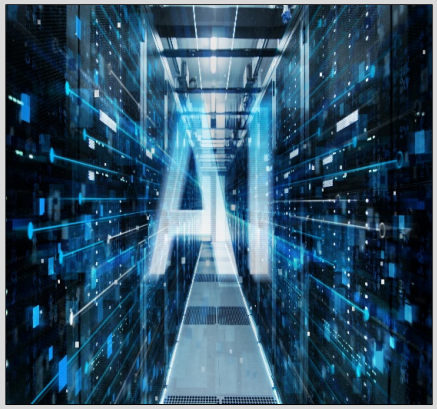


Artificial Intelligence Data Analysis (AIDA)

1st School for Heliophysicists

Prof. Dr. – Ing. Morris Riedel
Associated Professor
School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland
Research Group Leader, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany

@Morris Riedel @MorrisRiedel @MorrisRiedel

LECTURE 3

Unsupervised Learning – Computing

January 20, 2020
CINECA, Bologna, Italy

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

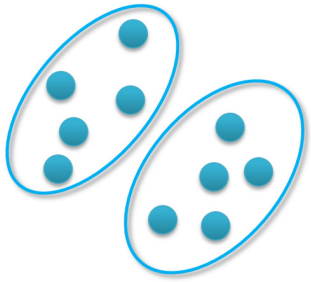
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

DEEP Projects

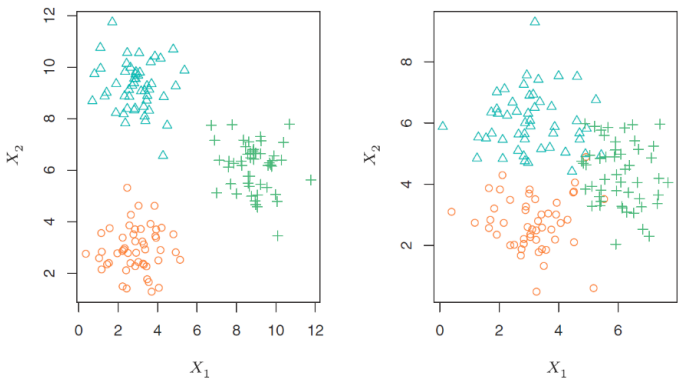
HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Review of Lecture 2 – Unsupervised Learning – Clustering

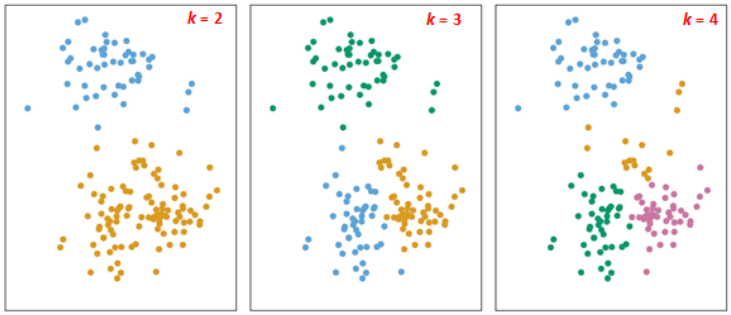
Clustering

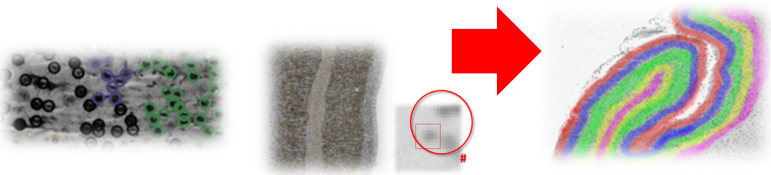


- No groups of data exist
- Create groups from data close to each other

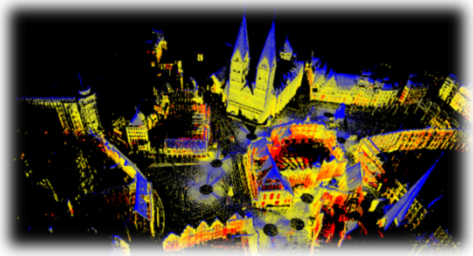


(number of cluster can be amiguities sometimes, hard for k-means)





(DBSCAN very effective, but two parameters)



[1] An Introduction to Statistical Learning

Outline of the School

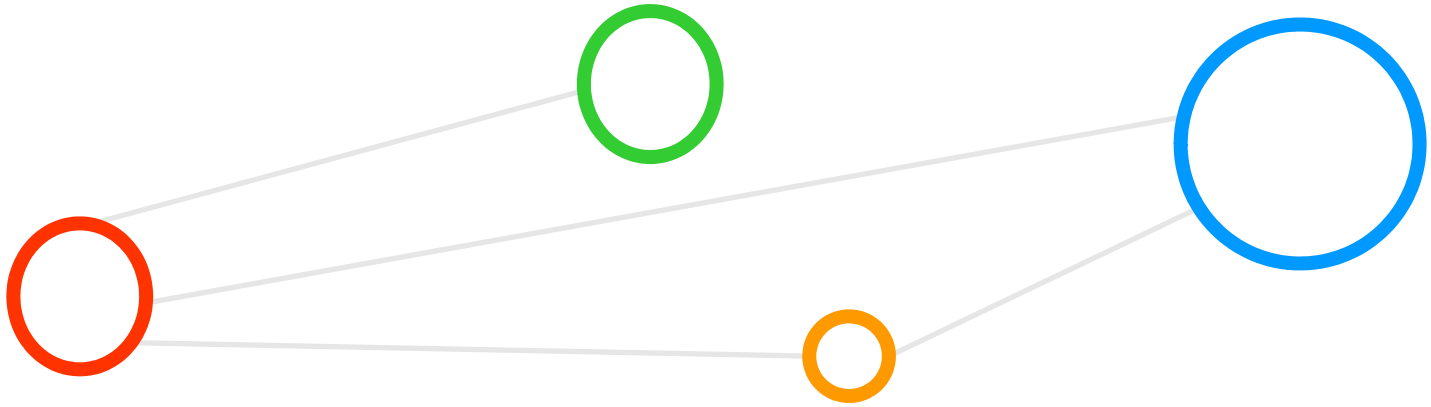
Time	**Day 1**	**Day 2**	**Day 3**
9 - 10	Welcome and intro to the school (Giovanni Lapenta, Jorge Amaya)	Space missions data acquisition (Hugo Breuillard)	Review of ML applied to heliophysics (Peter Wintoft)
10 - 11	Introduction and differences between AI, ML, NN and Big Data (Morris Riedel)	Data manipulation in python with pandas, xarray, and additional python tools (Geert Jan Bex)	Review of ML applied to heliophysics (Peter Wintoft)
	Coffee break	Coffee break	Coffee break
11:30 - 12:30	Unsupervised learning (Morris Riedel)	Feature engineering and data reduction (Geert Jan Bex)	Reinforcement learning (Morris Riedel)
	Lunch	Lunch	Lunch
14 - 15	Unsupervised learning (Morris Riedel)	Data reduction and visualization (Geert Jan Bex)	Physics informed ML (Romain Dupuis)
15 -16	Supervised learning (Morris Riedel)	CNN, DNN (Morris Riedel)	Explainable AI (Jorge Amaya)
	Coffee break	Coffee break	Coffee break
16:30 - 18:00	Supervised learning (Morris Riedel)	CNN, DNN (Morris Riedel)	Performance and tuning of ML (Morris Riedel)

Outline

- Unsupervised Learning – Computing in Scikit-Learn
  - Unsupervised Learning – Revisited
  - LIDAR and IRIS Application Examples
  - K-Means Algorithm Example & Options
  - DBSCAN Algorithm Example & Options
  - Comparions between Clustering Algorithms

- Unsupervised Learning – Computing in Parallel
  - HPDBSCAN Revisited & Smart Domain Decomposition
  - Big Data and Parallelization Benefits
  - Scaling with Number of Processors
  - Twitter Dataset Example
  - Other Unsupervised Approaches like Self Organizing Maps

Unsupervised Learning – Computing in Scikit-Learn



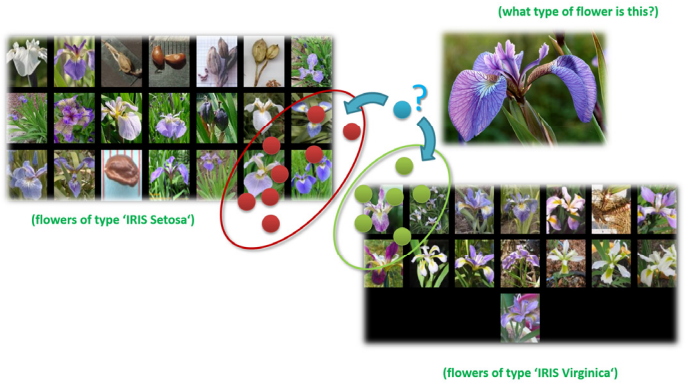
Learning Approaches – What means Learning from data?

