Clustering Algorithms and High Dimensional Data

Max Mihailescu

December 4th, 2019

Outline

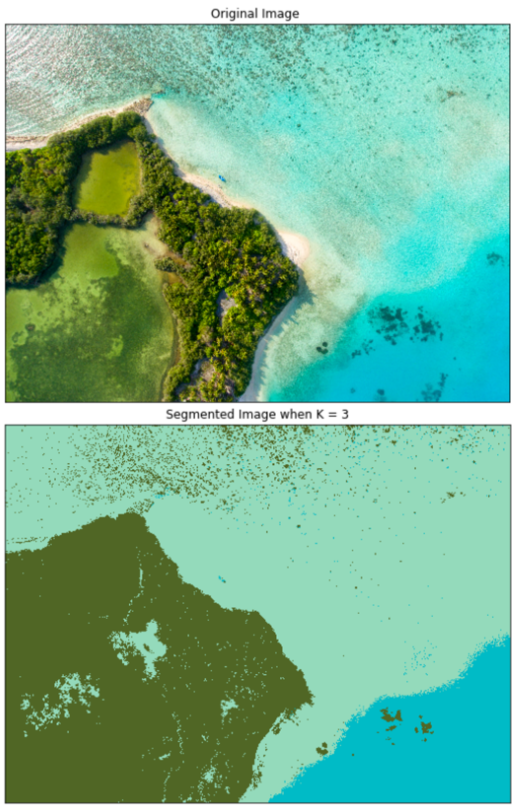
Introduction

Given a set of data-points, the goal of clustering is to group similar points together and have a high dissimilarity between the groups.

A data-point is characterized by a feature vector $X = \{x_1, x_2, ..., x_d\}$, where d is its dimension.

Various notions of similarity:

- Distance between objects
- Density around a given point
- Distribution of features



[1]

[1] https://towardsdatascience.com/introduction-to-image-segmentation-with-k-means-clustering-83fd0a9e2fc3

K-Means

Goal: Partition the data into k sets, so as to minimize the squared distances between points within a cluster.

K-Means

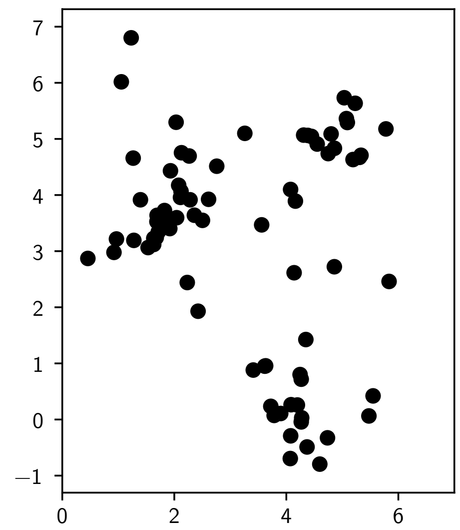
Parameters: *num_clusters*

```
Initialize num_clusters cluster centers
assign points to cluster based on nearest center
while not centres converged do
    Recalculate centers
    Reassign points to new cluster
end
```

K-Means

Parameters: *num_clusters*

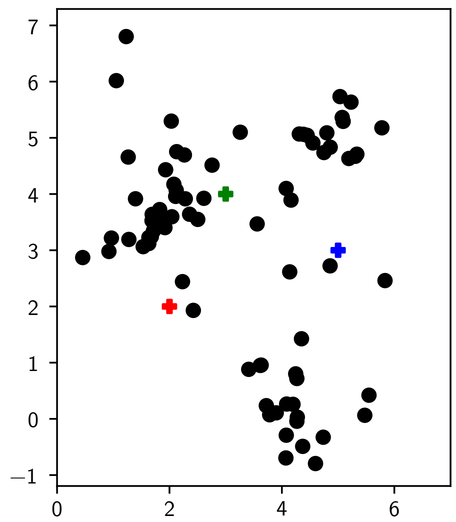
```
Initialize num_clusters cluster centers
assign points to cluster based on nearest center
while not centres converged do
    Recalculate centers
    Reassign points to new cluster
end
```



K-Means

Parameters: *num_clusters*

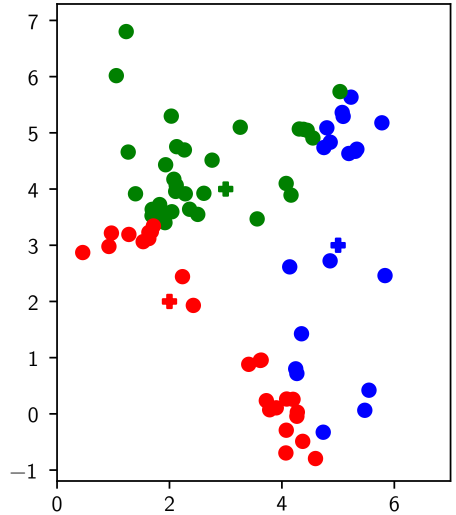
Initialize *num_clusters* cluster centers
assign points to cluster based on nearest center
while *not centres converged* **do**
    Recalculate centers
    Reassign points to new cluster
end



