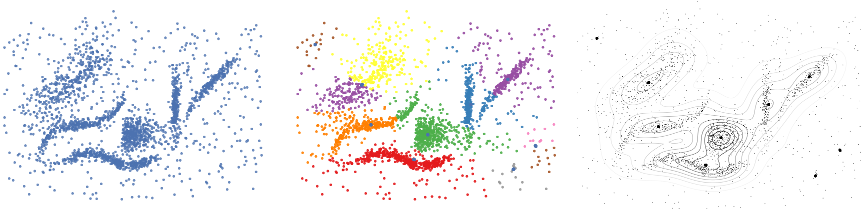


Clustertools

An overview and comparison of different clustering methods in Python

Luzie Helfmann, Johannes von Lindheim and Mattes Mollenhauer
Mathematical aspects in Machine Learning, SS17

Main goals

- Overview over various clustering methodologies
- Understand and compare approaches
- Implement specific algorithms in Python library

→ Clustertools package

Outline

- Clustertools package specifications

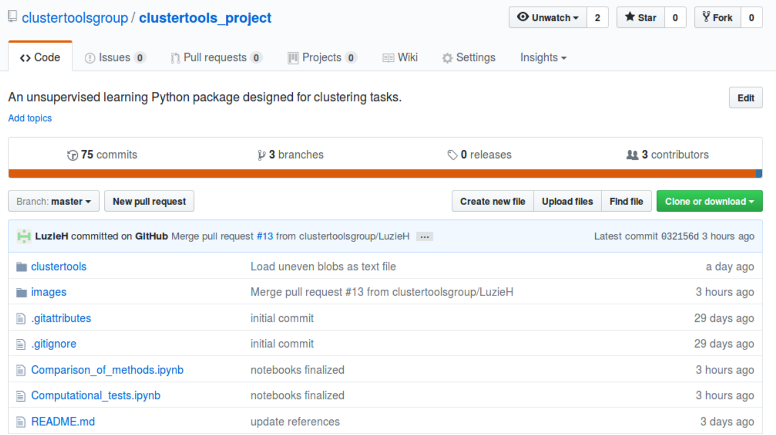
Outline

- Clustertools package specifications
- Methods - Package Contents
  - Distance-based methods
  - Density-based methods
  - Graph/similarity-based methods
  - Consensus Clustering

Outline

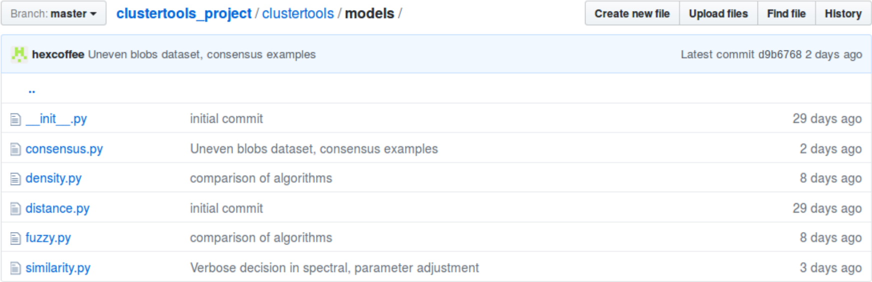
- Clustertools package specifications
- Methods - Package Contents
  - Distance-based methods
  - Density-based methods
  - Graph/similarity-based methods
  - Consensus Clustering
- Results
  - Trade-off
  - Comparison of the algorithms
  - Consensus Clustering

Our project: Clustertools Python package



https://github.com/clustertoolsgroup/clustertools_project

Clustertools Python package


Branch: master ▾ clustertools_project / clustertools / models /

Create new file | Upload files | Find file | History

hexcoffee Uneven blobs dataset, consensus examples — Latest commit d9b6768 2 days ago