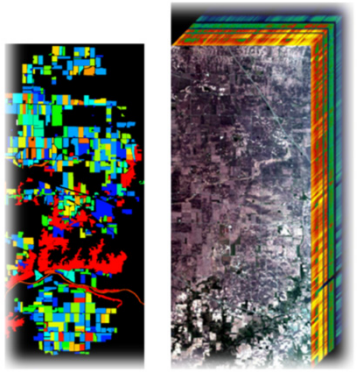
- **The basic meaning of learning is 'to use a set of observations to uncover an underlying process'**
- **The three different learning approaches are supervised, unsupervised, and reinforcement learning**

[14] Image sources: Species Iris Group of North America Database, www.signa.org

Supervised Learning

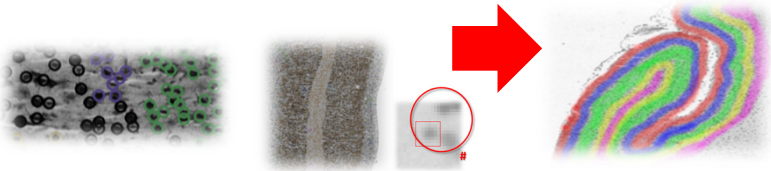
- Majority of methods follow this approach in this course
- Example: credit card approval based on previous customer applications



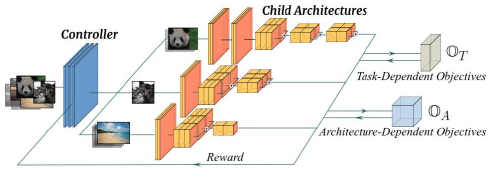


Unsupervised Learning

- Often applied before other learning → higher level data representation
- Example: Coin recognition in vending machine based on weight and size



[15] A.C. Cheng et al., 'InstaNAS: Instance-aware Neural Architecture Search', 2018



Reinforcement Learning

- Typical 'human way' of learning
- Example: Toddler tries to touch a hot cup of tea (again and again)

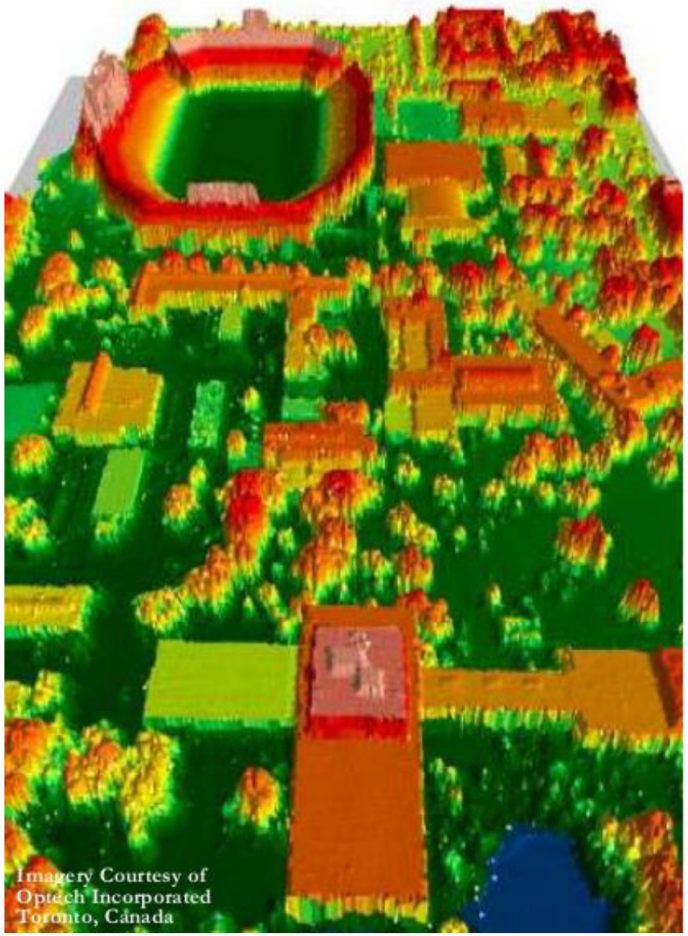
➢ **Day 1 offers details about unsupervised & supervised learning with examples & Day 3 offers an introduction to reinforcement learning**

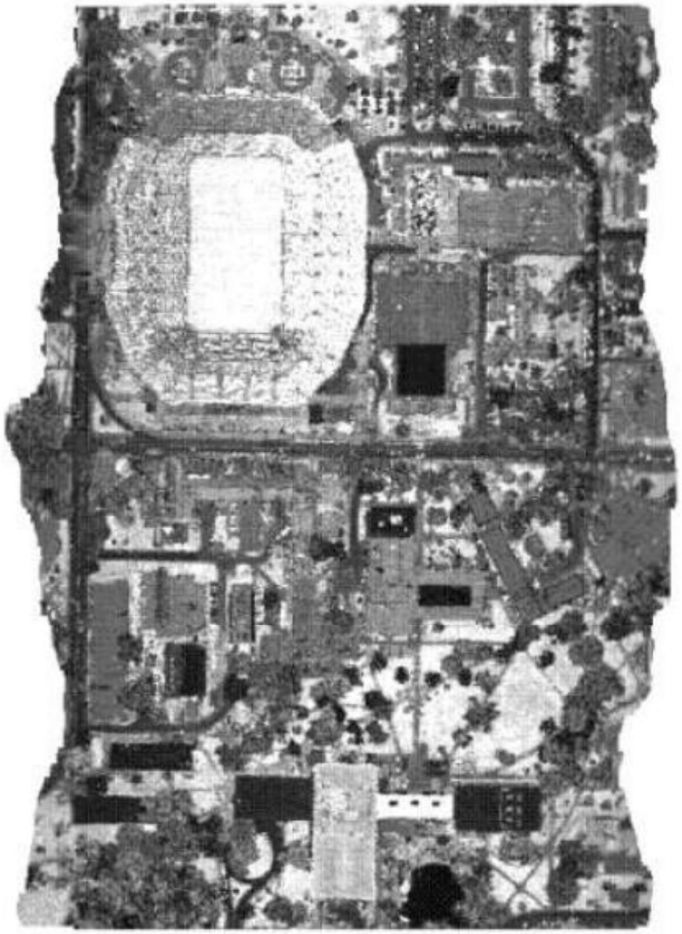
LIDAR Application Example

- **LIght Detection And Ranging**
  - Active Sensing System
  - Day or Night operation.
  - Ranging of the reflecting object based on time difference between emission and reflection.

- **What's NOT LiDAR?**
  - NOT Light/Laser Assisted RADAR
    - RADAR uses electro-magnetic (EM) energy in the radio frequency range;
    - LIDAR does not.
  - NOT all-weather
    - The target MUST be visible. Some haze is manageable, but fog is not
  - NOT able to `see through' trees
    - LIDAR sees around trees, not through them. Fully closed canopies (rain forests) cannot be penetrated
  - NOT a Substitute for Photography
    - For MOST users, LIDAR intensity images are NOT viable replacements for conventional or digital imagery

LIDAR Clustering Example

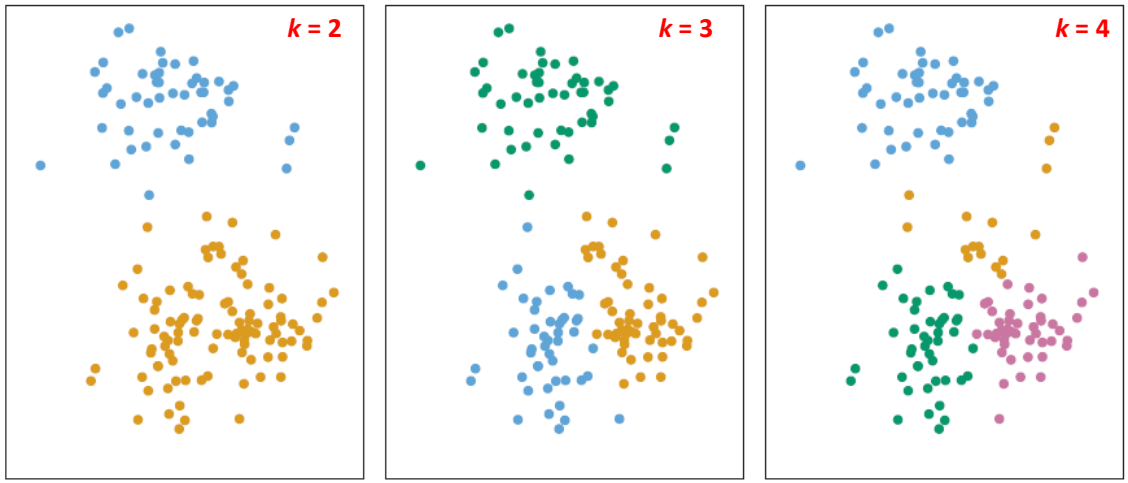




[2] G. Camps-Valls

Clustering Methods – K-Means Approach

- Approach Overview
  - Partitions a data set into *K* distinct (i.e. non-overlapping) clusters
  - Requires the definition of the desired number of clusters *K* in advance
  - Assigns each observation / data element to exactly one of the *K* clusters
  - Example: 150 observations; 2 dimensions; 3 different values of *K*



