CLUSTERING AND DENSITY ESTIMATION

Machine Learning for Autonomous Robots

Marc Otto[1] and Manuel Meder[2]
[1]DFKI, Robotics Innovation Center
[2]Robotics Group, University of Bremen

November 9, 2022 – Bremen, Deutschland

Intro

Unsupervised Learning

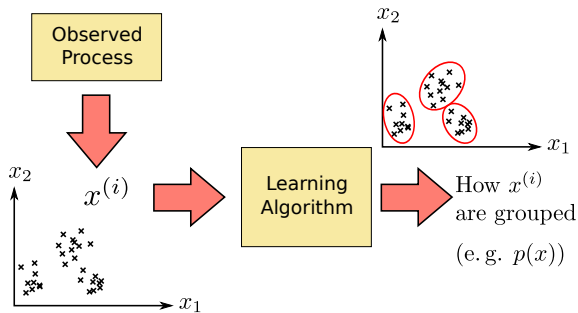
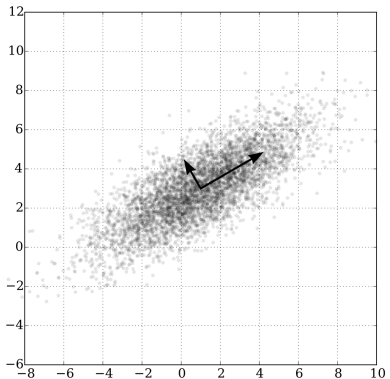


Figure: Unsupervised Learning Concept

Outputs (e.g. labels) are not available for any data point.

Unsupervised Learning

- Data clustering: dividing data into similar groups.
- Dimension reduction techniques: eliminating less important data in high dimensional spaces.



https://commons.wikimedia.org/wiki/File:GaussianScatterPCA.svg

Clustering

Clustering aims at partitioning n observations into k clusters. Observations that are assigned to the same cluster should be more "similar" than those belonging to different clusters.

- Requires a dissimilarity function defined on the observation space to measure similarity (e.g. euclidean distance)
- How to choose k? ($\rightarrow$ Exercise)
- There are $\frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}\binom{k}{i}i^n$ different clusterings
- $n = 100, k = 5 \rightarrow 10^{68}$. Thus: Intelligent ways of searching the clusterings space is required!

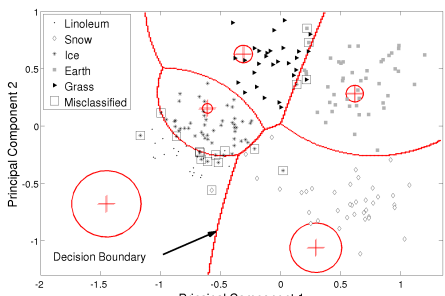
Examples

- Image segmentation
- Knowledge discovery in (any) gathered data (e.g., different soils a robot moves on)
- Compression (represent data by cluster centers)
- Group different objects for interaction with robot
- Outlier detection
- Grouping different movement types of humans (e.g., throws) for characterization and transfer onto robot

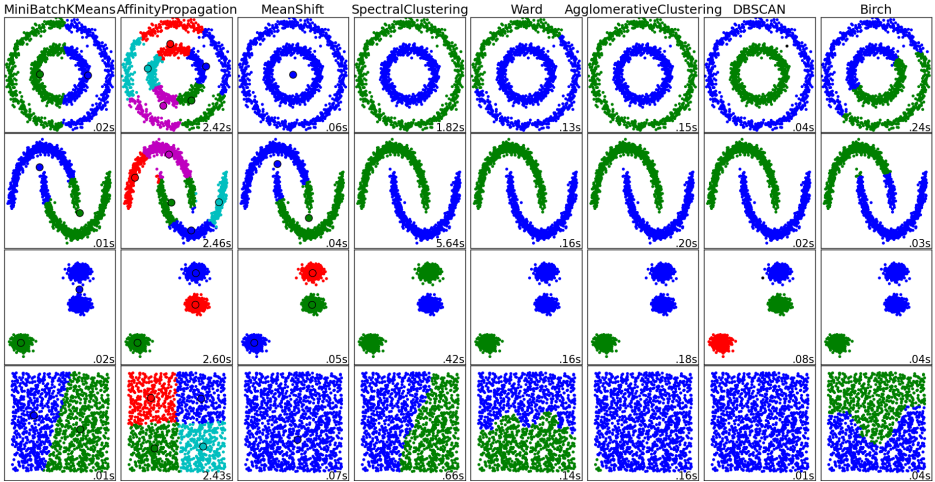
Application: Terrain Clustering

Clustering sensor data for autonomous terrain identification using time-dependency. Giguere, Philippe and Dudek, Gregory. http://dx.doi.org/10.1007/s10514-009-9114-2

