (2007)

- Assume that Q has a unique global minimum. Then any clustering algorithm A which minimizes Q_{emp} in some consistent way is stable for large n, that is $\limsup_{n\to\infty} stab(A, n) = 0$.
- Assume that the global minimum of Q is not unique. Then A is not stable.



This view is also supported by Krieger and Green (1999), Rakhlin and Caponnetto (2007)

Where does this leave us?

Practitioners say: in applications stability often works.

Theoreticians say: at least in the limit for $n \to \infty$ it is problematic.

????

Where is the catch?

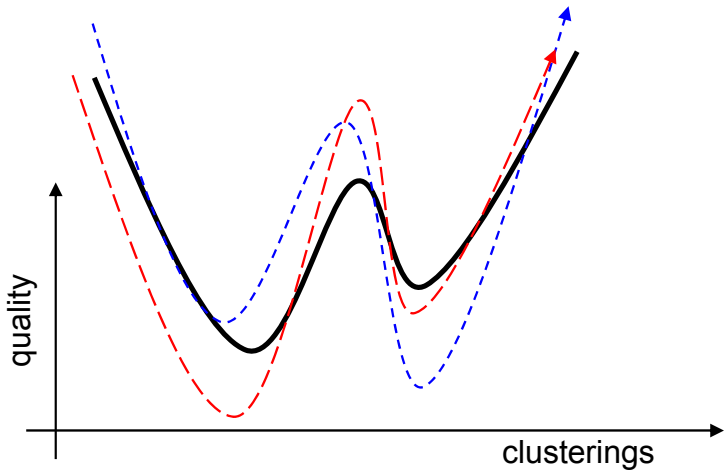
Catch 1:
large vs. small sample size

First catch: large vs. small sample size

The negative results only concern *large* sample size. What about small sample size? One possible explanation:

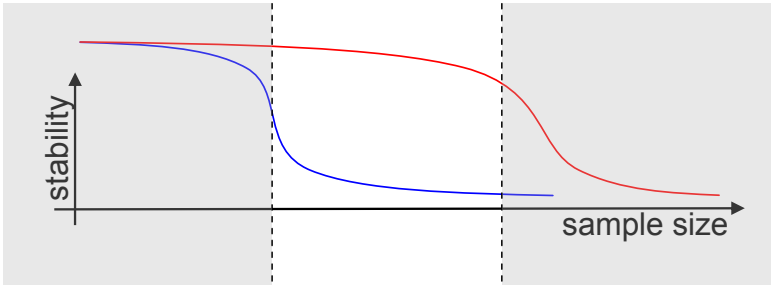
We can only estimate the quality function q up to a certain accuracy.

We cannot distinguish local and global minma.



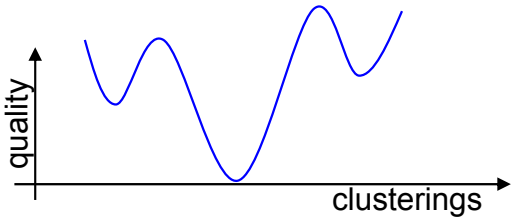
Possible solution: "stability window"

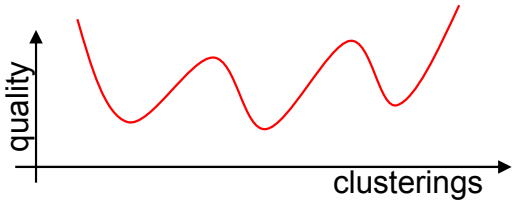
Stability as a criterion for model selection only works in a certain "window" of sample sizes:



A good clustering qualtiy function has the following property:

- For "the right k" it is already stable for small n
- For "a wrong k" it is only stable if n gets extremely large





However, don't find it likely that this really explains the theory/practice gap.