[1] An Introduction to Statistical Learning

Scikit-Learn K-Means Algorithm Example & Options

Examples

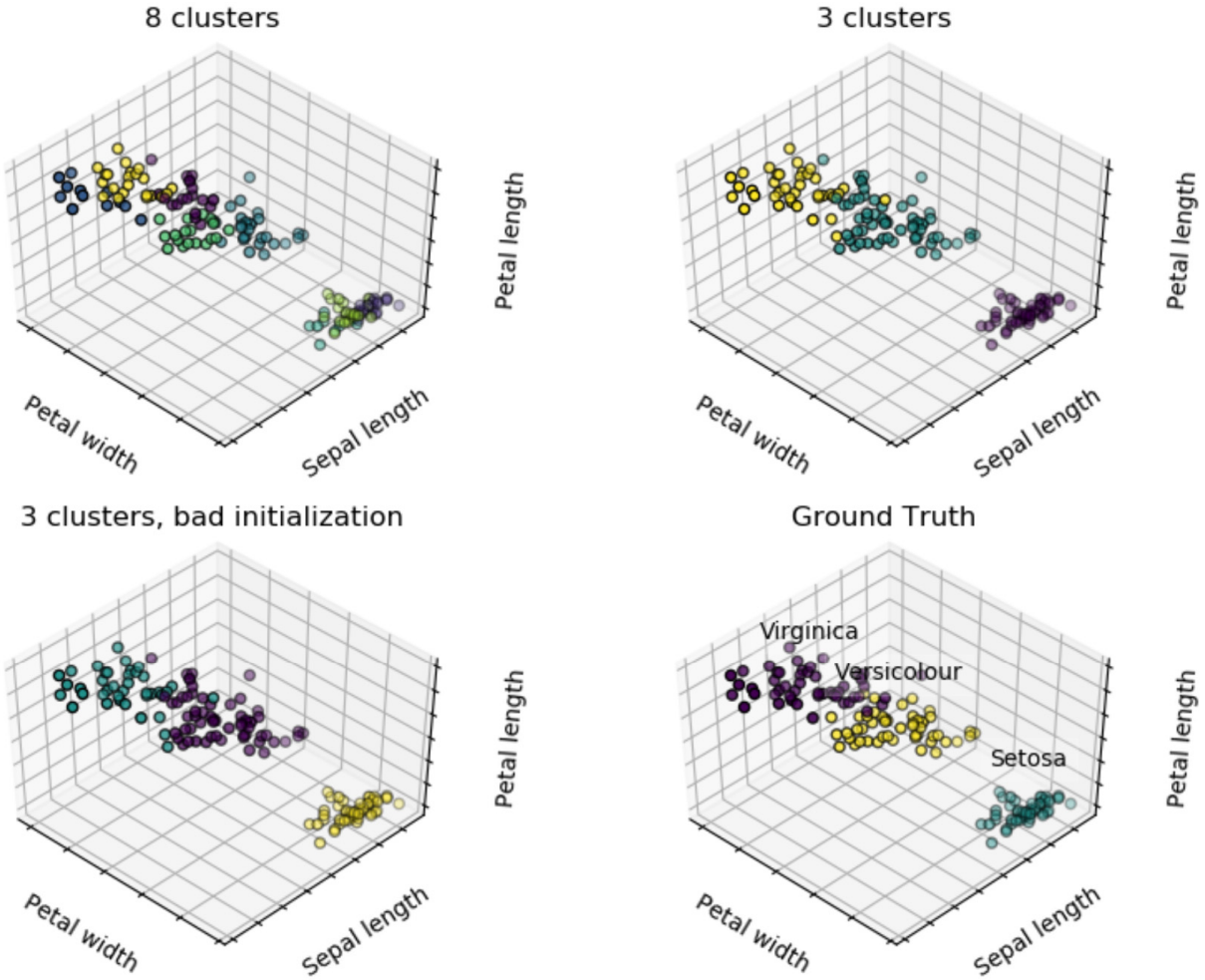
```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [10, 2], [10, 4], [10, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([1, 1, 1, 0, 0, 0], dtype=int32)
>>> kmeans.predict([[0, 0], [12, 3]])
array([1, 0], dtype=int32)
>>> kmeans.cluster_centers_
array([[10.,  2.],
       [ 1.,  2.]])
```

Methods

`fit`(self, X[, y, sample_weight])	Compute k-means clustering.
`fit_predict`(self, X[, y, sample_weight])	Compute cluster centers and predict cluster index for each sample.
`fit_transform`(self, X[, y, sample_weight])	Compute clustering and transform X to cluster-distance space.
`get_params`(self[, deep])	Get parameters for this estimator.
`predict`(self, X[, sample_weight])	Predict the closest cluster each sample in X belongs to.
`score`(self, X[, y, sample_weight])	Opposite of the value of X on the K-means objective.
`set_params`(self, **params)	Set the parameters of this estimator.
`transform`(self, X)	Transform X to a cluster-distance space.

[6] Scitkit-Learn

IRIS Dataset – Clustering Example



[6] Scitkit-Learn

Scikit-Learn – Clustering Example IRIS Flowers Part 1

```
print(__doc__)


# Code source: Gaël Varoquaux
# Modified for documentation by Jaques Grobler
# License: BSD 3 clause

import numpy as np
import matplotlib.pyplot as plt
# Though the following import is not directly being used, it is required
# for 3D projection to work
from mpl_toolkits.mplot3d import Axes3D

from sklearn.cluster import KMeans
from sklearn import datasets

np.random.seed(5)

iris = datasets.load_iris()
X = iris.data
y = iris.target

estimators = [('k_means_iris_8', KMeans(n_clusters=8)),
              ('k_means_iris_3', KMeans(n_clusters=3)),
              ('k_means_iris_bad_init', KMeans(n_clusters=3, n_init=1,
                                               init='random'))]

fignum = 1
titles = ['8 clusters', '3 clusters', '3 clusters, bad initialization']
for name, est in estimators:
    fig = plt.figure(fignum, figsize=(4, 3))
    ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
    est.fit(X)
    labels = est.labels_

    ax.scatter(X[:, 3], X[:, 0], X[:, 2],
               c=labels.astype(np.float), edgecolor='k')

    ax.w_xaxis.set_ticklabels([])
    ax.w_yaxis.set_ticklabels([])
    ax.w_zaxis.set_ticklabels([])
    ax.set_xlabel('Petal width')
    ax.set_ylabel('Sepal length')
    ax.set_zlabel('Petal length')
    ax.set_title(titles[fignum - 1])
    ax.dist = 12
    fignum = fignum + 1
```

[6] Scitkit-Learn

Scikit-Learn – Clustering Example IRIS Flowers Part 2 & Runtime

```
# Plot the ground truth
fig = plt.figure(fignum, figsize=(4, 3))
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)

for name, label in [('Setosa', 0),
                    ('Versicolour', 1),
                    ('Virginica', 2)]:
    ax.text3D(X[y == label, 3].mean(),
              X[y == label, 0].mean(),
              X[y == label, 2].mean() + 2, name,
              horizontalalignment='center',
              bbox=dict(alpha=.2, edgecolor='w', facecolor='w'))
# Reorder the labels to have colors matching the cluster results
y = np.choose(y, [1, 2, 0]).astype(np.float)
ax.scatter(X[:, 3], X[:, 0], X[:, 2], c=y, edgecolor='k')

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('Petal width')
ax.set_ylabel('Sepal length')
ax.set_zlabel('Petal length')
ax.set_title('Ground Truth')
ax.dist = 12

fig.show()
```

Total running time of the script: (0 minutes 0.565 seconds)

[6] Scitkit-Learn

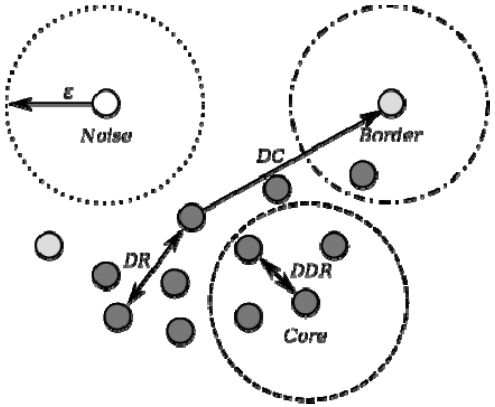
DBSCAN Algorithm

- DBSCAN Algorithm
  - Introduced 1996 and most cited clustering algorithm
  - Groups number of similar points into clusters of data
  - Similarity is defined by a distance measure (e.g. *euclidean distance*)

- Distinct Algorithm Features
  - Clusters a variable number of clusters
  - Forms arbitrarily shaped clusters (except ‘bow ties‘)
  - Identifies inherently also outliers/noise

- Understanding Parameters
  - Looks for a similar points within a given search radius → Parameter *epsilon*
  - A cluster consist of a given minimum number of points → Parameter *minPoints*

[6] Ester et al.



(MinPoints = 4)

(DR = Density Reachable)

(DDR = Directly Density Reachable)

(DC = Density Connected)

Exercise: DBSCAN Algorithm – Non-Trivial Example

- Compare K-Means vs. DBSCAN – How would K-Means work?