- 3 angular velocities
- 3 accelerations
- 6 motor currents
- 23 time points



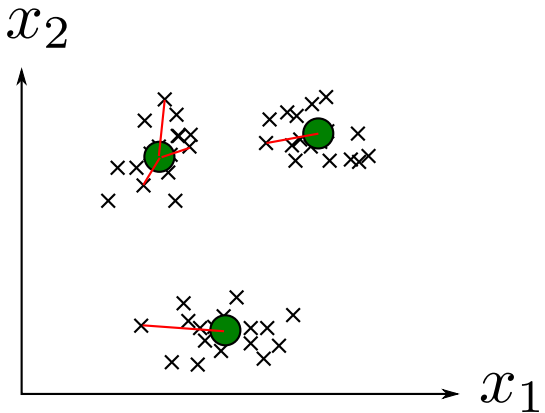
Scikit-learn Examples



http://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/_images/plot_cluster_comparison_1.png

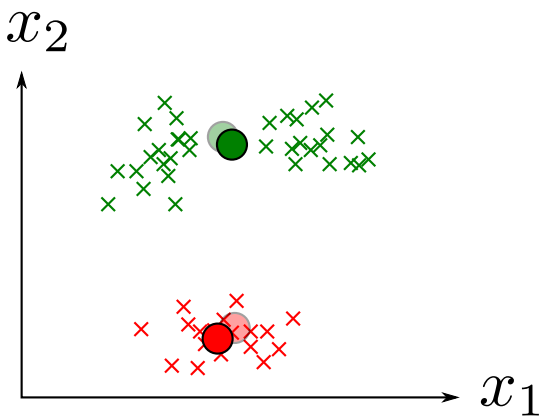
Centroid-Based C. (k-means)

k-means: Objective



- Let's cluster the data on the left...
- We are given $k = 3$ centers – where should they be located?
- What metric is minimized? Average euclidean distance from each point to its nearest center!

k-means: Algorithm



Algorithm:

1. Choose # of clusters ($k = 2$)
2. Randomly initialize cluster centers
3. Assign all datapoints to their nearest center
4. Relocate centers to "mean" of assigned datapoints
5. Repeat (3) and (4) until convergence

k-means: algorithm

Given n observation, assume k clusters exist:

1. Make initial guesses for each cluster mean (center) m_1, m_2, ..., m_k
2. Partition the observations based on the distances to cluster means:

$$C_i^l = \{x_p : ||x_p - m_i^l||_2 \leq ||x_p - m_i^j||_2 \ \ \forall 1 \leq j \leq K\}$$

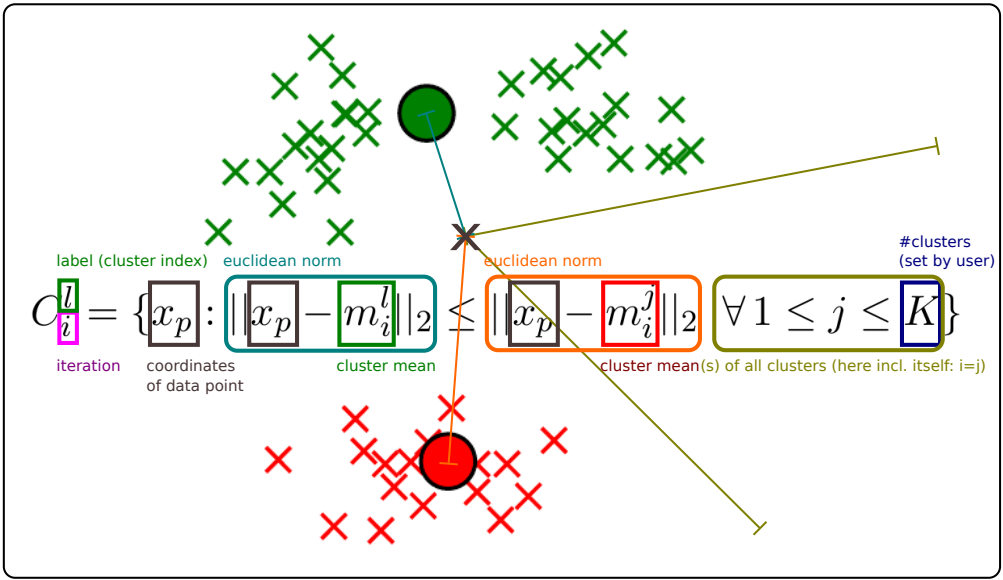
3. Replace m_i with the mean of all the samples in cluster l, i.e.,

$$m_{i+1}^l = \frac{1}{|C_i^l|} \sum_{x_p \in C_i^l} x_p$$

where $|C_i^l|$ is the number of data points in C_i^l

4. Go to 2 if not $C_i^l = C_{i-1}^l \forall l$

k-means: algorithm (partitioning formula)