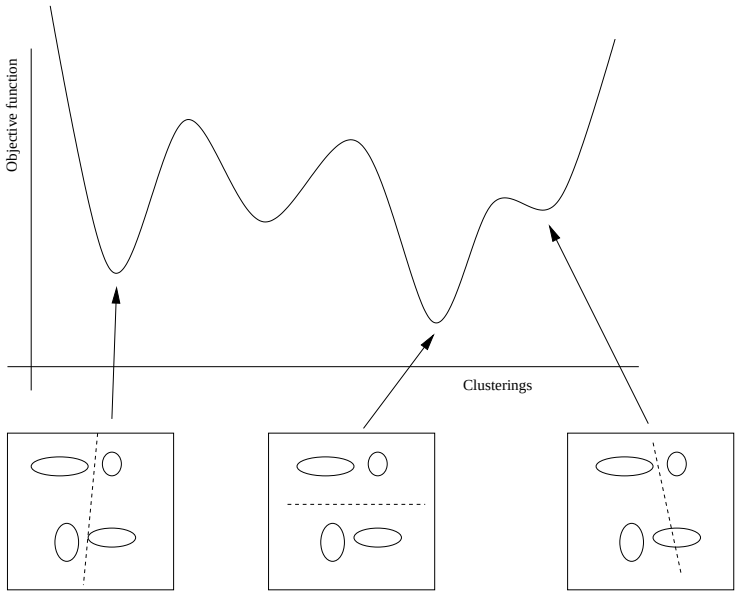
Catch 2:
global vs. local minimum

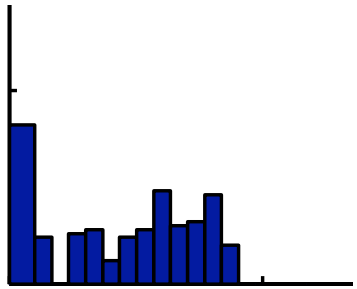
Second catch: attaining the global minimum

- The negative results only concern algorithms which find the global minimum of the objective function.
- However, most clustering algorithms try to solve NP hard problems. In practice, they often only end in local minima.
- To improve the quality of the local optimum, one often uses randomized algorithms (e.g., K-means with random initialization). Then we have a completely different story!
- Instead of stability with respect to resampling we mainly consider stability with respect to randomization of the algorithm!

Possible solution: exploring objective function

- Depending on initialization, the algorithm ends in different local minima.
- Can use stability to measure how different those minima are.





left: objective function for fixed $k = 2$;
right: histogram of distances of solutions.

Possible solution: exploring objective function (2)

To make this argument rigorous, would need to prove:

- Given a probability distribution with k "true clusters"
- Assume we run the clustering algorithm with many different initializations on one (or several) samples.
- Then, with high probability (over the randomization of the algorithm, and if applicable with respect to the sampling):
  - stability "finds" the "right k"
  - Reason: for "the right k", the objective function only has one distinct minimum in which the algorithm ends.

Big question on the way: how does the global geometry (e.g., number of local minima) of the objective function depend on the underlying distribution (e.g., number of clusters)?

Possible solution: exploring objective function (3)

Note that for many clustering algorithms:

- The qualitative behavior ("global geometry") of the objective functions is the same for most samples.
- Then resampling of the data is not really the crucial part!
- Can simply run algorithm on same sample with different randomization of the algorithm (e.g., initialization)
- This scenario is not covered in the negative arguments ...

Catch 3: What is "the right K", actually?

The "correct K", first approach

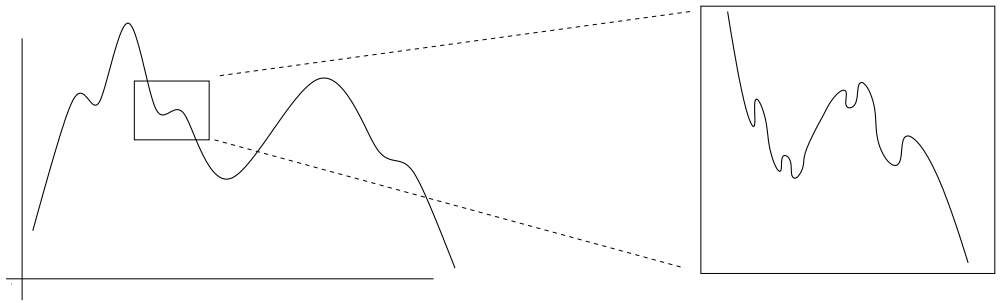
Define: Given the distribution P, what is the correct clustering?

- Sometimes this already implies what "the true K" is.
  - Example: "Clusters are disconnected components of the density."
  - Here K is uniquely defined.
- In most cases, have a parameter which directly or indirectly controls the number of clusters.
  - Example: "Clusters are disconnected components of level sets of the density." Here depending on the level t, different number of K possible.
  - Example: K-means
  - **Then we need a second definition: Given P, what is the "correct" number of clusters?**

Given a finite sample, now want to estimate K by some K_n.
Have to prove that estimator converges to the correct one.

The "correct" K, second approach

- Even if we know P, we can justify different numbers of clusters, depending on the "the scale" or "the resolution" we use to look at the distribution.
  Example: "Clusters correspond to modes of the density."