..		
__init__.py	initial commit	29 days ago
consensus.py	Uneven blobs dataset, consensus examples	2 days ago
density.py	comparison of algorithms	8 days ago
distance.py	initial commit	29 days ago
fuzzy.py	comparison of algorithms	8 days ago
similarity.py	Verbose decision in spectral, parameter adjustment	3 days ago

- self-contained library for different clustering algorithms
- only *numpy* and *scipy* dependency
- *scikit-learn* oriented API

Package contents

Branch: **master** ▾ **clustertools_project / clustertools / models / similarity.py** Find file | Copy path

hexcoffee Verbose decision in spectral, parameter adjustment 59a679e 3 days ago

2 contributors

```python
#----------------------------
#SpectralClustering
#----------------------------

class SpectralClustering(object):

    def __init__(self, data, k, similarity_measure='gaussian', laplacian='normalized', metric='euclidean',
                 kmeans_params = None, verbose=True, **kwargs):
        '''
        Graph-based Spectral Clustering, normalized cuts algorithm by default (normalized graph Laplacian),
        runs KMeans on the eigenspace data. It is planned to implement arbitrary clustering algorithms
        to work on the lower dimensional eigenspace.
        See https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf

        Args:
            data: (n,d)-shaped two-dimensional ndarray or graph adjacency matrix
```

Clustertools API

```
from clustertools.models.similarity import AffinityPropagation
from clustertools.load_data import load_fuzzy_data

data = load_fuzzy_data()

clustering_object = AffinityPropagation(data,max_iter=100, damp=0.3, similarity_measure='squared_distance',
                                        metric='euclidean', sensitivity_weights='median', n_break_storage=5)
clustering_object.fit()
```

```
Constructing squared distance matrix
terminated by break condition
73 iterations until termination.
Finished after 0:00:54.642282
```

Clustertools API

```
from clustertools.models.similarity import AffinityPropagation
from clustertools.load_data import load_fuzzy_data

data = load_fuzzy_data()

clustering_object = AffinityPropagation(data,max_iter=100, damp=0.3, similarity_measure='squared_distance',
                                        metric='euclidean', sensitivity_weights='median', n_break_storage=5)
clustering_object.fit()
```

```
Constructing squared distance matrix
terminated by break condition
73 iterations until termination.
Finished after 0:00:54.642282
```

access all clustering information via object instance properties:

clustering_object.cluster_labels

Methods: Distance-based methods

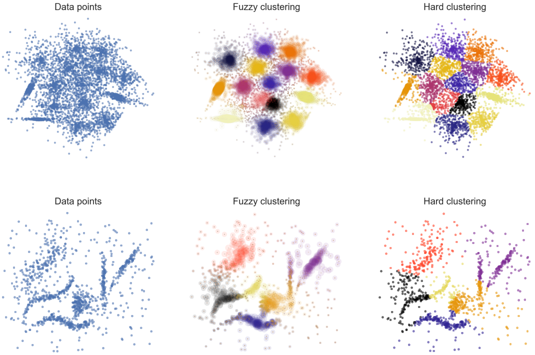


Figure: Fuzzy C-Means on two noisy data sets

- **Idea:** similarity of data points according to some distance measure

Methods: Distance-based methods

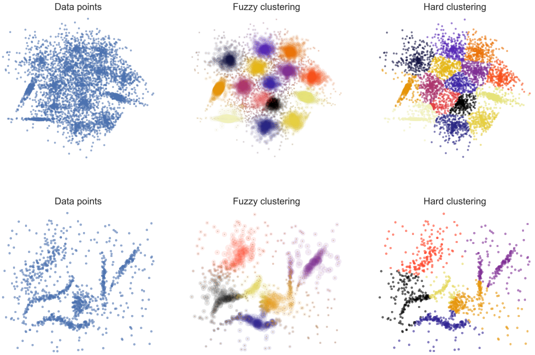


Figure: Fuzzy C-Means on two noisy data sets

- **Hard clustering:** strict membership of every data point to one cluster (implemented: K-Means, Regular Space Clustering)