Unclustered Data

Clustered Data

- **DBSCAN forms arbitrarily shaped clusters (except 'bow ties') where other clustering algorithms fail**

Scikit-Learn DBSCAN Algorithm Example & Options

Examples

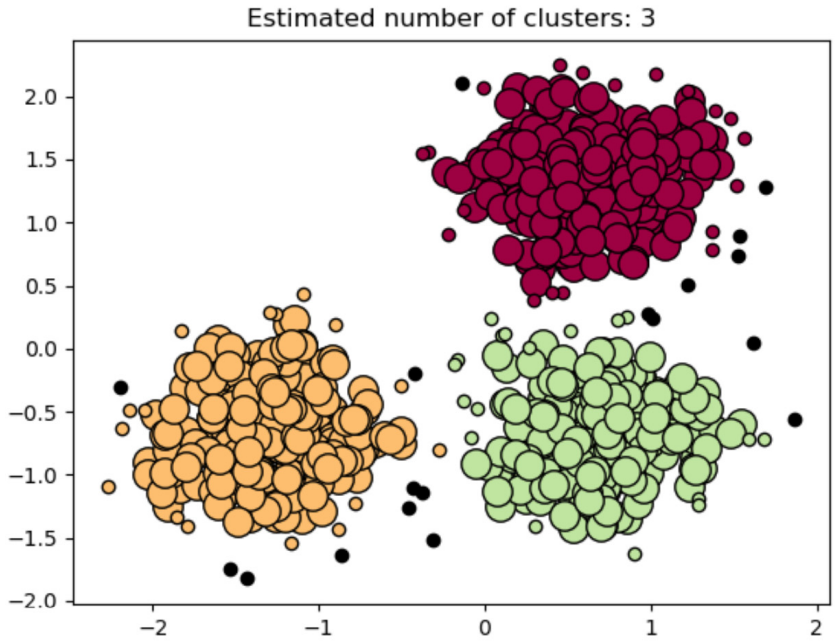
```
>>> from sklearn.cluster import DBSCAN
>>> import numpy as np
>>> X = np.array([[1, 2], [2, 2], [2, 3],
...               [8, 7], [8, 8], [25, 80]])
>>> clustering = DBSCAN(eps=3, min_samples=2).fit(X)
>>> clustering.labels_
array([ 0,  0,  0,  1,  1, -1])
>>> clustering
DBSCAN(eps=3, min_samples=2)
```

Methods

`fit`(self, X[, y, sample_weight])	Perform DBSCAN clustering from features, or distance matrix.
`fit_predict`(self, X[, y, sample_weight])	Perform DBSCAN clustering from features or distance matrix, and return cluster labels.
`get_params`(self[, deep])	Get parameters for this estimator.
`set_params`(self, **params)	Set the parameters of this estimator.

[6] Scitkit-Learn

DBSCAN Clustering Example – Outlier Detection



Out:

```
Estimated number of clusters: 3
Estimated number of noise points: 18
Homogeneity: 0.953
Completeness: 0.883
V-measure: 0.917
Adjusted Rand Index: 0.952
Adjusted Mutual Information: 0.916
Silhouette Coefficient: 0.626
```

[6] Scitkit-Learn

DBSCAN Clustering Example – Outlier Detection – Script 1

```
print(__doc__)

import numpy as np

from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler


# #############################################################################
# Generate sample data
centers = [[1, 1], [-1, -1], [1, -1]]
X, labels_true = make_blobs(n_samples=750, centers=centers, cluster_std=0.4,
                            random_state=0)

X = StandardScaler().fit_transform(X)

# #############################################################################
# Compute DBSCAN
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels_true, labels))
print("Completeness: %0.3f" % metrics.completeness_score(labels_true, labels))
print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels))
print("Adjusted Rand Index: %0.3f"
      % metrics.adjusted_rand_score(labels_true, labels))
print("Adjusted Mutual Information: %0.3f"
      % metrics.adjusted_mutual_info_score(labels_true, labels))
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(X, labels))
```

[6] Scitkit-Learn

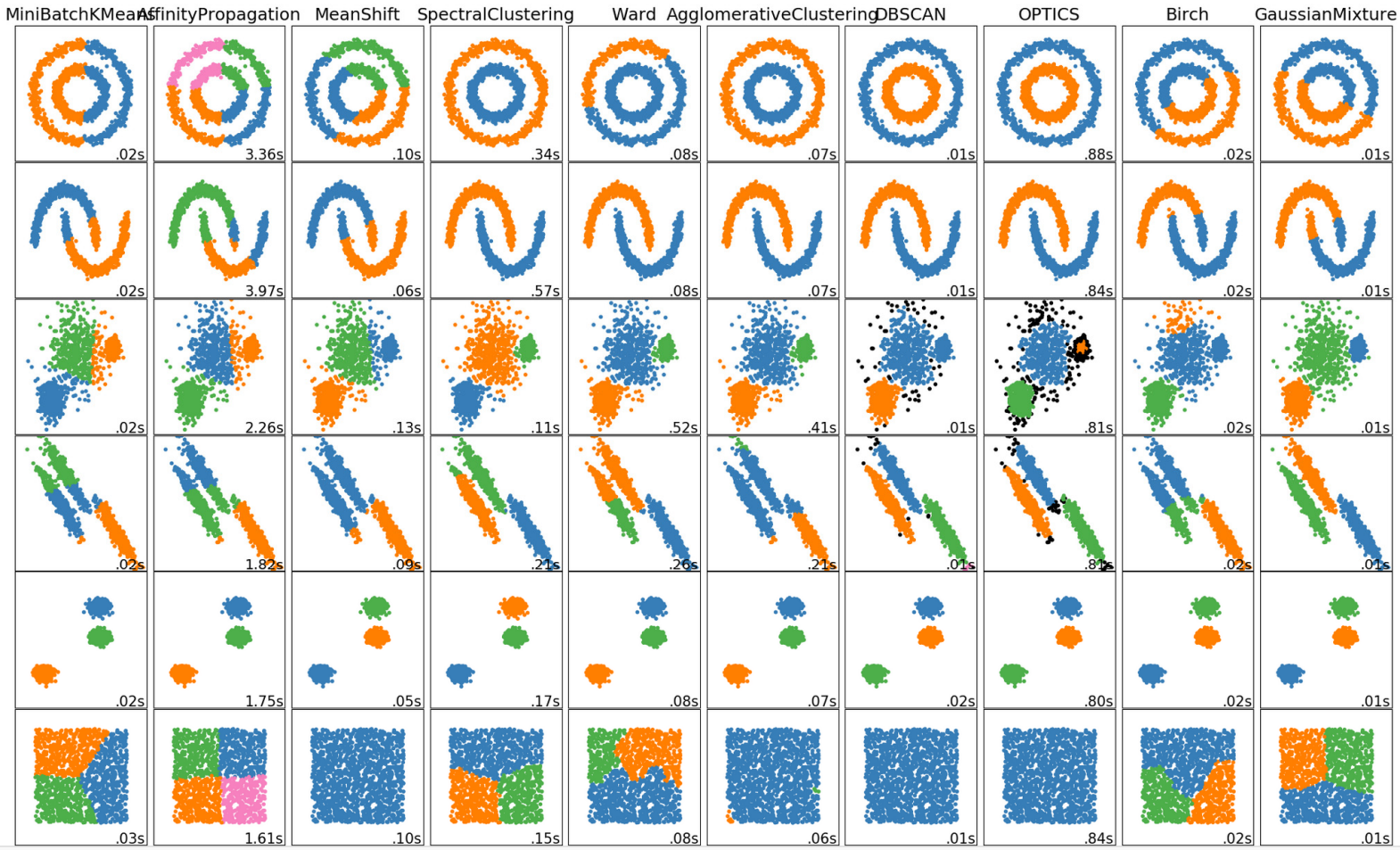
DBSCAN Clustering Example – Outlier Detection – Script 2 & Runtime

```
# ##############################################################################
# Plot result
import matplotlib.pyplot as plt

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=14)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)

plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```

[6] Scitkit-Learn

Total running time of the script: (0 minutes 0.416 seconds)

Scikit-Learn Clustering Mechansims Compared



[6] Scitkit-Learn

Scikit-Learn Clustering Mechansims Compared – Script 1

```
print(__doc__)

import time
import warnings

import numpy as np
import matplotlib.pyplot as plt

from sklearn import cluster, datasets, mixture
from sklearn.neighbors import kneighbors_graph
from sklearn.preprocessing import StandardScaler
from itertools import cycle, islice

np.random.seed(0)

# ============
# Generate datasets. We choose the size big enough to see the scalability
# of the algorithms, but not too big to avoid too long running times
# ============
n_samples = 1500
noisy_circles = datasets.make_circles(n_samples=n_samples, factor=.5,
                                      noise=.05)
noisy_moons = datasets.make_moons(n_samples=n_samples, noise=.05)
blobs = datasets.make_blobs(n_samples=n_samples, random_state=8)
no_structure = np.random.rand(n_samples, 2), None

# Anisotropicly distributed data
random_state = 170
X, y = datasets.make_blobs(n_samples=n_samples, random_state=random_state)
transformation = [[0.6, -0.6], [-0.4, 0.8]]
X_aniso = np.dot(X, transformation)
aniso = (X_aniso, y)

# blobs with varied variances
varied = datasets.make_blobs(n_samples=n_samples,
                             cluster_std=[1.0, 2.5, 0.5],
                             random_state=random_state)
```

[6] Scitkit-Learn

Scikit-Learn Clustering Mechansims Compared – Script 2