k-means: Properties

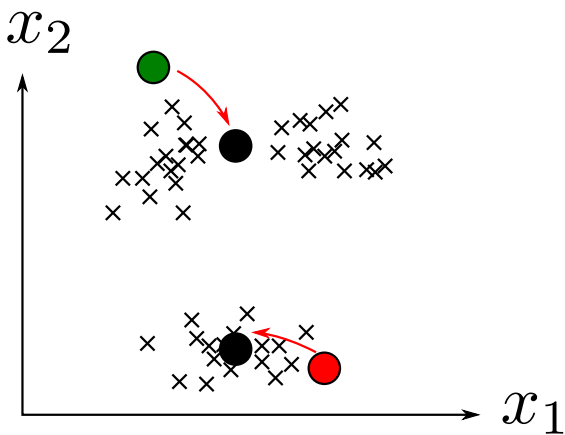
What are the pros of the Algorithm?

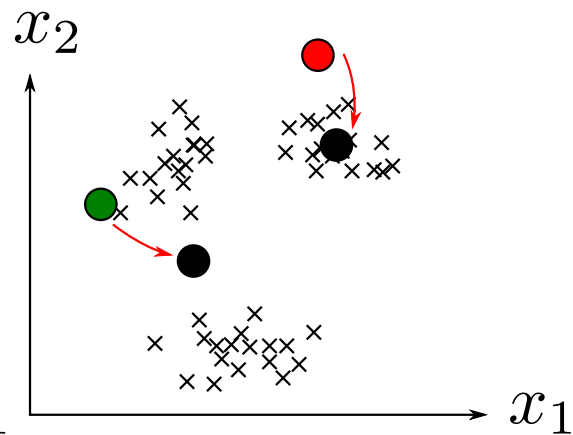
- ✓ Simple: easy to understand and to implement
- ✓ Efficient: $O(n)$ time complexity for n data-points
- ✓ Guaranteed convergence because it is a coordinate descent algorithm
- ✓ k-means is one of the most popular clustering algorithm

What are the cons of the Algorithm?

- ✗ The user needs to specify k.
- ✗ Sensitivity to outliers
- ✗ Sensitivity to initial selection of the means
- ✗ Final clustering is only locally optimal.
- ✗ Not suitable for discovering clusters that are NOT parts of hyper-ellipsoids.
- ✗ Cannot express uncertainty of cluster assignment

k-means: Properties (different initialization)





Distribution-Based C. (GMM)

Density Estimation: Motivation

- k-means assigns each datapoint to exactly one cluster
- Probabilistic clustering methods assign each datapoint to a cluster with a certain probability
- Probabilistic clustering methods are a subfield of the general concept of Density Estimation

Density Estimation

Density Estimation is the construction of an estimate of an unknown probability density function (pdf). This estimate is constructed based on a set of observed data points that are thought of as a random sample drawn from this pdf.

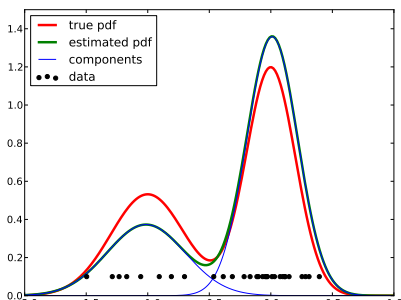
Density Estimation: Motivation

A density estimate $p(x, y, z)$ can be used e.g. for the following purposes:

- Prediction: e. g. expectation of $p(x|y, z)$, $p(x, z|y)$, $p(z|y), \ldots$
- Generative Model: generate more data from the same distribution as the observed data by sampling from the estimated pdf.
- Confidence measures (i. e. "belief"): using the variance at a given point
- System-identification: Build mathematical models of dynamical systems from measured data

Density Estimation: Tools

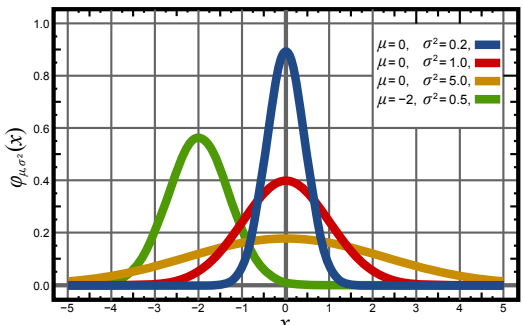
Estimate pdf of some observed data.