K-Means

Parameters: *num_clusters*

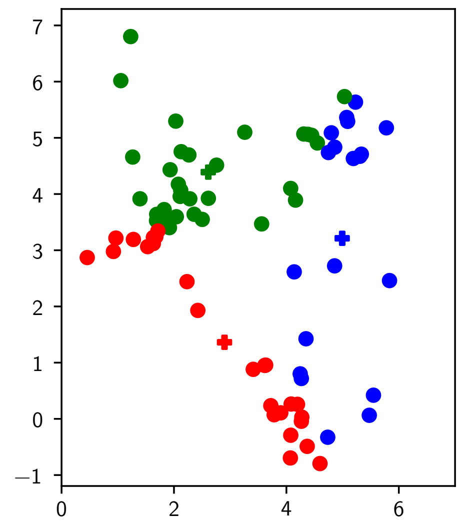
Initialize *num_clusters* cluster centers
assign points to cluster based on nearest center
while *not centres converged* **do**
    Recalculate centers
    Reassign points to new cluster
end



K-Means

Parameters: *num_clusters*

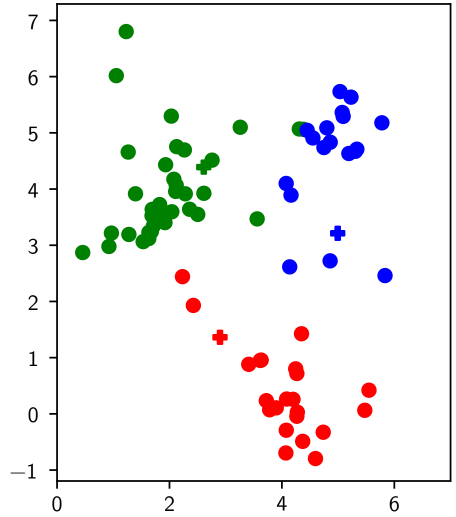
Initialize *num_clusters* cluster centers
assign points to cluster based on nearest center
while *not centres converged* **do**
    Recalculate centers
    Reassign points to new cluster
end



K-Means

Parameters: *num_clusters*

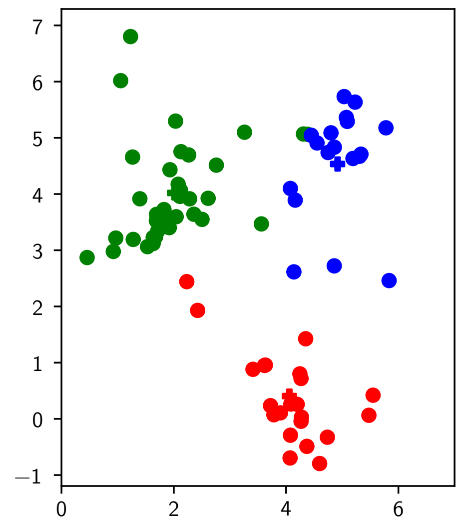
Initialize *num_clusters* cluster centers
assign points to cluster based on nearest center
while *not centres converged* **do**
    Recalculate centers
    Reassign points to new cluster
end



K-Means

Parameters: *num_clusters*

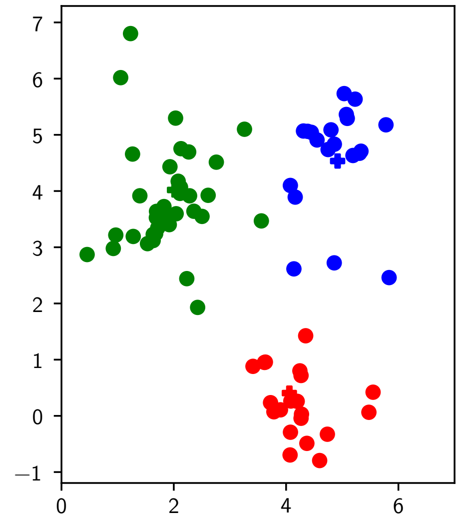
Initialize *num_clusters* cluster centers
assign points to cluster based on nearest center
while *not centres converged* **do**
    Recalculate centers
    Reassign points to new cluster
end



K-Means

Parameters: *num_clusters*

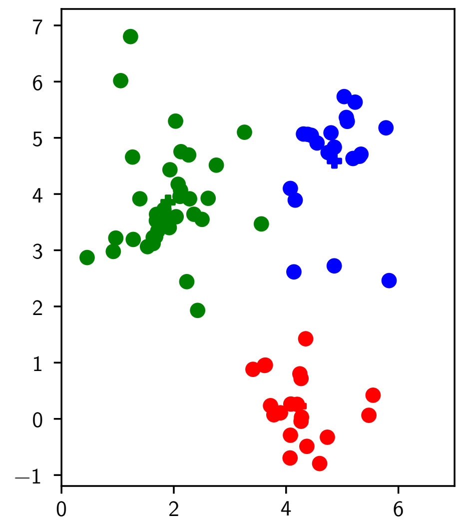
Initialize *num_clusters* cluster centers
assign points to cluster based on nearest center
while *not centres converged* **do**
    Recalculate centers
    Reassign points to new cluster
end