```
# ============
# Set up cluster parameters
# ============
plt.figure(figsize=(9 * 2 + 3, 12.5))
plt.subplots_adjust(left=.02, right=.98, bottom=.001, top=.96, wspace=.05,
                    hspace=.01)

plot_num = 1

default_base = {'quantile': .3,
                'eps': .3,
                'damping': .9,
                'preference': -200,
                'n_neighbors': 10,
                'n_clusters': 3,
                'min_samples': 20,
                'xi': 0.05,
                'min_cluster_size': 0.1}

datasets = [
    (noisy_circles, {'damping': .77, 'preference': -240,
                     'quantile': .2, 'n_clusters': 2,
                     'min_samples': 20, 'xi': 0.25}),
    (noisy_moons, {'damping': .75, 'preference': -220, 'n_clusters': 2}),
    (varied, {'eps': .18, 'n_neighbors': 2,
              'min_samples': 5, 'xi': 0.035, 'min_cluster_size': .2}),
    (aniso, {'eps': .15, 'n_neighbors': 2,
             'min_samples': 20, 'xi': 0.1, 'min_cluster_size': .2}),
    (blobs, {}),
    (no_structure, {})]
```

[6] Scitkit-Learn

Scikit-Learn Clustering Mechansims Compared – Script 3

```
for i_dataset, (dataset, algo_params) in enumerate(datasets):
    # update parameters with dataset-specific values
    params = default_base.copy()
    params.update(algo_params)

    X, y = dataset

    # normalize dataset for easier parameter selection
    X = StandardScaler().fit_transform(X)

    # estimate bandwidth for mean shift
    bandwidth = cluster.estimate_bandwidth(X, quantile=params['quantile'])

    # connectivity matrix for structured Ward
    connectivity = kneighbors_graph(
        X, n_neighbors=params['n_neighbors'], include_self=False)
    # make connectivity symmetric
    connectivity = 0.5 * (connectivity + connectivity.T)

    # ============
    # Create cluster objects
    # ============
    ms = cluster.MeanShift(bandwidth=bandwidth, bin_seeding=True)
    two_means = cluster.MiniBatchKMeans(n_clusters=params['n_clusters'])
    ward = cluster.AgglomerativeClustering(
        n_clusters=params['n_clusters'], linkage='ward',
        connectivity=connectivity)
    spectral = cluster.SpectralClustering(
        n_clusters=params['n_clusters'], eigen_solver='arpack',
        affinity="nearest_neighbors")
    dbscan = cluster.DBSCAN(eps=params['eps'])
    optics = cluster.OPTICS(min_samples=params['min_samples'],
                            xi=params['xi'],
                            min_cluster_size=params['min_cluster_size'])
    affinity_propagation = cluster.AffinityPropagation(
        damping=params['damping'], preference=params['preference'])
    average_linkage = cluster.AgglomerativeClustering(
        linkage="average", affinity="cityblock",
        n_clusters=params['n_clusters'], connectivity=connectivity)
    birch = cluster.Birch(n_clusters=params['n_clusters'])
    gmm = mixture.GaussianMixture(
        n_components=params['n_clusters'], covariance_type='full')
```

[6] Scitkit-Learn

Scikit-Learn Clustering Mechansims Compared – Script 4

```
clustering_algorithms = (
    ('MiniBatchKMeans', two_means),
    ('AffinityPropagation', affinity_propagation),
    ('MeanShift', ms),
    ('SpectralClustering', spectral),
    ('Ward', ward),
    ('AgglomerativeClustering', average_linkage),
    ('DBSCAN', dbscan),
    ('OPTICS', optics),
    ('Birch', birch),
    ('GaussianMixture', gmm)
)
```

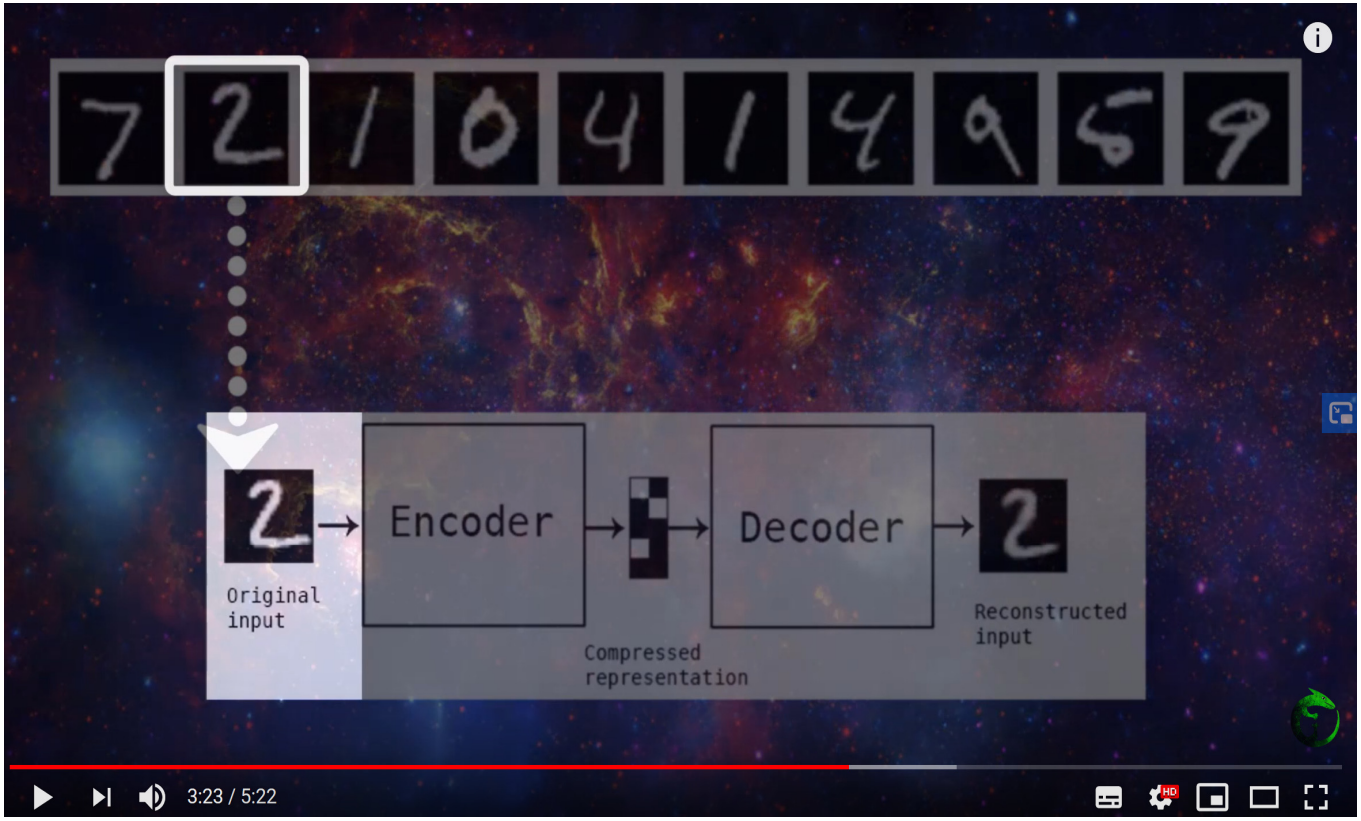
[6] Scitkit-Learn

Scikit-Learn Clustering Mechansims Compared – Script 5

```
    for name, algorithm in clustering_algorithms:
        t0 = time.time()

        # catch warnings related to kneighbors_graph
        with warnings.catch_warnings():
            warnings.filterwarnings(
                "ignore",
                message="the number of connected components of the " +
                "connectivity matrix is [0-9]{1,2}" +
                " > 1. Completing it to avoid stopping the tree early.",
                category=UserWarning)
            warnings.filterwarnings(
                "ignore",
                message="Graph is not fully connected, spectral embedding" +
                " may not work as expected.",
                category=UserWarning)
            algorithm.fit(X)

        t1 = time.time()
        if hasattr(algorithm, 'labels_'):
            y_pred = algorithm.labels_.astype(np.int)
        else:
            y_pred = algorithm.predict(X)

        plt.subplot(len(datasets), len(clustering_algorithms), plot_num)
        if i_dataset == 0:
            plt.title(name, size=18)

        colors = np.array(list(islice(cycle(['#377eb8', '#ff7f00', '#4daf4a',
                                             '#f781bf', '#a65628', '#984ea3',
                                             '#999999', '#e41a1c', '#dede00']),
                                      int(max(y_pred) + 1))))
        # add black color for outliers (if any)
        colors = np.append(colors, ["#000000"])
        plt.scatter(X[:, 0], X[:, 1], s=10, color=colors[y_pred])

        plt.xlim(-2.5, 2.5)
        plt.ylim(-2.5, 2.5)
        plt.xticks(())
        plt.yticks(())
        plt.text(.99, .01, ('%.2fs' % (t1 - t0)).lstrip('0'),
                 transform=plt.gca().transAxes, size=15,
                 horizontalalignment='right')
        plot_num += 1

