Parallel & Scalable Machine Learning
Introduction to Machine Learning Algorithms

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

LECTURE 3

Unsupervised Clustering and Applications

March 6th, 2018
JSC, Germany
Review of Lecture 2

- **PRACE**
  - High Performance Computing (HPC)
  - Pan-European supercomputing infrastructure
  - Offers HPC resources on leading edge capability systems
  - Peer-review of research proposals

- **JURECA (Mon, Wed) & JUROPA3 (Tue)**
  - Tutorial HPC systems
  - Accounts train004-train050

- **Scheduler & Job scripts**
  - Define parallel job properties
  - Reservation in job scripts change daily
  - Job submit with sbatch <jobscript>; check status with squeue –u userid

[1] LLView Tool
Outline
Outline of the Course

1. Introduction to Machine Learning Fundamentals
2. DEEP Projects and Parallel Computing Basics
3. Unsupervised Clustering and Applications
4. Unsupervised Clustering Challenges & Solutions
5. Supervised Classification and Learning Theory Basics
6. Classification Applications, Challenges, and Solutions
7. Support Vector Machines and Kernel Methods
8. Practicals with SVMs
9. Validation and Regularization Techniques
10. Practicals with Validation and Regularization
11. Parallelization Benefits
12. Cross-Validation Practicals

Day One – beginner
Day Two – moderate
Day Three – expert
Outline

- Unsupervised Clustering
  - Clustering Methods and Approaches
  - K-Means & K-Median Clustering Algorithms
  - R Introduction & Simple Application Examples
  - DBSCAN Clustering Algorithm
  - Parallel HPDBSCAN & HDF5 Data Format

- Point Cloud Applications
  - Introduction to Application Domain
  - BremenDatasets & Locations
  - Hierarchical Data Format (HDF) Basics
  - Parallel HPDBSCAN Algorithm
  - Different Application Examples
Unsupervised Clustering
Methods Overview

- Statistical data mining methods can be roughly categorized in classification, clustering, or regression augmented with various techniques for data exploration, selection, or reduction.

### Classification
- Groups of data exist
- New data classified to existing groups

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

### Regression
- Identify a line with a certain slope describing the data
What means Learning – Revisited

- 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

- **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

- **Reinforcement Learning**
  - Typical ‘human way’ of learning
  - Example: Toddler tries to touch a hot cup of tea (again and again)
Learning Approaches – Unsupervised Learning – Revisited

- Each observation of the predictor measurement(s) has no associated response measurement:
  - Input \( x = x_1, \ldots, x_d \)
  - No output
  - Data \( (x_1), \ldots, (x_N) \)

- Goal: Seek to understand relationships between the observations
  - Clustering analysis: check whether the observations fall into distinct groups

- Challenges
  - No response/output that could supervise our data analysis
  - Clustering groups that overlap might be hardly recognized as distinct group

---

Unsupervised learning approaches seek to understand relationships between the observations
Unsupervised learning approaches are used in clustering algorithms such as k-means, etc.
Unsupervised learning works with data = [input, ---]

[2] An Introduction to Statistical Learning
Learning Approaches – Unsupervised Learning Use Cases

- **Earth Science Data (PANGAEA, cf. Lecture 1)**
  - Automatic quality control and event detection
  - Collaboration with the University of Gothenburg
  - Koljoefjords Sweden – Detect water mixing events

- **Human Brain Data**
  - Analyse human brain images as brain slices
  - Segment cell nuclei in brain slice images
  - Step in detecting layers of the cerebral cortex

- **Point Cloud Data**
  - Analysis of point cloud datasets of various sizes
  - 3D/4D LIDAR scans of territories (cities, ruins, etc.)
  - Filter noise and reconstruct objects

This clustering lecture uses a point cloud dataset of the city of Bremen as one concrete example
Unsupervised Learning – Earth Science Data Example

- Earth Science Data Repository
  - Time series measurements (e.g. salinity)
  - Millions to billions of data items/locations
  - Less capacity of experts to analyse data