K-Means

Parameters: *num_clusters*

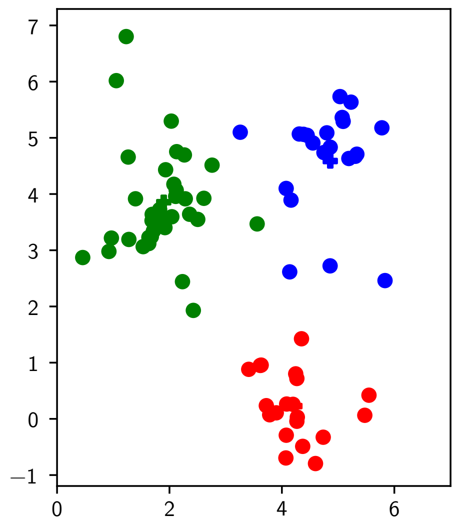
Initialize *num_clusters* cluster centers
assign points to cluster based on nearest center
while *not centres converged* **do**
    Recalculate centers
    Reassign points to new cluster
end



K-Means

Parameters: *num_clusters*

Initialize *num_clusters* cluster centers
assign points to cluster based on nearest center
while *not centres converged* **do**
  Recalculate centers
  Reassign points to new cluster
end



K-Means

- Number of clusters is an input parameter
- Based on a notion of distance
- Tends to find spherical clusters of the same size
- Unstable with respect to initial starting points

DBScan

Goal: Find connected regions in space that are highly populated.

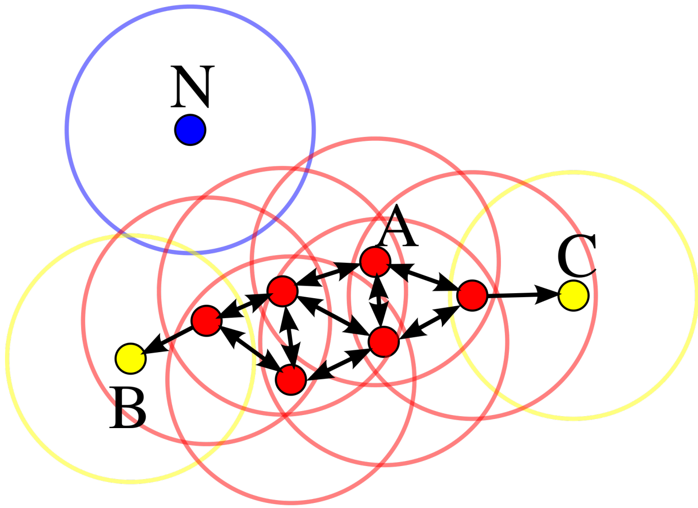
---

[a] Chire, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=17045963

DBScan

Goal: Find connected regions in space that are highly populated.

- Core points: Have at least *minPts* neighbors within distance *eps*
- Directly reachable points: At least one core point within distance *eps*

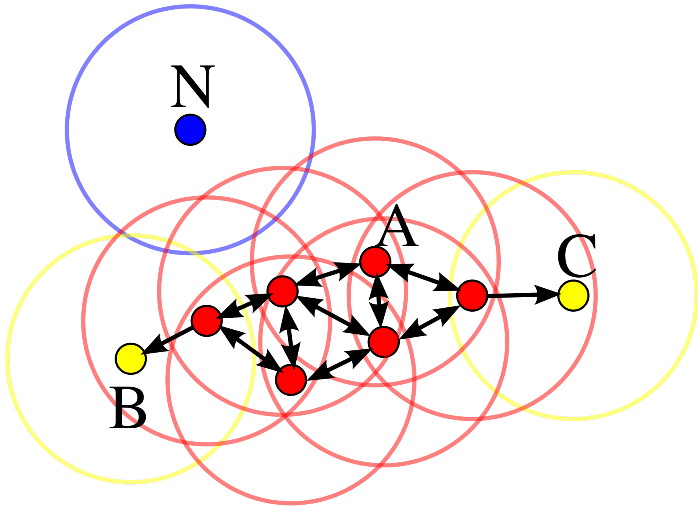


[a] Chire, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=17045963

DBScan

Goal: Find connected regions in space that are highly populated.

- Core points: Have at least *minPts* neighbors within distance *eps*
- Directly reachable points: At least one core point within distance *eps*



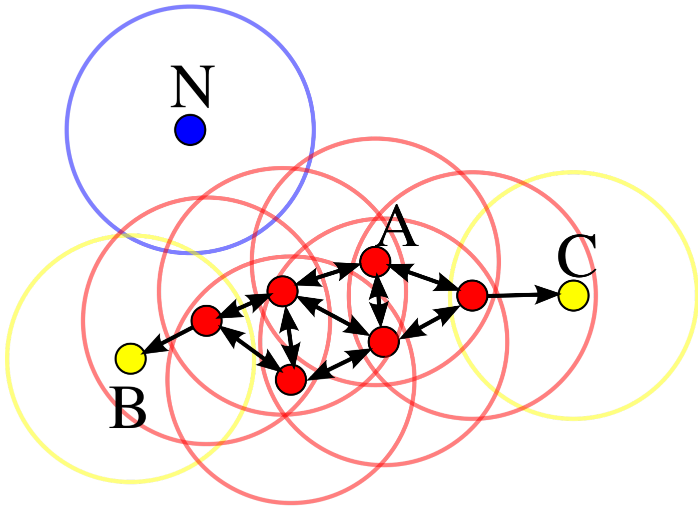
B is reachable from *A*, if there exists a path of following directly reachable points.

[a] Chire, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=17045963

DBScan

Goal: Find connected regions in space that are highly populated.