plt.show()
```

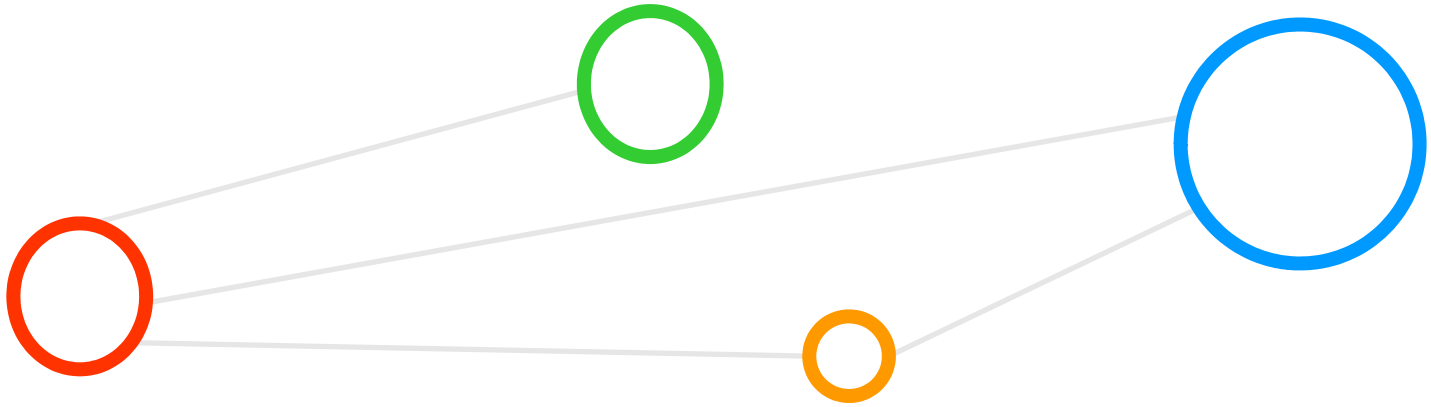
[6] Scitkit-Learn

[Video] Summary Unsupervised Learning



[8] YouTube Video, Unsupervised Learning Explained

Unsupervised Learning – Computing in Parallel



Review of Parallel DBSCAN Implementations

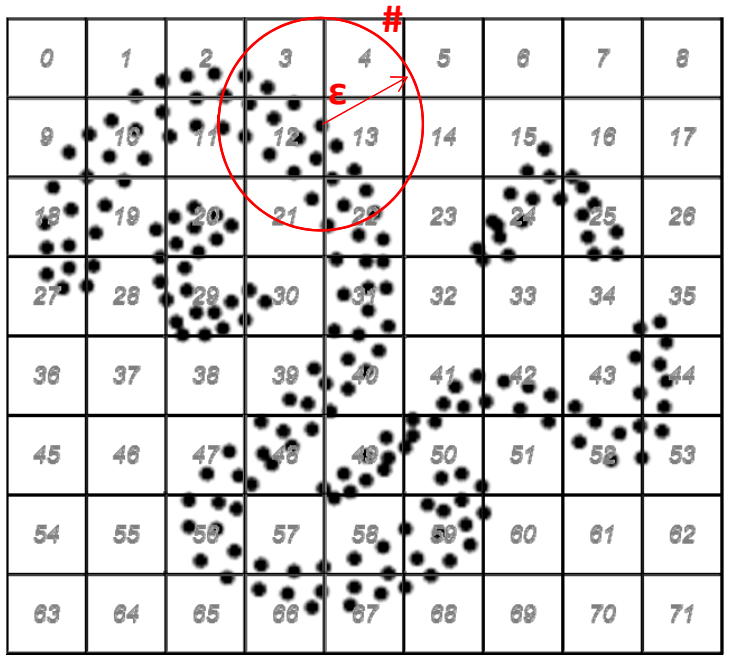
Technology	Platform Approach	Analysis
HPDBSCAN (authors implementation)	C; MPI; OpenMP	Parallel, hybrid, DBSCAN
Apache Mahout	Java; Hadoop	K-means variants, spectral, no DBSCAN
Apache Spark/MLlib	Java; Spark	Only k-means clustering, No DBSCAN
scikit-learn	Python	No parallelization strategy for DBSCAN
Northwestern University PDSDBSCAN-D	C++; MPI; OpenMP	Parallel DBSCAN

[4] M. Goetz, M. Riedel et al., **'On Parallel and Scalable Classification and Clustering Techniques for Earth Science Datasets'** ***, 6th Workshop on Data Mining in Earth System Science, International Conference of Computational Science (ICCS)***

HDBSCAN Algorithm Details

- Parallelization Strategy
  - Smart ‘Big Data‘ Preprocessing into Spatial Cells (‘indexed‘)
  - OpenMP and HDF5 parallel I/O
  - MPI (+ optional OpenMP hybrid)

- Preprocessing Step
  - Spatial indexing and redistribution according to the point localities
  - Data density based chunking of computations

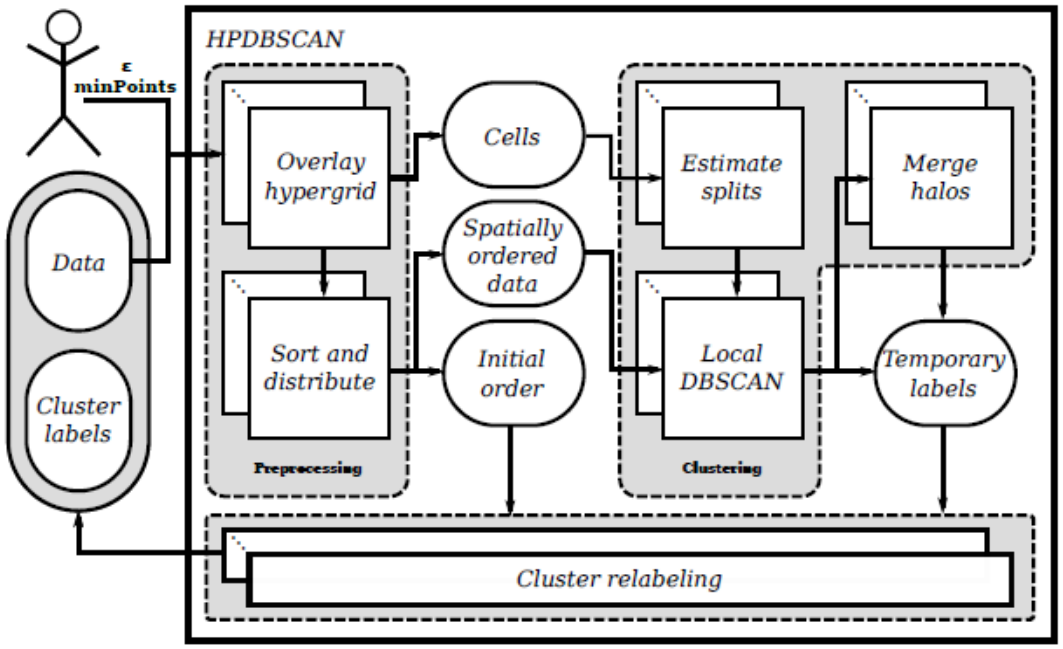


[3] M.Goetz, M. Riedel et al., ‘HPDBSCAN – Highly Parallel DBSCAN’, MLHPC Workshop at Supercomputing 2015

- Computational Optimizations
  - Caching of point neighborhood searches
  - Cluster merging based on comparisons instead of zone reclustering

HPDBSCAN – Smart Domain Decomposition Example

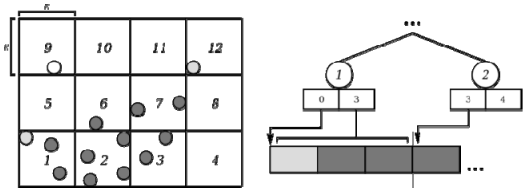
- Parallelization Strategy
  - Chunk data space equally
  - Overlay with hypergrid