- **Idea:** similarity of data points according to some distance measure

Methods: Distance-based methods

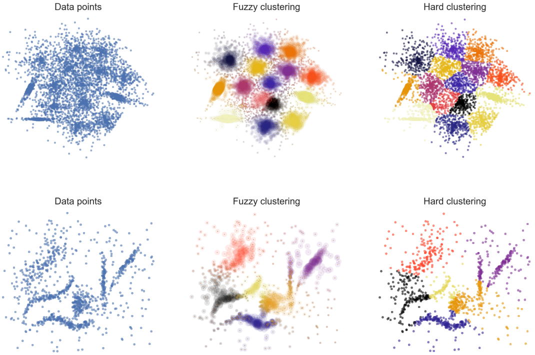


Figure: Fuzzy C-Means on two noisy data sets

- **Idea:** similarity of data points according to some distance measure

- **Hard clustering:** strict membership of every data point to one cluster (implemented: K-Means, Regular Space Clustering)
- **Fuzzy (soft) clustering:** a point has some degree of membership to every cluster (implemented: Fuzzy C-Means)

Methods: Density-based methods

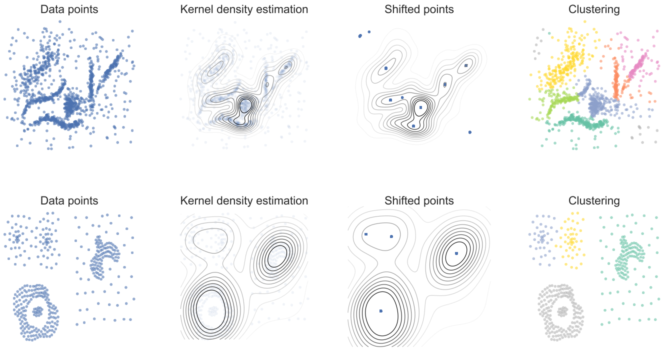


Figure: Mean Shift Algorithm on two data sets

- **Idea:** clusters are defined as regions of higher density
  (implemented: Mean Shift, DBSCAN)