- Core points: Have at least *minPts* neighbors within distance *eps*
- Directly reachable points: At least one core point within distance *eps*



B is reachable from A, if there exists a path of following directly reachable points.
B and C are density connected if, they are reachable from a common point A.

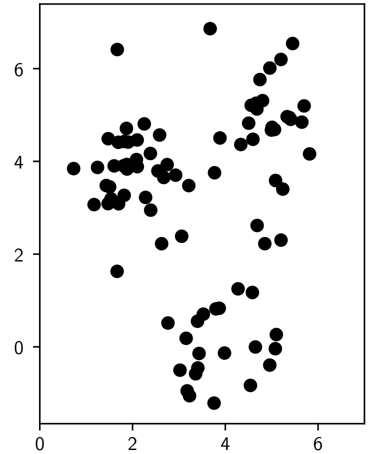
---

[a] Chire, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=17045963

DBScan

Parameters: *minPts*, *eps*

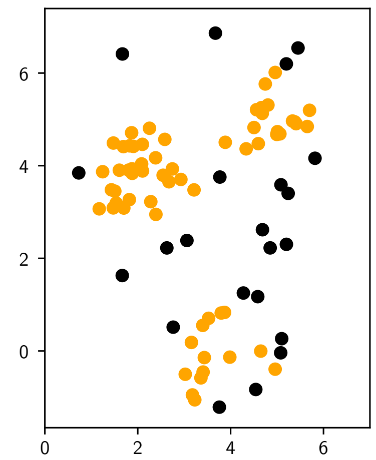
1. Find all neighbors closer than *eps* of every point
2. Identify core points with more than *minPts* neighbors
3. Create clusters of neighboring core points
4. Assign the remaining points to a nearby cluster or noise



DBScan

Parameters: *minPts*, *eps*

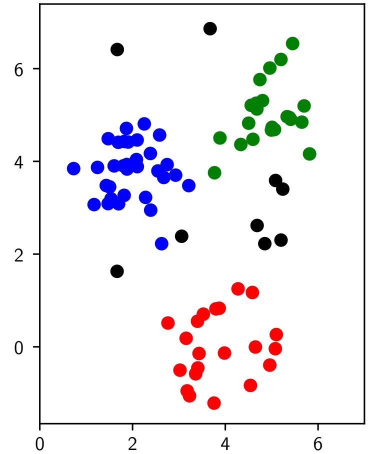
1. Find all neighbors closer than *eps* of every point
2. Identify core points with more than *minPts* neighbors
3. Create clusters of neighboring core points
4. Assign the remaining points to a nearby cluster or noise



DBScan

Parameters: *minPts*, *eps*

1. Find all neighbors closer than *eps* of every point
2. Identify core points with more than *minPts* neighbors
3. Create clusters of neighboring core points
4. Assign the remaining points to a nearby cluster or noise



DBScan

- Groups by point density in a neighborhood that is controlled using input parameters
- Finds arbitrarily shaped clusters
- Has a notion of noise
- Generally stable between runs
- Parameters require knowledge of the data

The Curse of Dimensionality

Data quickly becomes sparse in high dimensions:

- Consider a problem with 50 dimensions of binary features and 10^{12} observations.
- Then we still only have examples for $\frac{10^{12}}{2^{50}} \approx 0.089\%$ of categories.

The Curse of Dimensionality

Data quickly becomes sparse in high dimensions:

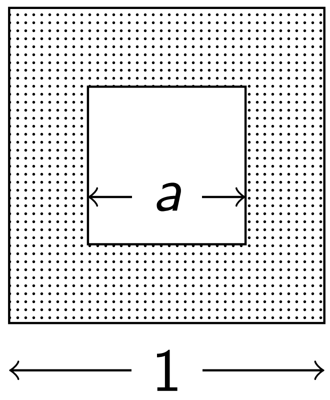
- Consider a problem with 50 dimensions of binary features and 10^{12} observations.
- Then we still only have examples for $\frac{10^{12}}{2^{50}} \approx 0.089\%$ of categories.

Problem 1: It becomes impossible to correctly estimate distributions.

The Curse of Dimensionality

Our intuition fails in higher dimensions:

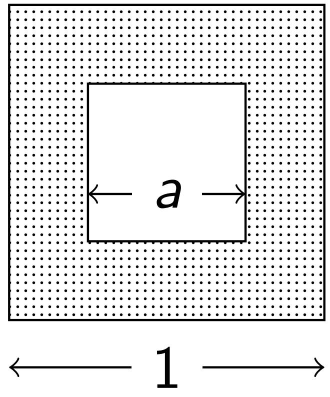
- Consider uniformly distributed data on a cube in d dimensions with side length 1.
- How big can we make a so that more than 95% of the probability mass lies inside the dotted region?



The Curse of Dimensionality

Our intuition fails in higher dimensions:

- Consider uniformly distributed data on a cube in d dimensions with side length 1.
- How big can we make a so that more than 95% of the probability mass lies inside the dotted region?

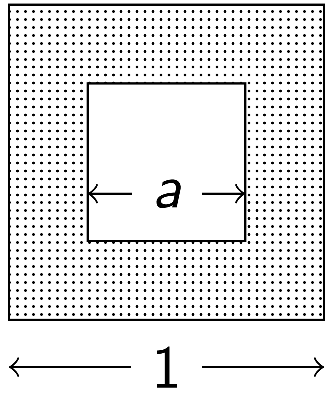


$$0.95 \leq P(X \in dotted)$$

The Curse of Dimensionality

Our intuition fails in higher dimensions:

- Consider uniformly distributed data on a cube in d dimensions with side length 1.
- How big can we make a so that more than 95% of the probability mass lies inside the dotted region?