- Selected Scientific Case
  - Data from Koljöfjords in Sweden (Skagerrak)
  - Each measurement small data, but whole sets are ‘big data’
  - Automated water mixing event detection & quality control (e.g. biofouling)
  - Verification through domain experts
Unsupervised Learning – Human Brain Data Example

Research activities jointly with T. Dickscheid et al. (Juelich Institute of Neuroscience & Medicine)
Learning Approaches – Unsupervised Learning Challenges

- Practice: The number of clusters can be ambiguities
Unsupervised Learning – Different Clustering Approaches

- Clustering approaches can be categorized into four different approaches:
  1. hierarchical
  2. centroid
  3. density
  4. distribution
Unsupervised Learning – Clustering Methods

- Characterization of clustering tasks
  - No prediction as there is no associated response Y to given inputs X
  - Discovering interesting facts & relationships about the inputs X
  - Partitioning of data in subgroups (i.e. ‘clusters’) previously unknown
  - Being more subjective (and more challenging) than supervised learning

- Considered often as part of ‘exploratory data analysis’
  - Assessing the results is hard, because no real validation mechanism exists
  - Simplifies data via a ‘small number of summaries’ good for interpretation

- Clustering are a broad class of methods for discovering previously unknown subgroups in data
Selected Clustering Methods

- **K-Means Clustering** – Centroid based clustering
  - Partitions a data set into K distinct clusters (centroids can be artificial)

- **K-Medoids Clustering** – Centroid based clustering (variation)
  - Partitions a data set into K distinct clusters (centroids are actual points)

- **Sequential Agglomerative hierarchic nonoverlapping (SAHN)**
  - Hierarchical Clustering (create tree-like data structure → ‘dendrogram’)

- **Clustering Using Representatives (CURE)**
  - Select representative points / cluster – as far from one another as possible

- **Density-based spatial clustering of applications + noise (DBSCAN)**
  - Assumes clusters of similar density or areas of higher density in dataset
Clustering Methods – Similarity Measures

- How to partition data into distinct groups?
  - Data in same (homogenous) groups are somehow ‘similar’ to each other
  - Data not in same sub-groups are somehow ‘different’ from each other
  - Concrete definitions of ‘similarity’ or ‘difference’ often domain-specific

- Wide variety of similarity measures exist, e.g. distance measures
  - Jaccard Distance, Cosine Distance, Edit Distance, Hamming Distance, ...

A distance measure in some space is a function \( d(x,y) \) that takes two points in the space as arguments and produces a real number.

- Often used ‘similarity measure’ example
  - Distance-based: Euclidean distance
    \[
    d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
    \] (ruler distance)
  - n-dimensional Euclidean space: A space where points are vectors of \( n \) real numbers
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 \( K \)

[Image of scatter plots with different values of \( k \)]

[2] An Introduction to Statistical Learning
Clustering Methods – K-Means Algorithm

0. Set the desired number of clusters $K$
   • Picking the right number $k$ is not simple (➔ later)

1. Randomly assign a number from 1 to $K$ to each observation
   • Initializes cluster assignments for the observations
   • Requires algorithm execution multiple times
     (results depend on random assignment, e.g. pick ‘best’ after 6 runs)

2. Iterate until the cluster assignments stop changing
   a. For each of the $K$ clusters: compute the cluster centroid
      • The $k$th cluster centroid is the vector of the $p$ feature means
        for all the observations in the $k$th cluster
   b. Assign each observation to the cluster $K$ whose centroid is closest
      • The definition of ‘closest’ is the Euclidean distance
1. Randomly assign a number from 1 to K to each observation
2. Iterate until the cluster assignments stop changing
   a. For each of the K clusters: compute the cluster centroid [centroids appear and move]
   b. Assign each observation to the cluster K whose centroid is closest [Euclidean distance]
Clustering Methods – K-Means Usage

**Advantages**
- Handles large datasets (larger than hierarchical cluster approaches)
- Move of observations / data elements between clusters (often improves the overall solution)