Methods: Graph/similarity-based methods

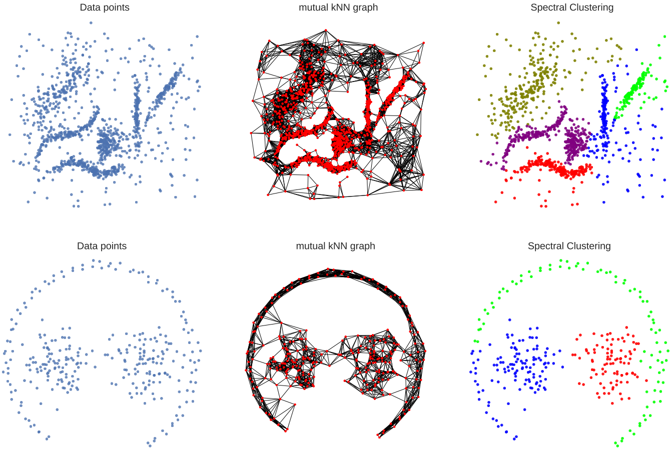


Figure: kNN-adjacency based Spectral Clustering

Methods: Graph/similarity-based methods

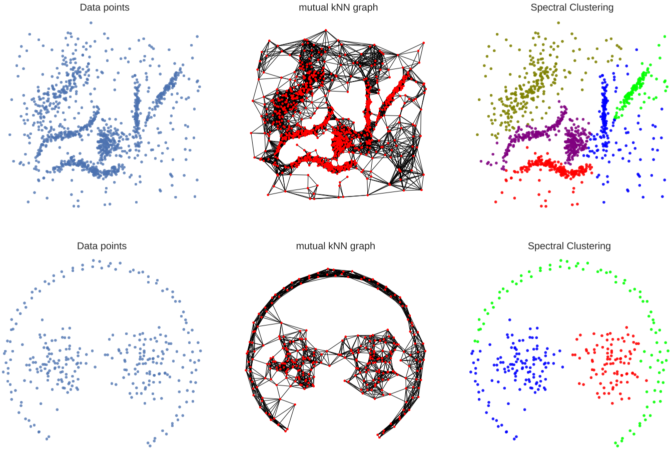


Figure: kNN-adjacency based Spectral Clustering

Implemented: Affinity Propagation, Spectral Clustering, Hierarchical Clustering

Methods: Consensus Clustering

Find clustering $\mathcal{C}^*$ for given set of existing clusterings $\Lambda = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$

$\mathcal{C}^*$ should represent a “common denominator”

Find clustering $\mathcal{C}^*$ for given set of existing clusterings $\Lambda = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$

$\mathcal{C}^*$ should represent a “common denominator”

Motivation:

- Robustness

Find clustering $\mathcal{C}^*$ for given set of existing clusterings $\Lambda = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$

$\mathcal{C}^*$ should represent a “common denominator”

Motivation:

- Robustness
- Knowledge reuse

Find clustering $\mathcal{C}^*$ for given set of existing clusterings $\Lambda = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$

$\mathcal{C}^*$ should represent a “common denominator”

Motivation:

- Robustness
- Knowledge reuse
- Distributed computing

Methods: Consensus Clustering

Find clustering $\mathcal{C}^*$ for given set of existing clusterings $\Lambda = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$

$\mathcal{C}^*$ should represent a “common denominator”

Motivation:

- Robustness
- Knowledge reuse
- Distributed computing
- Parameter search (own contribution)

Normalized mutual information between two random variables X, Y:

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}. \tag{1}$$

Methods: Normalized Mutual Information (NMI)

Normalized mutual information between two random variables X, Y:

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}. \quad (1)$$

Set $X = \mathcal{C}_i, Y = \mathcal{C}_j \Rightarrow$ estimation $\phi^{\mathrm{NMI}}(\mathcal{C}_i, \mathcal{C}_j)$ from data.

Normalized mutual information between two random variables X, Y:

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}. \tag{1}$$

Set $X = \mathcal{C}_i, Y = \mathcal{C}_j \Rightarrow$ estimation $\phi^{\mathrm{NMI}}(\mathcal{C}_i, \mathcal{C}_j)$ from data.

Objective function for consensus clustering (average NMI):

$$\phi^{\mathrm{ANMI}}(\mathcal{C}*, \Lambda) = \frac{1}{m}\sum_{i=1}^{m} \phi^{\mathrm{NMI}}(\mathcal{C}*, \mathcal{C}_i). \tag{2}$$

Normalized mutual information between two random variables X, Y:

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}}. \tag{1}$$

Set $X = \mathcal{C}_i, Y = \mathcal{C}_j \Rightarrow$ estimation $\phi^{\mathrm{NMI}}(\mathcal{C}_i, \mathcal{C}_j)$ from data.

Objective function for consensus clustering (average NMI):

$$\phi^{\mathrm{ANMI}}(\mathcal{C}*, \Lambda) = \frac{1}{m}\sum_{i=1}^{m}\phi^{\mathrm{NMI}}(\mathcal{C}*, \mathcal{C}_i). \tag{2}$$

Direct maximization / greedy approaches do not work :(

Freie Universität Berlin

Reclustering points:

- Hamming distance on points (count different labels)
- Recluster with affinity based algorithm

Reclustering points:

- Hamming distance on points (count different labels)
- Recluster with affinity based algorithm

Reclustering clusters:

- Jaccard distance on all clusters: $\#(C_i \cap C_j) \,/\, \#(C_i \cup C_j)$
- Assign every cluster to a meta-cluster
- Compete for points