Approaches:

- Histograms
- *Non-parametric:* Kernel-density estimation
- *Parametric:* Gaussian Mixture Models (GMMs)

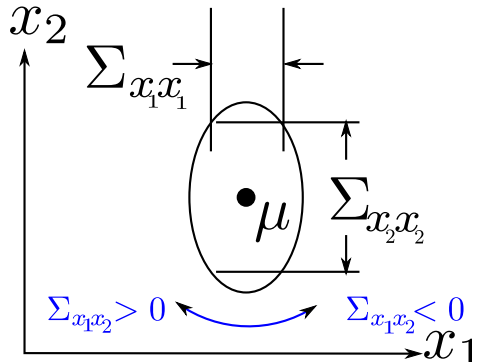
Multivariate Gaussian: Overview



source: wikipedia

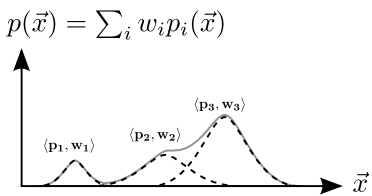
$$p(x) = \mathcal{N}_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-(x-\mu)^2/(2\sigma^2)}$$

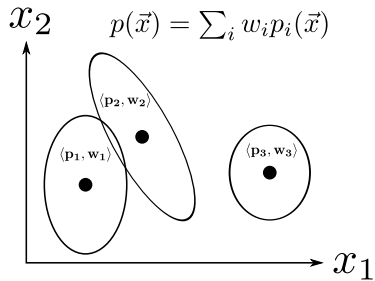
Multivariate Gaussian: Overview



$$p(x) = \mathcal{N}_{\mu,\Sigma}(x) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Gaussian Mixture Model (GMM)





Density Estimation with a Single Gaussian

- Given data points $x_1, \ldots, x_n$ that are assumed to come from single gaussian with unknown mean μ and covariance matrix Σ
- How can we estimate the parameter θ? $(\theta = (\mu, \Sigma))$
- Maximum Likelihood: Maximize the likelihood L of parameters θ for observations $x_1, \ldots, x_n$: $\theta^* = \arg\max_\theta L(\theta \mid x_1, \ldots, x_n)$

Likelihood

Likelihood is how likely the pdf p for parametrization θ will generate $x_1, \ldots, x_n$:

$$L(\theta \mid x_1, \ldots, x_n) = p_\theta(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_\theta(x_i)$$

Density Estimation with a Single Gaussian

- Likelihood in the case of single Gaussian: $L(\theta \mid x_1, \ldots, x_n) = \prod_{i=1}^{n} p_\theta(x_i) = \prod_{i=1}^{n} \mathcal{N}_{\mu,\Sigma}(x_i)$

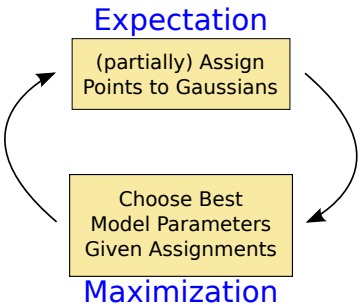
Max L. Estimator for a single Gaussian

There exists a closed form solution:
$\hat{\mu} = \frac{1}{n}\sum_i x_i, \qquad \hat{\Sigma} = \frac{1}{n}\sum_i (x_i - \mu)(x_i - \mu)^T$

Density Estimation with GMMs

- Parameters in the case of a GMM with k Gaussians are: $\theta = (\mu_1, \Sigma_1, w_1, \ldots, \mu_k, \Sigma_k, w_k)$
- Task: Determine the best set of GMM parameters θ to fit a set of observations $x_1, \ldots, x_n$
- Likelihood in the case of GMM model with k Gaussians:
$$L(\theta \mid x_1, \ldots, x_n) = \prod_{i=1}^{n} p_\theta(x_i) = \prod_{i=1}^{n} \left(\sum_{j=1}^{k} w_j \cdot \mathcal{N}_{\mu_j, \Sigma_j}(x_i) \right)$$
- Maximum Likelihood estimate cannot be determined in closed form
- Use iterative optimization algorithm called "Expectation Maximization" to find a (local) maximum of the Likelihood function

EM Algorithm: Concept



General EM Algorithm:

E-Step (Expectation)

Use the current parameters values to evaluate the (partial) memberships

M-Step (Maximization)

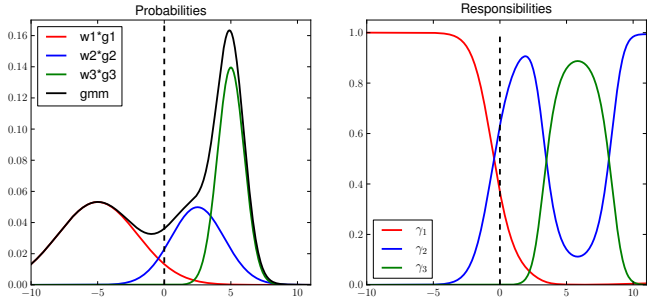
Estimate the distribution parameters to maximize the likelihood of the data.

EM Algorithm: E-Step for GMM

The E-step estimates the share γ_{ij} of the observation x_i in the j^{th} Gaussian.