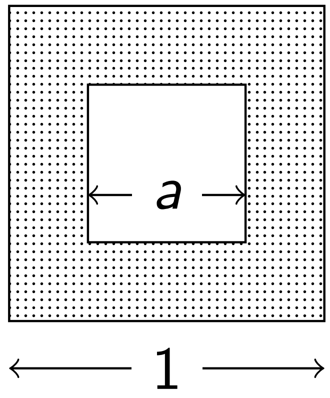


$$0.95 \leq P(X \in dotted) = P(X \in bigcube) - P(X \in smallcube) = 1 - a^d$$

The Curse of Dimensionality

Our intuition fails in higher dimensions:

- Consider uniformly distributed data on a cube in d dimensions with side length 1.
- How big can we make a so that more than 95% of the probability mass lies inside the dotted region?

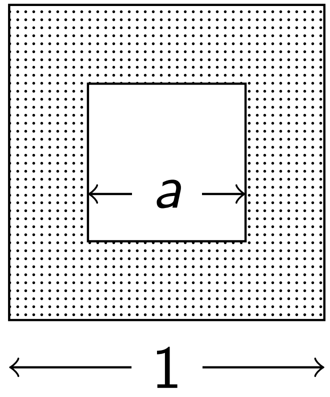


$$0.95 \leq P(X \in dotted) = P(X \in bigcube) - P(X \in smallcube) = 1 - a^d$$
$$\Leftrightarrow a \leq \sqrt[d]{0.05}$$

The Curse of Dimensionality

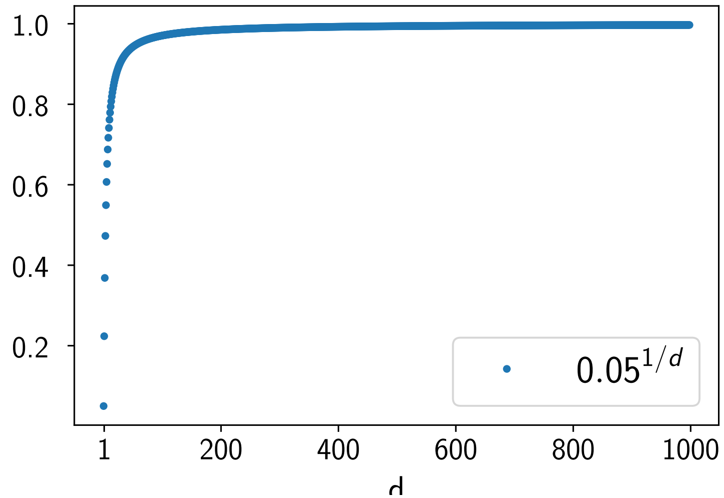
Our intuition fails in higher dimensions:

- Consider uniformly distributed data on a cube in d dimensions with side length 1.
- How big can we make a so that more than 95% of the probability mass lies inside the dotted region?



$$0.95 \leq P(X \in dotted) = P(X \in bigcube) - P(X \in smallcube) = 1 - a^d$$

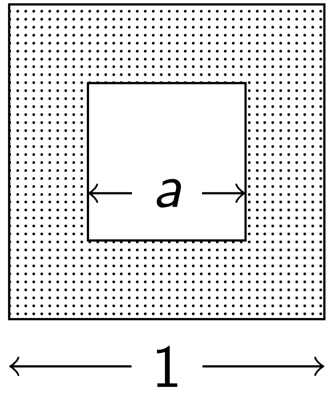
$$\Leftrightarrow a \leq \sqrt[d]{0.05}$$



The Curse of Dimensionality

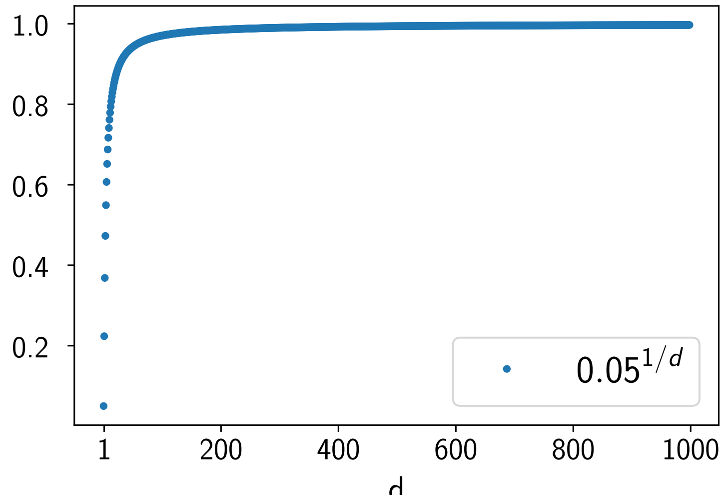
Our intuition fails in higher dimensions:

- Consider uniformly distributed data on a cube in d dimensions with side length 1.
- How big can we make a so that more than 95% of the probability mass lies inside the dotted region?



$$0.95 \leq P(X \in dotted) = P(X \in bigcube) - P(X \in smallcube) = 1 - a^d$$

$$\Leftrightarrow a \leq \sqrt[d]{0.05}$$



Problem 2: Regions with a high density can be empty.