Results: Trade-off

- **Issue I:** choice of parameters has big effect on outcome

Results: Trade-off

- **Issue I:** choice of parameters has big effect on outcome
- **Issue II:** some algorithms don’t scale well with n and d

Results: Trade-off

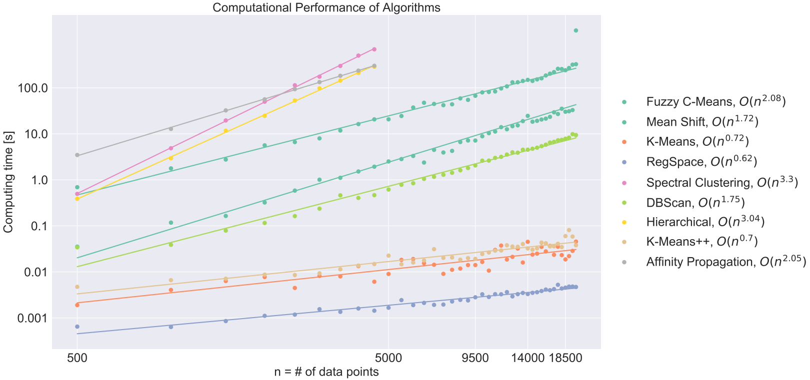


Figure: Computing time of algorithms vs. number of data points on a log-log plot

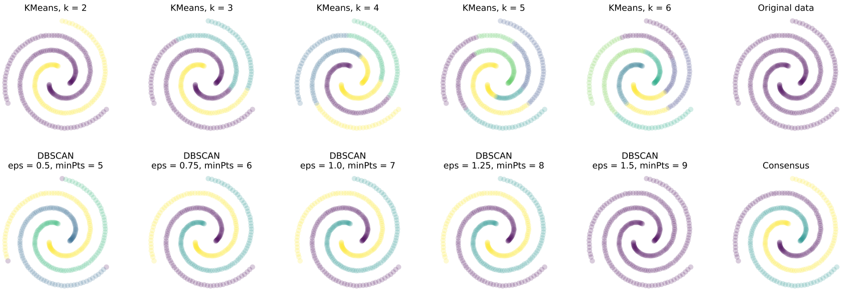
Results: Trade-off

- **Issue I:** choice of parameters has big effect on outcome
- **Issue II:** some algorithms don’t scale well with n and d
- **Issue III:** shape-dependent clustering, existence of outliers and noise