E-Step: Calculate $\gamma_{ij} = \frac{w_j \cdot \mathcal{N}_{\mu_j, \Sigma_j}(x_i)}{\sum_{j'=1}^{k} w_{j'} \cdot \mathcal{N}_{\mu_{j'}, \Sigma_{j'}}(x_i)}$

- In contrast to k-means ($\gamma_{ij} \in \{0, 1\}$), in EM γ_{ij} can take on any arbitrary value in $[0, 1]$
- γ_{ij} is called the "responsibility" of the j^{th} Gaussian for the i^{th} datapoint



EM Algorithm: M-Step for GMM

The M step chooses

- new model parameters (μ_j, Σ_j) for each Gaussian that maximize the likelihood of the data belonging to it (weighted by the responsibilities).
- the mixture coefficients w_j so that gaussians which are responsible for many data points get larger mixture coefficients.

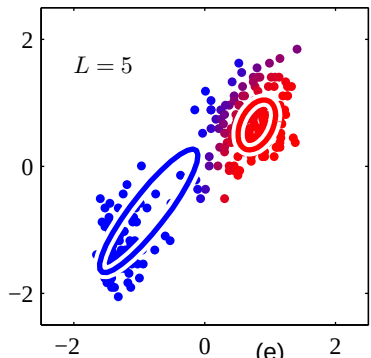
Maximum Likelihood Estimator for Gaussian

$$\hat{\mu} = \frac{1}{n} \sum_i x_i$$
$$\hat{\Sigma} = \frac{1}{n} \sum_i (x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

EM Updates (GMM)

$$n_j = \sum_i \gamma_{ij}$$
$$\hat{\mu}_j = \frac{1}{n_j} \sum_i \gamma_{ij} x_i$$
$$\hat{\Sigma}_j = \frac{1}{n_j} \sum_i \gamma_{ij} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T$$
$$w_j = n_j / n$$

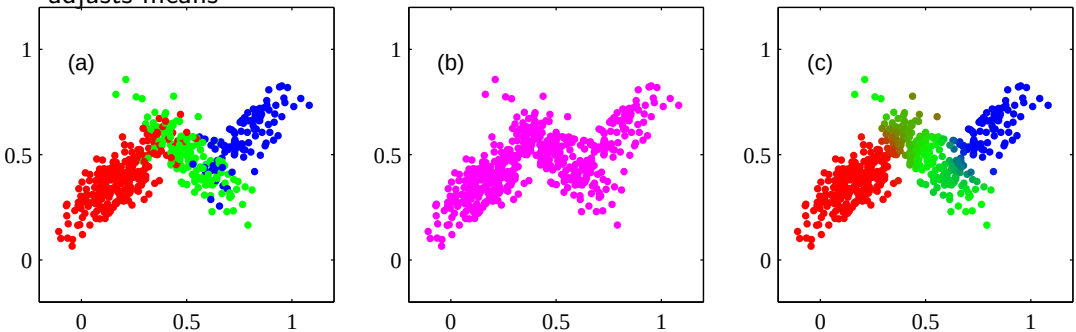
EM algorithm on Old Faithful dataset



M-Step, Iteration 5

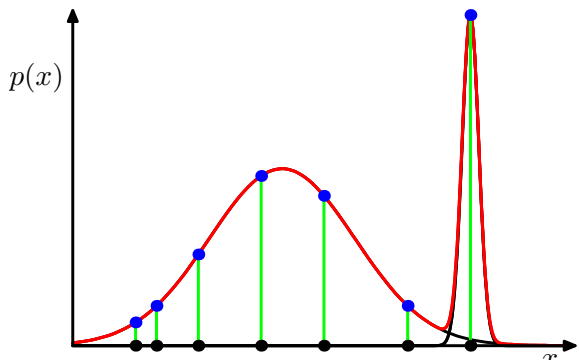
EM: Properties

- EM increases likelihood of GMM at each step
- EM is a very general algorithm, not only for GMM
  - Forward-backward algorithm for Hidden Markov Models
  - Positron emission tomography (PET) scan reconstruction
  - In principle anywhere Maximum Likelihood estimates are needed
- Learning covariance matrices of GMMs is more powerful than k-means, which only adjusts means



EM+GMM: Issues / Solutions

- Potential issues:
  - Gaussian collapses to a point (likelihood $\rightarrow$ infinity)



EM+GMM: Issues / Solutions

- Potential issues:
  - Gaussian collapses to a point (likelihood $\rightarrow$ infinity)
  - Two Gaussians converge to identical parameters
- Approaches
  - When Gaussians get too small or too near each other, randomly redistribute them.
  - Choose initial parameters reasonably (e.g. use k-means for finding initial means)
  - Bayesian version of EM (takes into account prior probability of model parameters)
- Open Issues: Number of clusters has to be defined. Only hyper-ellipsoid clusters can be modeled. Dependence on initialization.