**Disadvantages**
- Use of ‘means’ implies that all variables must be continuous
- Severaly affected by datasets with outliers (→ means)
- Perform poorly in cases with non-convex (e.g. U-shaped) clusters

**‘Big Data’ Application Example**
- Image processing: 7 million images
- 512 features/attributes per image;
- 1 million clusters
- 10000 Map tasks; 64GB broadcasting;
- 20 TB intermediate data in shuffling;

[Video] K-Means Clustering

[5] Animation of the k-means clustering algorithm, YouTube Video
The tool R is a free software environment

- Many **functions/algorithms** used for statistical computing and graphics
- It is a command-line tool with many **libraries** to download ‘instantly’
- Despite of command-line, there are **sophisticated graphics possible**

**Usage**

- R uses **functions** to perform operations, use `?funcname` for help
- Call a function with arguments/inputs: `funcname(input1, input2)`

**Selected Hints**

- Hitting `[up]` n times for previous commands (e.g. slightly modify)
- `[strg+l]` clears console

```
[train001@jrl06 tools]$ module load GCC/7.2.0

Due to MODULEPATH changes, the following have been reloaded:
1) binutils/.2.29

The following have been reloaded with a version change:
1) GCCcore/.5.4.0 => GCCcore/.7.2.0

[train001@jrl06 tools]$ module load ParaStationMPI/5.2.0-1
[train001@jrl06 tools]$ module load R/3.4.2
```
Exercises – Use K-Means Algorithm in R changing K values
Serial Tool: Statistical Computing with R – Startup

- Remember to load modules!

```
[train001@jrl06 tools]$ R

R version 3.4.2 (2017-09-28) -- "Short Summer"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

> ]
Clustering Methods – K-Means with R

- **Function** `kmeans()`
  - `kmeans(x, c, iter.max, nstart, alg, trace)`

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Numeric matrix of data</td>
</tr>
<tr>
<td>c</td>
<td>Centers: number of k clusters or set of initial (distinct) cluster centres</td>
</tr>
<tr>
<td>iter.max</td>
<td>maximum number of iterations</td>
</tr>
<tr>
<td>nstart</td>
<td>If centers number k: amount of random sets</td>
</tr>
<tr>
<td>alg</td>
<td>Different types of Algorithms (default:</td>
</tr>
<tr>
<td>trace</td>
<td>true/false: trace information on the algorithm progress</td>
</tr>
</tbody>
</table>
Clustering Methods – K-Means with R Example

- Prepare artificial dataset
  - Input x: 50 observations; two dimensional data
  - \texttt{set.seed()} function to guarantee reproducible results
    - \texttt{x = matrix(rnorm(50*2), ncol=2)}
    - \texttt{x[1:25,1]=x[1:25,1]+3}
    - \texttt{x[1:25,2]=x[1:25,2]-4}

- Call function \texttt{kmeans()}
  - \( K = 2 \)
  - Output placeholder \texttt{km.out}

- Retrieve cluster assignments \texttt{km.out$cluster}
  - Visualize clusters better with \texttt{plot()}
    - \texttt{plot(x, col=(km.out$cluster+1),main="K-means clustering results (k=2)", + xlab="",ylab="",pch=20,cex=2)}
Example: Visualize data points with `plot()`
- Using different colors for data points

```r
> plot(x, col=1, main="K-means clustering results (k=2)", xlab="", ylab="", pch=16, cex=2)
```

- For e.g. ‘multi-class’ problems above 8 colors
  - Use different ‘data point types’
[R Tool] Data Visualization – Different Data Point Types

- Example: Clustering output with `plot()`
  - Using different types for data points, `cex = [1,2,..]` magnifies

```r
> plot(x, col=(km.out$cluster+1),main="K-means clustering results (k=2)", xlab="", ylab="", pch=1, cex=1)
```
Working with Rattle: Load and Startup

- Rattle GUI for R
  - Uses a workspace
  - Use mouse instead of commands
  - Loaded with `library()`
  - Start with `rattle()`