- We can even have infinitely many clusters of P.
- Now goal is different: On the finite sample, construct as many reliable clusters as possible
  - If n is small, only look for major clusters.
  - The larger n, the more clusters should be constructed.
  - Make sure that the clusters are not just sampling artifacts.

Idea: hierarchy of cluster core sets

Constructing cluster core sets:

- Fix a clustering algorithm $\mathcal{A}$ and its parameters
- Fix a threshold parameter $t \in [0, 1]$
- Repeatedly draw subsamples and cluster them using $\mathcal{A}$
- For each pair of points (x, y), evaluate frequency of ending up in the same cluster: $r(x, y) \in [0, 1]$
- Compute cluster core sets: sets of points $X_{i_1}, \ldots, X_{i_s}$ such that for all pairs $r(X_{i_u}, X_{i_v}) \geq t$.

If we vary the threshold parameter t, get a hierarchy of core sets. The threshold t controls the "confidence" we want to have.

What I like about core set approach

- Makes statements about individual clusters (or even: pairs of points)
- Allows "don't know statement" for large parts of the space.
- The threshold parameter t does not change the resolution (or number of clusters), but the "confidence" we have in the clusters
- The intuition that "stability" serves as a "measure of reliability" is more explicit.

Using core sets to choose K?

- have to choose a parameter K for the basis clustering algorithm
- then get a confidence statement whether this resolution leads to reliable clusters
- Core set approach does not determine the true number of clusters but just whether the given number of clusters leads to reliable results
- Only makes a statement whether core sets are reliable, no statement about remaining points.

Core sets vs. "traditional" stability

- Don't need to make statement about all clusters, can only talk about "significant ones"
- Circumvents the question about "the right K", don't need to come up with one value of K in the end
- Implicitly allows all parameters K for which the resulting clusters are reliable.
- Thus the criticism outlined above does not apply, but the reason is that we cheated a bit on the way. We simply changed the question from "finding the right number of clusters" to "finding all reliable clusters".
- But maybe this is what people actually need in practice?

Summary

- Many open issues and questions related to stability.
- It am reasonably convinced that stability indeed can be a very useful tool.
- But in all cases, the theory of why, and more importantly, when it works is missing!

Admittedly, I am quite puzzled about stability, don't really know what to believe and what not, and I am curious where all this is leading to!

References

Ben-David, S., Pál, D., and Simon, H.-U. (2007). Stability of k -means clustering. In N. H. Bshouty and C. Gentile (Eds.), *Conference on learning theory (COLT)* (pp. 20–34). Springer.

Ben-David, S., von Luxburg, U., and Pál, D. (2006). A sober look on clustering stability. In G. Lugosi and H. Simon (Eds.), *Proceedings of the 19th Annual Conference on Learning Theory (COLT)* (pp. 5 – 19). Springer, Berlin.

Ben-Hur, A., Elisseeff, A., and Guyon, I. (2002). A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing* (pp. 6 – 17).

Bertoni, A. and Valentini, G. (2007). Model order selection for bio-molecular data clustering. *BMC Bioinformatics*, *8*.

Bittner et al. (2000). Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, *406*, 536 – 540.

References (2)

Fridlyand, J. and Dudoit, S. (2001). *Applications of resampling methods to estimate the number of clusters and to improve the accuracy of a clustering method* (Technical Report No. 600). ???

Kerr, M. K. and Churchill, G. A. (2001). Bootstrapping cluster analysis: Assessing the reliability of conclusions from microarray experiments. *PNAS*, *98*(16), 8961 – 8965.

Krieger, A. and Green, P. (1999). A cautionary note on using internal cross validation to select the number of clusters. *Psychometrika*, *64*(3), 341 – 353.

Lange, T., Roth, V., Braun, M., and Buhmann, J. (2004). Stability-based validation of clustering solutions. *Neural Computation*, *16*(6), 1299 – 1323.

References (3)

Levine, E. and Domany, E. (2001). Resampling Method for Unsupervised Estimation of Cluster Validity. *Neural Computation*, *13*(11), 2573 – 2593.

Meila, M. (2003). Comparing clusterings by the variation of information. In *Colt* (p. 173-187).

Rakhlin, A. and Caponnetto, A. (2007). Stability of *k*-means clustering. In B. Schölkopf, J. Platt, and T. Hoffman (Eds.), *Advances in neural information processing systems 19.* MIT Press, Cambridge, MA.

Smolkin, M. and Ghosh, D. (2003). Cluster stability scores for microarray data in cancer studies. *BMC Bioinformatics*, *4*.