Density-Based C. (DBSCAN)

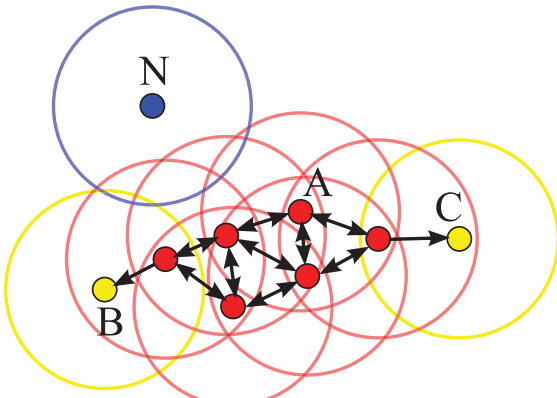
DBSCAN: Motivation

Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise

- Circumvent definition or exhaustive testing for parameter k
- Avoid/reduce influence of noise on the clusters
- Model arbitrary shapes
- Only parameters, which can be more easily chosen/optimized
- Enable scalable calculation for large scale databases

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA, 1996, pp. 226–231.

DBSCAN: Visualization



Schubert, Erich, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN." ACM Transactions on Database Systems 42, no. 3 (August 24, 2017): 1-21. https://doi.org/10.1145/30683

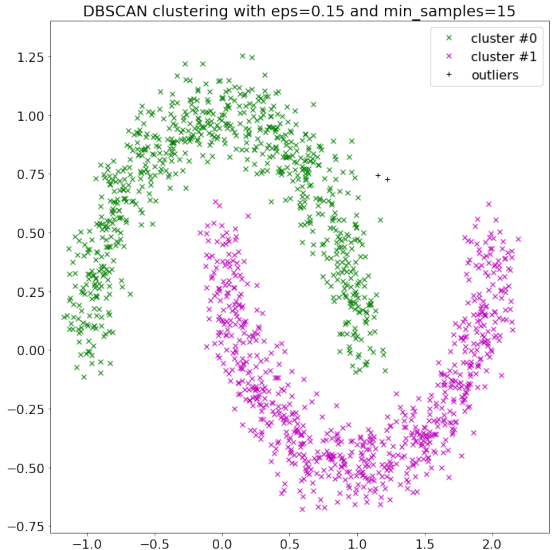
DBSCAN: Algorithm I

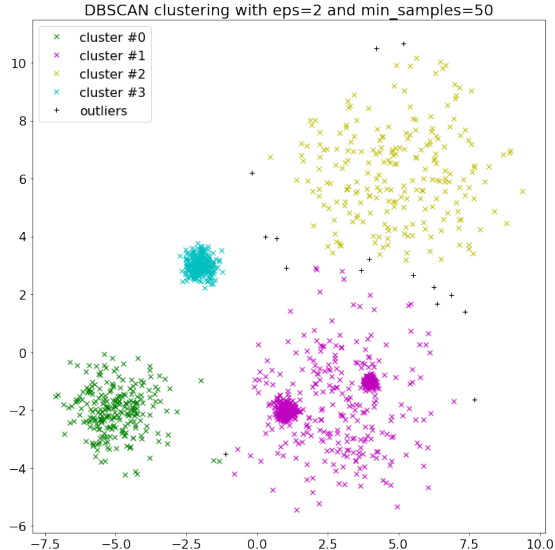
```
def DBSCAN(data, eps, MinPts):
""" Main method iterating over all points """
  C = 1  # current cluster id
  # set cluster to 0 (noise) and visited to False
  init(data)  # for each data point
  for point in data:
    if point.visited:
      continue  # handle next point
    point.visited = True
    # List of all points in eps distance to the point
    # excluding point
    NeighborPts = regionQuery(data, point, eps)
    # point is counted for the number of minimum samples
    if len(NeighborPts) + 1 >= MinPts:
      point.cluster = C
      expandCluster(data, NeighborPts, C, eps, MinPts)
      C += 1  # next cluster id
```

DBSCAN: Algorithm II

```
def expandCluster(data, NeighborPts, C, eps, MinPts):
""" Add all fitting points to cluster C """
  while len(NeighborPts) > 0:
    point = NeighborPts.pop(0)
    if not point.visited:
      point.visited = True
      new_neighbors = regionQuery(data, point, eps)
      if len(new_neighbors) + 1 >= MinPts:
        NeighborPts.extend(new_neighbors)
      # border point if len(new_neighbors) + 1 < MinPts
    if point.cluster == 0:
      point.cluster = C
```

DBSCAN: Examples



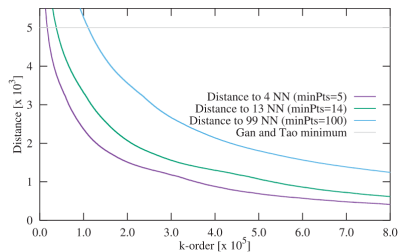


DBSCAN: Properties

- Cluster are regions with dense object distribution
- Every object of a cluster must be in a region within other objects (density threshold)
- Density threshold consists of
  - ϵ, Environment radius
  - *MinPts*, Minimum number of objects within environment
- Heuristic for the Hyper-Parameters: k-distance Graph