Results: Comparison of the algorithms

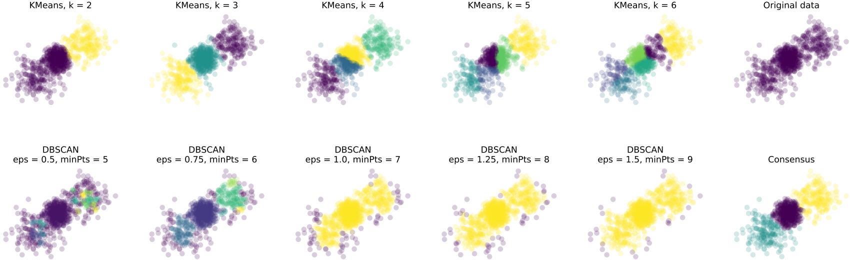
Method	Parameters	Comput. Complexity	Advantages	Limitations	Possible extensions
Affinity Propagation	Similarity matrix	$\mathcal{O}(n^2\ iter)$	Works on abstract similarity relations, exemplar weighting and sensitivity parameter tuning possible	Needs optimal configuration of iteration damping and sensitivity parameters	Hierarchical Affinity Propagation
DBScan	Density parameters ϵ, m_{points}	$\mathcal{O}(n^2)$	Noise/outlier detection, detection of highly nonlinear shapes	Highly sensitive to density parameters	HDBSCAN, input parameter estimators
Fuzzy C-Means	Number of clusters k and fuzzier m	$\mathcal{O}(nk^2d\ iter)$	Noise/outlier detection	k needs to be known/approximated in advance, strong dependence on chosen distance metric, not deterministic, might only converge to local minimum	estimation of number of clusters, Kernel C-Means, C-Means++
Hierarchical clustering	Number of clusters k and/or maximum linkage distance l	$\mathcal{O}(n^2 \log(n))$ in general, $\mathcal{O}(n^2 \log(n))$ for single-link and complete-link	Can detect highly nonlinear structures, only works on a distance measure between the points (observations themselves not needed)	The number of clusters or a stopping distance threshold, as well as the linkage function must be specified, depends heavily on the given distance function	Variety of different distance or linkage functions
K-Means K-Means++	Number of clusters k	$\mathcal{O}(nkd\ iter)$ (Lloyd iteration) worst case $iter = 2^{\Omega(\sqrt{2})}$	Computational speed, many results on convergence, complexity and objective function available	k needs to be known/approximated in advance, not deterministic, might only converge to local minimum	Kernel K-Means, K-Medioids
Mean Shift	Bandwidth parameter for kernel density estimation (kde)	$\mathcal{O}(n^2\ iter)$, kde doesn't scale well with d	no knowledge about shapes or number of clusters assumed, used a lot for image segmentation	too expensive for most applications	bandwidth estimators, code could be parallelized
Regular Space	Minimal distance ϵ_{min}	$\mathcal{O}(nk)$, worst case $\mathcal{O}(n^2)$ k is parameter dependent	Computational speed, useful for large-scale statespace discretizations	Restricted interpretability	Kernel extension
Spectral Clustering	Number of clusters k	$\mathcal{O}(n^3)$	Can detect highly nonlinear structures, works on abstract similarity graph	k has to be know/approximated, few results on different graph/adjacency types, strongly dependent on choice of graph, very slow	Landmark and/or sparse methods, more efficent eigensolvers

Results: Consensus Clustering - Good DBSCAN



- The consensus clusterer is able to extract the correct clustering from the good DBSCAN clusterings

Results: Consensus Clustering - Good KMeans



- The consensus clusterer finds the right three blobs from the KMeans clusterings

We propose:

1. Fit clustering algorithm with range of parameters
2. Produce consensus clustering
3. Pick NMI-maximizer

We propose:

1. Fit clustering algorithm with range of parameters
2. Produce consensus clustering
3. Pick NMI-maximizer

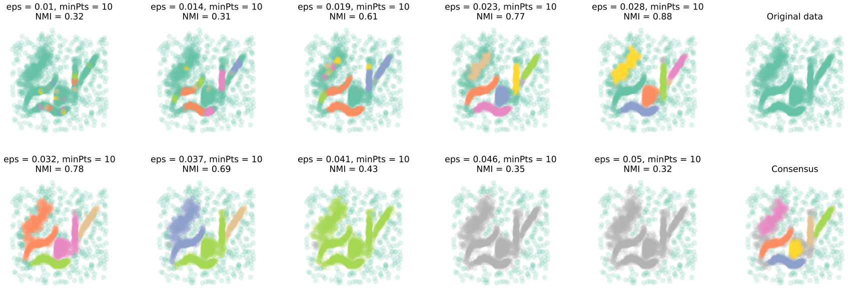
⇒ objective function for search of parameters, despite lack of labels

Results: Consensus Clustering - Parameter Search

We propose:

1. Fit clustering algorithm with range of parameters
2. Produce consensus clustering
3. Pick NMI-maximizer

⇒ objective function for search of parameters, despite lack of labels



Our presentation, paper (with more details and references to the literature) and Python package can be found on: https://github.com/clustertoolsgroup/clustertools_project

Disclaimer: Our K-Means, Regspace and DBSCAN implementations come from the project Markov Chains and Markov State Models with Prof. Noé.