Optigrid

A grid based approach to overcome the problems of distance and density in high dimensional spaces.

Hinneburg and Keim, *Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering*

Goal: Partition the space into grid-cells such that dense regions do not get cut in half.

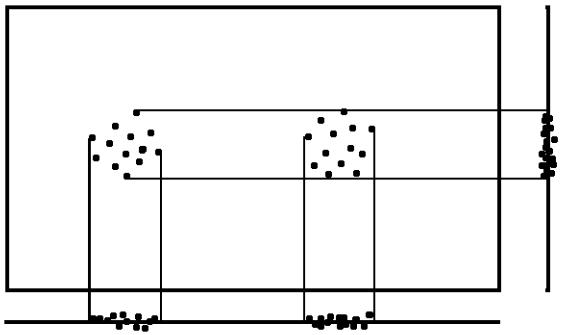
Optigrid

A grid based approach to overcome the problems of distance and density in high dimensional spaces.

Hinneburg and Keim, *Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering*

Goal: Partition the space into grid-cells such that dense regions do not get cut in half.

Key Observation: If the density at a point in one coordinate is low, it is low at all points in the high dimensional space that get projected onto that point.



Optigrid

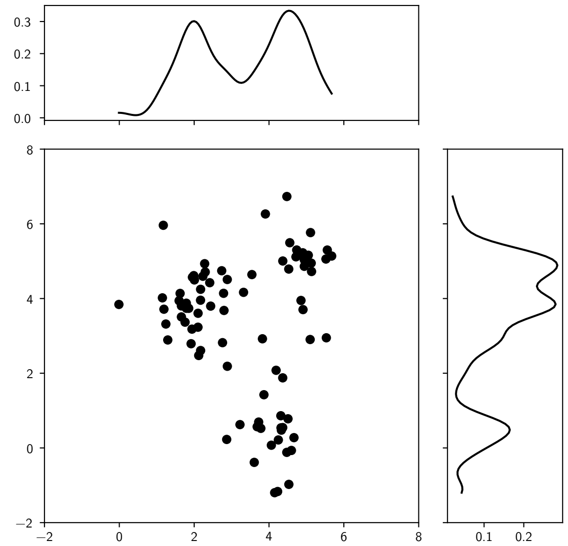
Input: Data
Parameters: *q*, *maxCutScore*, *noiseLevel*

Calculate Projections to lower dimensional space
Approx. Densities and find peaks above *noiseLevel*
Find q best cutting planes with score $\leq$ *maxCutScore*
if *cutting planes were found* **then**
    Divide data into subgrids $C_1, \ldots C_n$
    foreach *subgrid* **do**
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end

Optigrid

Input: Data
Parameters: q, $maxCutScore$, $noiseLevel$

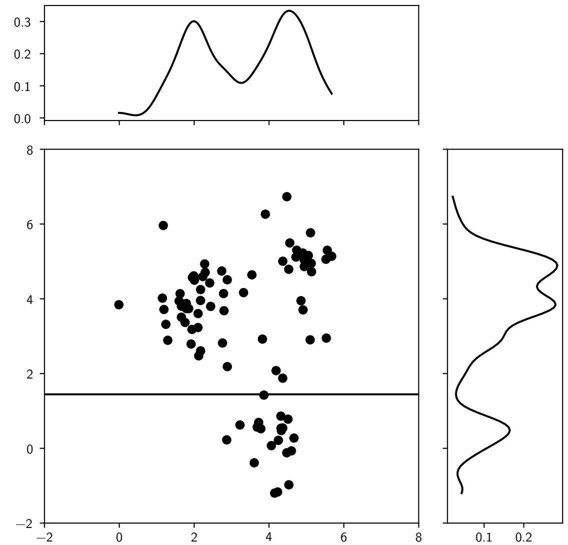
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above $noiseLevel$
Find q best cutting planes with score $\leq maxCutScore$
if *cutting planes were found* **then**
    Divide data into subgrids $C_1, \ldots C_n$
    foreach *subgrid* **do**
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end



Optigrid

Input: Data
Parameters: q, $maxCutScore$, $noiseLevel$

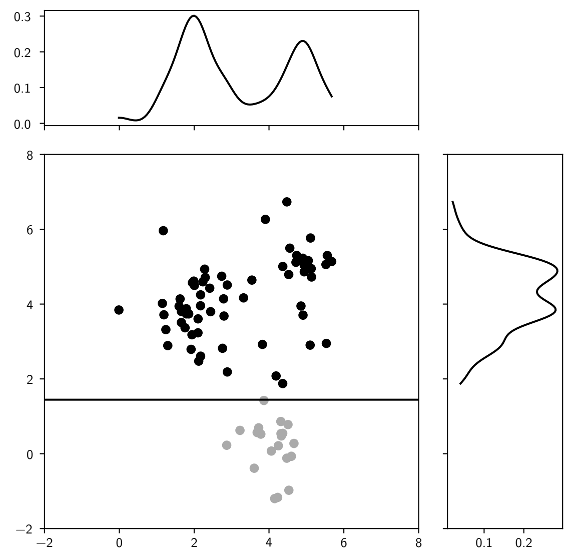
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above $noiseLevel$
Find q best cutting planes with score $\leq maxCutScore$
if *cutting planes were found* **then**
    Divide data into subgrids $C_1, \ldots C_n$
    foreach *subgrid* **do**
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end