Heuristic for the Hyper-Parameters: k-distance Graph

- MinPts $\geq$ Dimension $+ 1$, or 4 for $2D$
- Plot k-distance graph for different $k = \text{MinPts} - 1$
- $\text{dist}_k(p)$ = maximum distance to its k nearest neighbors
- x-Axes: points sorted by their dist_k value
- Determine threshold point p, where the graph forms an "elbow"
- $eps(k) = \text{dist}_k(p)$, but as small as possible



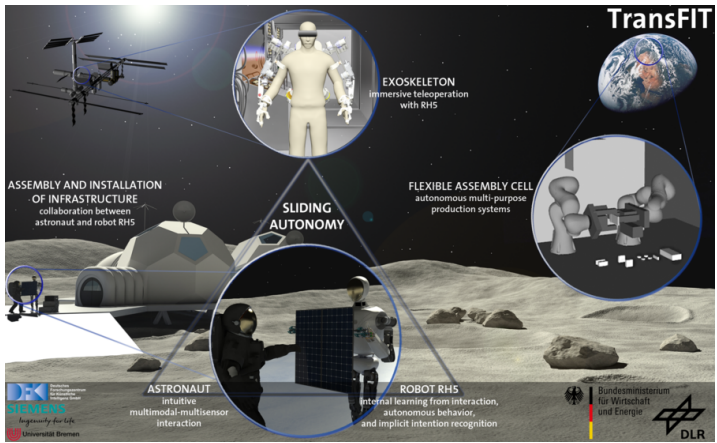
Schubert, Erich; Sander, Jörg; Ester, Martin; Kriegel, Hans Peter; Xu, Xiaowei (July 2017). "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". ACM Trans. Database Syst. 42 (3): 19:1–19:21.

DBSCAN - Pros and Cons

- ✓ Number of clusters (or Gaussians) not defined beforehand as in k-means or GMM
- ✓ Arbitrary (connected) shapes can be modeled (*even in arbitrary data spaces*)
- ✓ Noise is modeled
- ✓ DBSCAN* is deterministic (DBSCAN possibly not on border points)
- ✓ Expert domain knowledge can be used to select ϵ and *MinPts*
- ✗ Dependence on the distance measure (holds for all clustering algorithms)
- ✗ Large differences in densities cannot be modeled.
- ✗ $O(n^2) \rightarrow$ can be improved with indexing to $O(n \log n)$

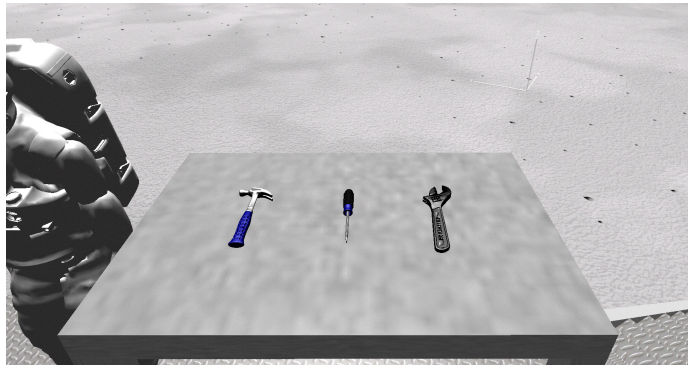
TransFIT

Flexible Interaction for infrastructures establishment by means of teleoperation and direct collaboration; transfer into industry 4.0



https://robotik.dfki-bremen.de/typo3temp/_processed_/4/1/csm_20180219_MY_TransFIT_logo_FINAL_V3_fd46420b26.png

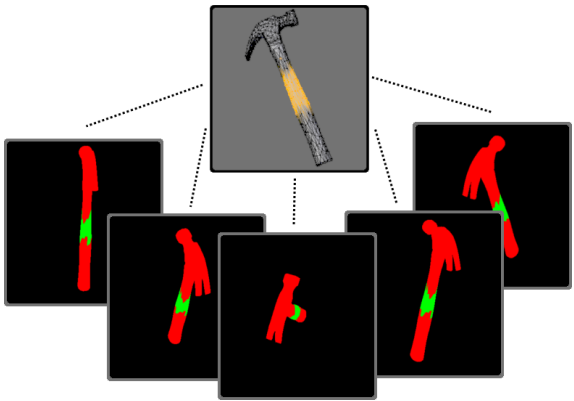
Grasping objects - but where?