  - Apply cost heuristic
  - Redistribute points (data locality)
  - Execute DBSCAN locally
  - Merge clusters at chunk edges
  - Restore initial order
- Data organization
  - Use of HDF5
  - Cluster Id / noise ID stored in HDF5 file

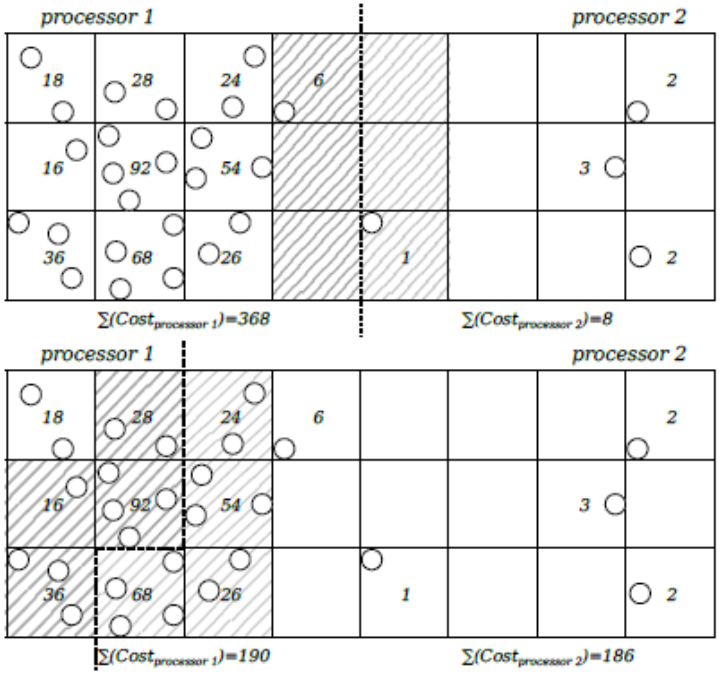


[3] M.Goetz, M. Riedel et al., 'HPDBSCAN – Highly Parallel DBSCAN', MLHPC Workshop at Supercomputing 2015

HPDBSCAN – Domain Decomposition

- Parallelization Strategy
  - Chunk data space equally
  - Overlay with hypergrid
  - Apply cost heuristic
  - Redistribute points (data locality)
  - Execute DBSCAN locally
  - Merge clusters at chunk edges
  - Restore initial order
- Data organization
  - Use of HDF5
  - Cluster Id / noise ID stored in HDF5 file

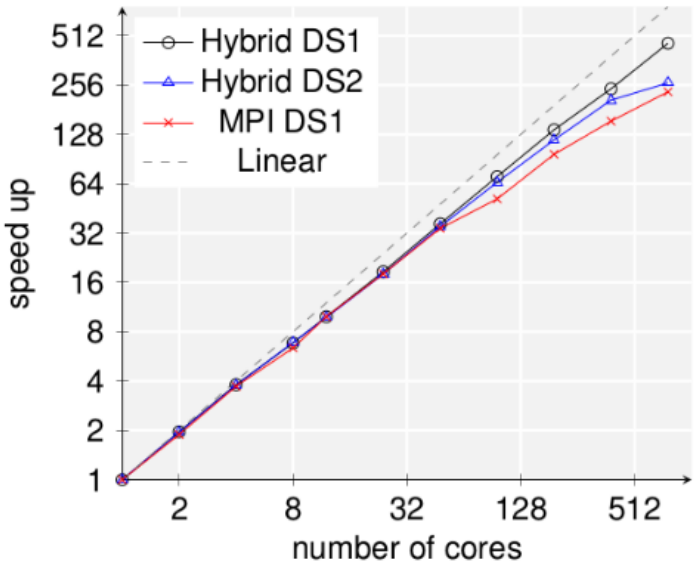




[3] M.Goetz, M. Riedel et al., 'HPDBSCAN – Highly Parallel DBSCAN', MLHPC Workshop at Supercomputing 2015

HPDBSCAN – Scaling

- Parallelization Strategy
  - Chunk data space equally
  - Overlay with hypergrid
  - Apply cost heuristic
  - Redistribute points (data locality)
  - Execute DBSCAN locally
  - Merge clusters at chunk edges
  - Restore initial order
- Data organization
  - Use of HDF5
  - Cluster Id / noise ID stored in HDF5 file

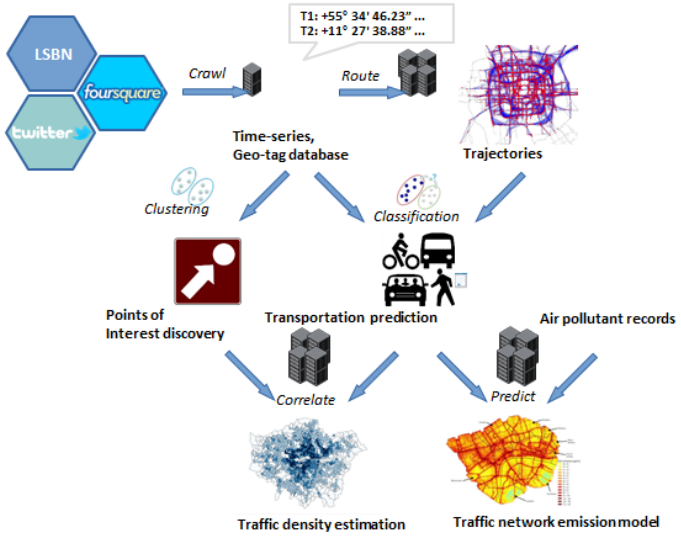


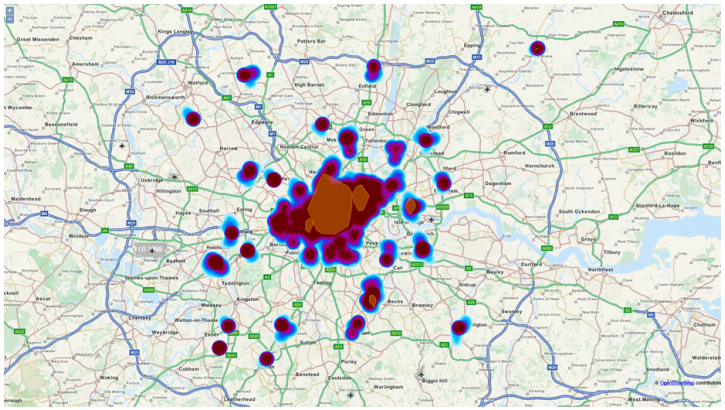
(DS1 = Bremen; DS2 = Twitter)

[3] M.Goetz, M. Riedel et al., 'HPDBSCAN – Highly Parallel DBSCAN', MLHPC Workshop at Supercomputing 2015

Twitter Dataset & Locations – Revisited

- Twitter streaming API data
  - Containts 1% of all geo-tagged of the UK in June 2014 (e.g. London)





- **The Twitter Dataset is encoded in the HDF5 format (binary)**
- **You need your own copy of the file in your home directory to cluster!**