Optigrid

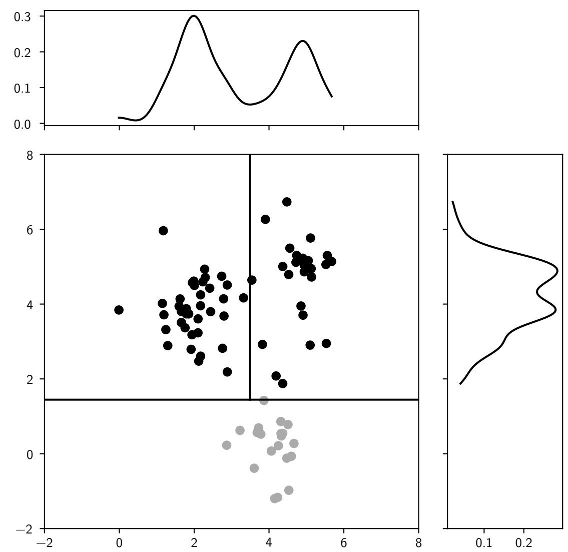
Input: Data
Parameters: q, $maxCutScore$, $noiseLevel$

```
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above noiseLevel
Find q best cutting planes with score ≤ maxCutScore
if cutting planes were found then
    Divide data into subgrids C_1, ...C_n
    foreach subgrid do
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end
```



Optigrid

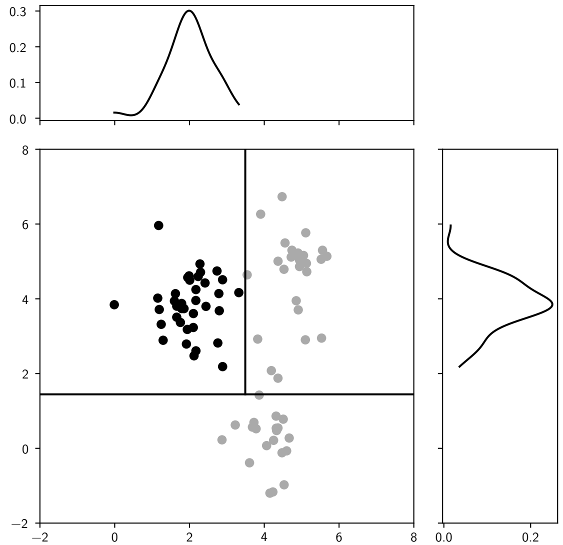
```
Input: Data
Parameters: q, maxCutScore, noiseLevel
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above noiseLevel
Find q best cutting planes with score ≤ maxCutScore
if cutting planes were found then
    Divide data into subgrids C_1, ...C_n
    foreach subgrid do
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end
```



Optigrid

Input: Data
Parameters: *q*, *maxCutScore*, *noiseLevel*

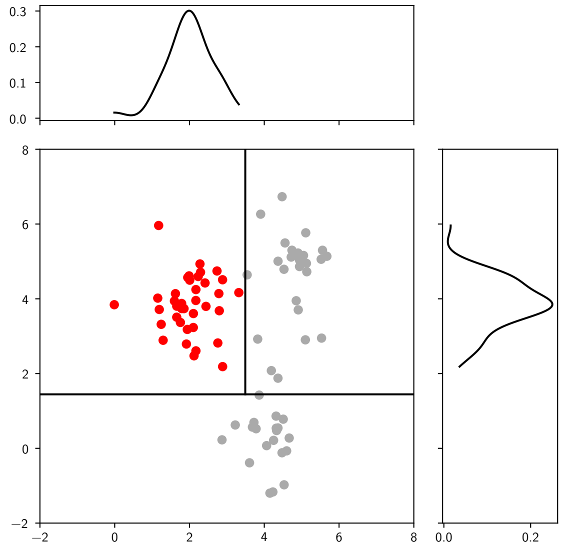
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above *noiseLevel*
Find *q* best cutting planes with score $\leq$ *maxCutScore*
if *cutting planes were found* **then**
    Divide data into subgrids $C_1, \ldots C_n$
    foreach *subgrid* **do**
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end



Optigrid

Input: Data
Parameters: *q*, *maxCutScore*, *noiseLevel*

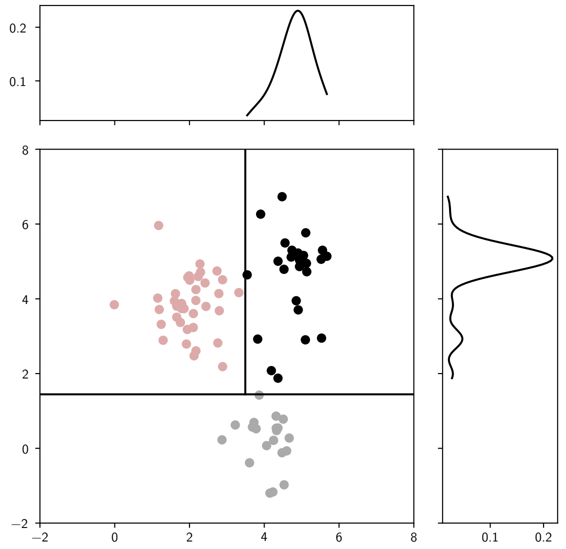
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above *noiseLevel*
Find q best cutting planes with score $\leq$ *maxCutScore*
if *cutting planes were found* **then**
    Divide data into subgrids $C_1, \ldots C_n$
    foreach *subgrid* **do**
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end