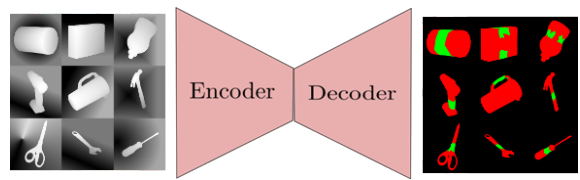
- pose (location & orientation) depends on context (task)
- pose depends on geometry of object
- unknown objects should also work

R. Detry and J. Papon and L. Matthies, Task-oriented grasping with semantic and geometric scene understanding, *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3266-3273, 2017.

Filtering for task context



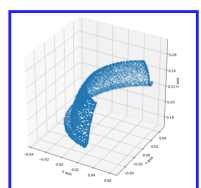
- Object from YCB dataset
- Locations are selected by human

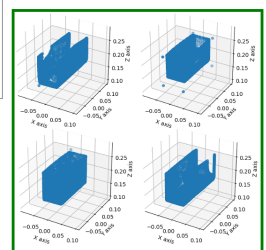


- Autoencoder is trained to learn context locations
- Mapping from depth data to filter mask

Search for geometric optimal grasp pose

Agglomerative hierarchical clustering



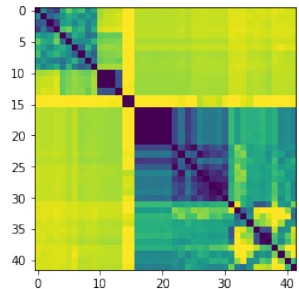


Search for geometric optimal grasp pose

- Generation of point cloud samples (Grasp shapes) from known models
- Samples are gripper dependent
- Grasp shapes (grasp relevant parts) are independent of objects
- Clustering of similar object parts from different objects

Clustering for dictionary of prototypes

Dissimilarity measure: Kernel Density Estimation (KDE)



Non-linear transform of dissimilarities:

- Similar shapes are pushed closer together
- Dissimilar shapes spread apart
- Well-suited geometrical space for clustering
  $\Rightarrow$ Clustering with normalized graph cuts

$$d_{\text{new}} = \exp\left(-\frac{d^2}{\sigma}\right)$$

Normalized Graph Cut

Graph construction:

- Each grasp shape is a node ($v \in V$)
- Edges weights are the transformed dissimilarities ($e_{ij} \in E$)
- Cuts divide V into two disjoint sets A and B

Inter-Cluster distances (cost of cut):

$$cut(A, B) = \sum_{i \in A, j \in B} e_{ij}$$

Association within the sets:

$$assoc(A, V) = \sum_{i \in A, j \in V} e_{ij}$$

(Summed weight of all edges that touch A)

Normalized Graph Cut

Minimization of:

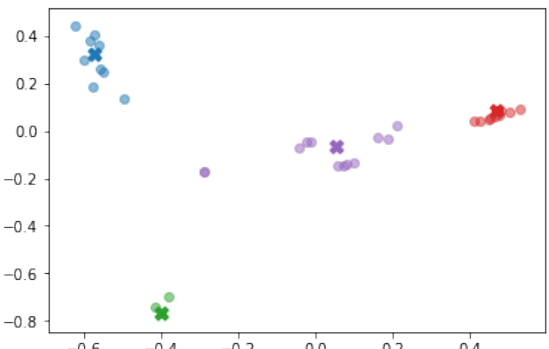
$$cut_{\text{norm}} = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

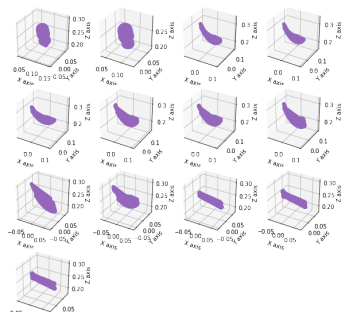
- Denominator noramlizes the cut cost
  $\Rightarrow$ Thus penalizes very small clusters
- Approximation of the optimization problem (related to spectral partitioning[1])

[1] https://towardsdatascience.com/spectral-clustering-for-beginners-d08b7d25b4d8

Visualization of clusters:

2D approximation of the data using Kernel PCA:

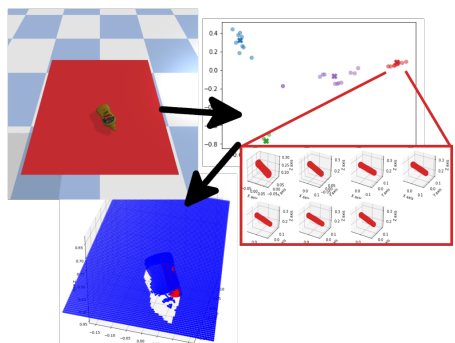




- Selection of cluster representatives

Grasp pose estimation

- Matching of prototypes to scene point cloud
- Using score from Kernel Density Estimation (KDE)
- Pose sampling using Markov Chain Monte Carlo (MCMC)



Thank You!

Please feel free to ask questions in the forums.