```
[train001@jrl04 twitter]$ pwd
/homea/hpclab/train001/data/twitter
[train001@jrl04 twitter]$ ls -al
total 317312
drwxr-xr-x 2 train001 hpclab       512 Jan 14 23:00 .
drwxr-xr-x 8 train001 hpclab       512 Jan 14 22:06 ..
-rw-r--r-- 1 train001 hpclab 265636608 Jan 13  2017 twitter.h5
-rw-r--r-- 1 train001 hpclab  59272032 Jan 13  2017 twitterSmall.h5
```

[5] Twitter Dataset

B2SHARE
Store and Share Research Data

JURECA HPC System – HPDBSCAN Job Script

```
#!/bin/bash
#SBATCH --job-name=HPDBSCAN
#SBATCH -o HPDBSCAN-%j.out
#SBATCH -e HPDBSCAN-%j.err
#SBATCH --nodes=4
#SBATCH --ntasks=4
#SBATCH --ntasks-per-node=4
#SBATCH --time=01:00:00
#SBATCH --cpus-per-task=4
#SBATCH --reservation=ml-hpc-1

export OMP_NUM_THREADS=4

# location executable
HPDBSCAN=/homea/hpclab/train001/tools/hpdbscan/dbscan

# your own copy of bremen small
TWITTERSMALLDATA=/homea/hpclab/train001/twitterSmall.h5

# your own copy of bremen big
TWITTERBIGDATA=/homea/hpclab/train001/twitter.h5

srun $HPDBSCAN -m 40 -e 0.0001 -t 12 $TWITTERSMALLDATA
```

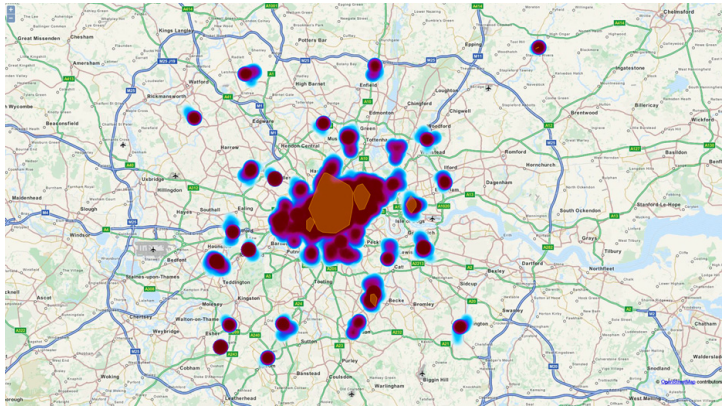
- **Job submit using command: sbatch <jobscript>**
- **Remember your <jobid> that is returned from the sbatch command**
- **Show status of the job then with: squeue –u <your-user-id>**

(parameters of DBSCAN and file to be clustered)

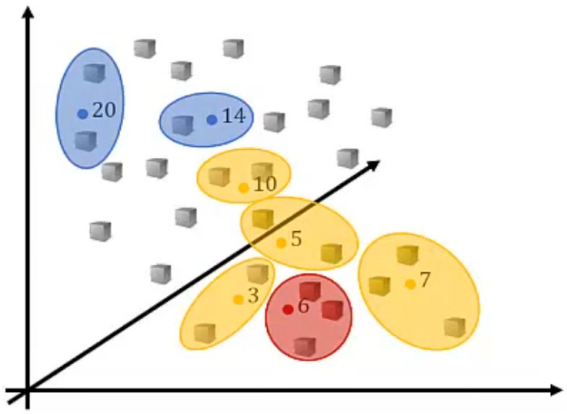
JURECA HPC System – HPDBSCAN Check Outcome

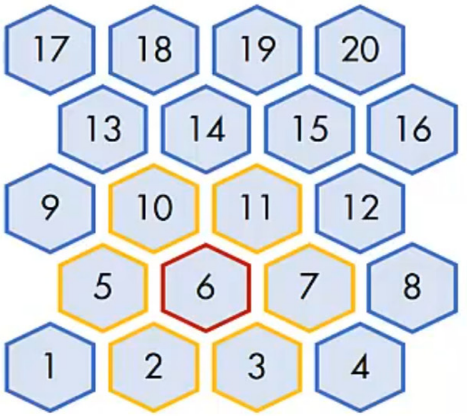
```
[train001@jrl04 hpdbscan]$ more HPDBSCAN-4632910.out
Calculating Cell Space...
        Computing Dimensions... [OK] in 0.002393
        Computing Cells...      [OK] in 0.498816
        Sorting Points...       [OK] in 0.891462
        Distributing Points...  [OK] in 2.576206
DBSCAN...
        Local Scan...           [OK] in 1.375779
        Merging Neighbors...    [OK] in 0.000586
        Adjust Labels ...       [OK] in 0.013686
        Rec. Init. Order ...    [OK] in 0.640681
        Writing File ...        [OK] in 0.006232
Result...
        8976    Clusters
        906807  Cluster Points
        2797544 Noise Points
        757369  Core Points
Took: 6.189666s
```

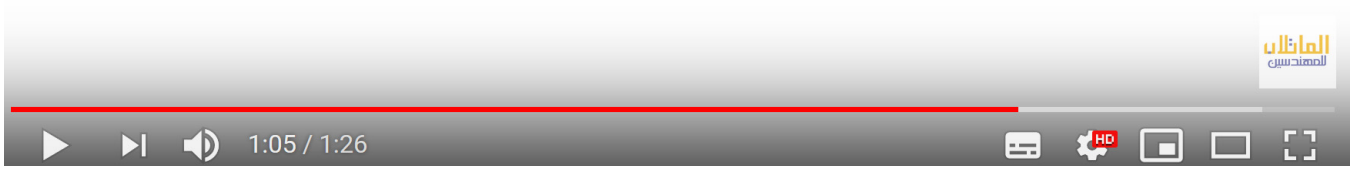
- **The outcome of the clustering processis written directly into the HDF5 file using cluster IDs and noise IDs**



[Video] Self Organizing Maps

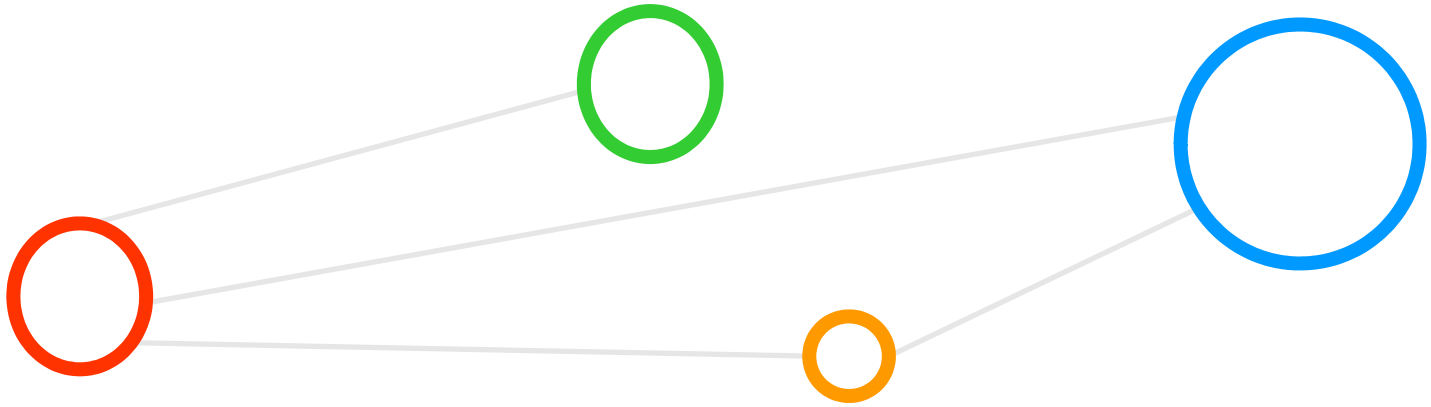






[7] YouTube Video, Self Organizing Maps

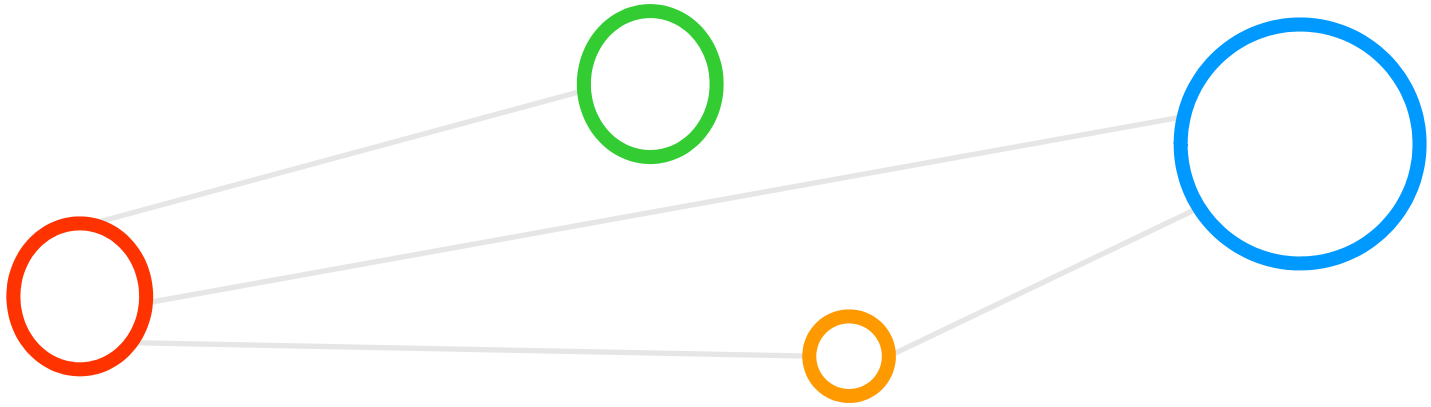
Lecture Bibliography



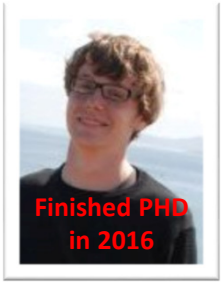
Lecture Bibliography

- [1] An Introduction to Statistical Learning with Applications in R, Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [2] Camps Vall, Online
- [3] M.Goetz, M. Riedel et al.,'HPDBSCAN – Highly Parallel DBSCAN', Proceedings of MLHPC Workshop at Supercomputing 2015, Online: https://www.researchgate.net/publication/301463871_HPDBSCAN_highly_parallel_DBSCAN
- [4] M. Goetz, M. Riedel et al.,' On Parallel and Scalable Classification and Clustering Techniques for Earth Science Datasets' 6th Workshop on Data Mining in Earth System Science, Proceedings of the International Conference of Computational Science (ICCS), Reykjavik, Online: http://www.proceedings.com/26605.html
- [5] B2SHARE, 'HPDBSCAN Benchmark test files', contains Twitter dataset, Online: http://hdl.handle.net/11304/6eacaa76-c275-11e4-ac7e-860aa0063d1f
- [6] Scikit-Learn, Online: https://scikit-learn.org
- [7] YouTube Video, Self Organizing Maps, Online: https://www.youtube.com/watch?v=5CvJWtxytIk
- [8] YouTube Video, Unsupervised Learning Explained, Online: https://www.youtube.com/watch?v=lEfrr0Yr684

Acknowledgements



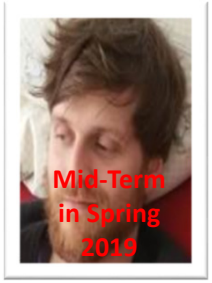
Acknowledgements – High Productivity Data Processing Research Group



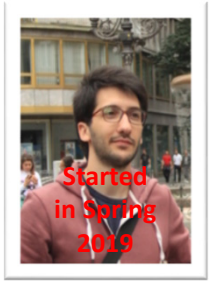
PD Dr.
G. Cavallaro

Senior PhD
Student A.S. Memon

Senior PhD
Student M.S. Memon



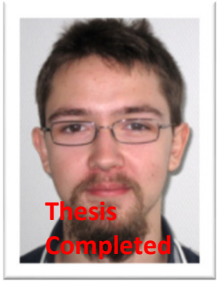
PhD Student
E. Erlingsson



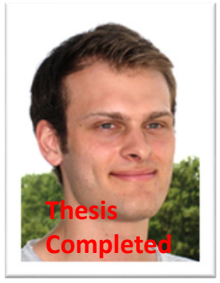
PhD Student
S. Bakarat

PhD Student
R. Sedona

Dr. M. Goetz
(now KIT)



MSc M.
Richerzhagen
(now other division)



MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now
Soccerwatch.tv)

MSc Student
G.S. Guðmundsson
(Landsverkjun)

This research group has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 763558 (DEEP-EST EU Project)


services
measurement
within
fusion
new
funding
Policy-based
concepts
device
Analysis
Centre
forms
cross-disciplinary
resources
climate
Computer
expertise
Approaches
construction
Fusion
year
like
Computational
Test
E.g
Computing
disciplines
results
Enable
Cross-Disciplinary
PB
community
Environment
Modelling
different
EU
methods
High
big
nodes
Key
scientific
important
Science
model
international
centers
storage
basis
Infrastructure
files
SMAQ
engineering
References
etc
DKRZ
Use
simulations
ENES
technologies
formats
brain
increasing
EUDAT
physical
IPCC
Big
Project
Data
via
processing
Common
TB
MapReduce
create
Methods
computing
Using
European
databases
GB
Supercomputing
Work
images
codes
analysis
stored
Resources
analyze
Large
Summary
data-intensive
Service
Scientific
levels
often
use
DLCL
Level
Research
external
JSC
computational
Matters
Concrete
performance
research
e.g
science
HPC
Juelich
datasets
acquisition
large-scale
compute
system
Landscape
System
Climate
modelling
AR5
Hardware
access
hundreds
Nutshell
Services
management
DLCLs
Understanding
structures
Technologies
Earth
distributed
cores
Structure
current
network
Germany
Simulation
directory
project
General
Provide
NASA
Energy
systems
day
Processing
Health