Optigrid

Input: Data
Parameters: *q*, *maxCutScore*, *noiseLevel*

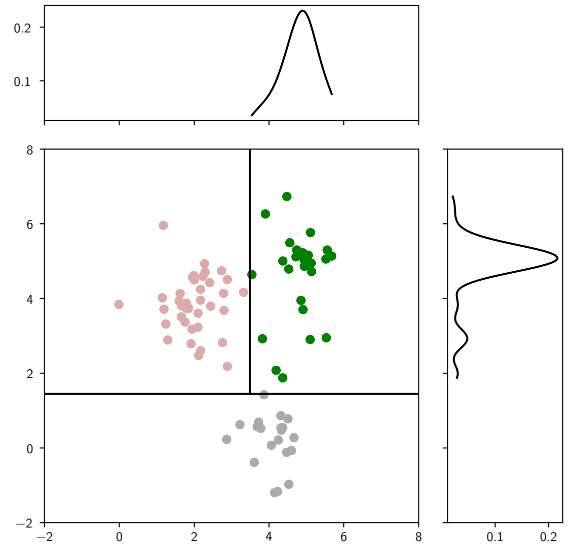
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above *noiseLevel*
Find *q* best cutting planes with score $\leq$ *maxCutScore*
if *cutting planes were found* **then**
    Divide data into subgrids $C_1, \dots C_n$
    foreach *subgrid* **do**
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end



Optigrid

Input: Data
Parameters: q, $maxCutScore$, $noiseLevel$

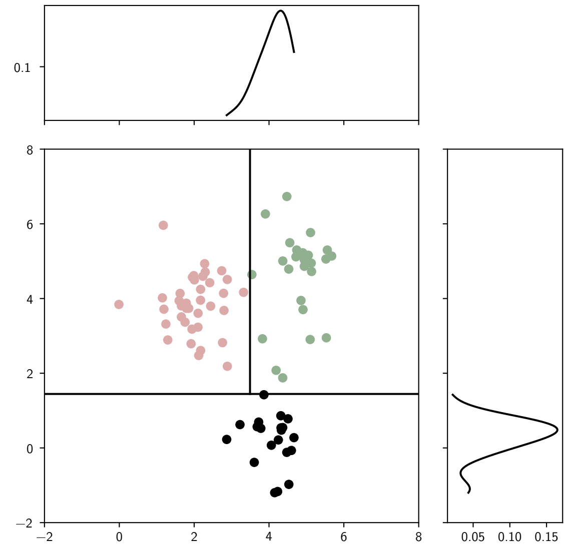
```
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above noiseLevel
Find q best cutting planes with score ≤ maxCutScore
if cutting planes were found then
    Divide data into subgrids C_1, ... C_n
    foreach subgrid do
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end
```



Optigrid

Input: Data
Parameters: *q*, *maxCutScore*, *noiseLevel*

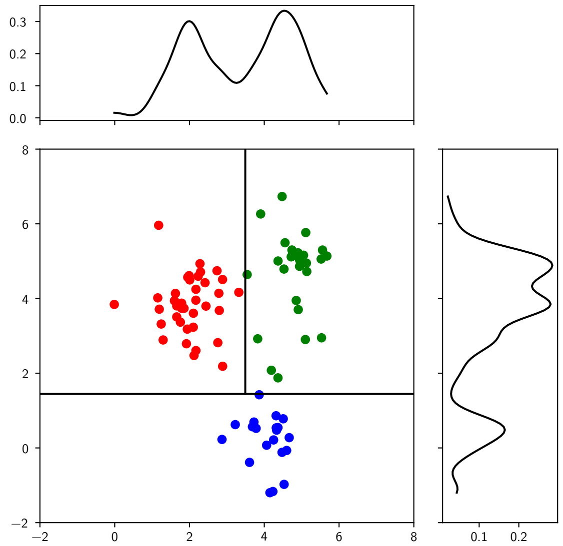
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above *noiseLevel*
Find *q* best cutting planes with score $\leq$ *maxCutScore*
if *cutting planes were found* **then**
    Divide data into subgrids $C_1, \ldots C_n$
    foreach *subgrid* **do**
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end



Optigrid

Input: Data
Parameters: q, *maxCutScore*, *noiseLevel*

```
Calculate Projections to lower dimensional space
Approx. Densities and find peaks above noiseLevel
Find q best cutting planes with score ≤ maxCutScore
if cutting planes were found then
    Divide data into subgrids C_1, ... C_n
    foreach subgrid do
        Optigrid(subgrid)
    end
else
    Mark subgrid as cluster
end
```



Optigrid

- Divides the input space into an irregular grid
- Calculations happen in low dimensions
- Finds clusters that can be enclosed in a rectangle
- Parameters can be chosen by analyzing the density distributions

Outlook

Can we find any clusters in the parameter settings of the Linac3 ion source?