```r
> library(rattle)
Rattle: Ein kostenloses grafisches Interface für Data Mining mit R.
Version 2.6.27 r142 Copyright (c) 2006-2013 Togaware Pty Ltd.
Geben Sie 'rattle()' ein, um Ihre Daten mischen.
> rattle()
> 
[7] Rattle brochure
Working with Rattle: Load different files – Examples

- **Scientific measurement data – Koljoefjords**
  - data.tab → original dataset
  - data_ok.tab → header removed
  - data_reform.tab → header reformatted

- **Shop data – Reykjavik area**
  - Shop.csv → shop data
  - Challenges: different languages / encoding

- **Scientific big data example – brain images**
  - ~700 images: ~40 GB, ~14 MB/image RGB, ~8MB/image mask)
  - Challenges: Data representation, e.g. brain slice images
  - Challenges: Memory limits in serial programs
  - Possible solutions: smart sampling and/or parallelization

- **Serial tools like R, Matlab, scikit-learn show limitations when working on large datasets (big data)**
Clustering Methods – K-Means with Rattle
Selected Clustering Methods

- **K-Means Clustering** – Centroid based clustering
  - Partitions a data set into $K$ distinct clusters (centroids can be artificial)

- **K-Medoids Clustering** – Centroid based clustering (variation)
  - Partitions a data set into $K$ distinct clusters (centroids are actual points)

- **Sequential Agglomerative hierarchic nonoverlapping (SAHN)**
  - Hierarchical Clustering (create tree-like data structure → ‘dendrogram’)

- **Clustering Using Representatives (CURE)**
  - Select representative points / cluster – as far from one another as possible

- **Density-based spatial clustering of applications + noise (DBSCAN)**
  - Assumes clusters of similar density or areas of higher density in dataset
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 \( \text{epsilon} \)
  - A cluster consist of a given minimum number of points
    → Parameter \( \text{minPoints} \)
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
Exercises – Use DBSCAN Algorithm in R
[Video] DBSCAN Clustering

[9] DBSCAN, YouTube Video
Point Cloud Applications
Point Cloud Applications

- ‘Big Data’: 3D/4D laser scans
  - Captured by robots or drones
  - Millions to billion entries
  - Inner cities (e.g. Bremen inner city)
  - Whole countries (e.g. Netherlands)

- Selected Scientific Cases
  - Filter noise to better represent real data
  - Grouping of objects (e.g. buildings)
  - Study level of continuous details
Point Cloud Application Example – Within Buildings

- Point based rendering example
  - Aachen Cathedral based on 3D laser scans and photos
  - Points are rendered as textured and blended splats
  - Visualisation can run in real-time on a desktop PC showing 6 million splats based of a 120 million point laser scan

[10] Aachen Cathedral Point Cloud Rendering, YouTube Video
Bremen Dataset & Locations – Attention: Your Own Copy!

- Different clusterings of the inner city of Bremen
  - Using smart visualizations of the point cloud library (PCL)
  - Big Bremen (81 mio points) & sub sampled Small Bremen (3 mio points)

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

```bash
[train001@jrl07 bremen]$ pwd
/homea/hpclab/train001/data/bremen
[train001@jrl07 bremen]$ ls -al
```

```
total 1342208
   drwxr-xr-x 2 train001 hpclab  512 Jan 14  09:58 .
   drwxr-xr-x 4 train001 hpclab  512 Jan 14  08:38 ..
-rw-r--r-- 1 train001 hpclab 1302382632 Jan 14  09:56 bremen.h5
-rw-r--r-- 1 train001 hpclab   72002416 Jan 14  08:25 bremenSmall.h5
```
Exercises – Explore & Copy Bremen HDF5 Datasets (binary)
Exercises – Explore & Copy Bremen HDF5 Datasets (binary)

- Copy Bremen datasets to your own home directory (~)
  
  ```
  [train001@jrl07 bremen]$ pwd
  /homea/hpclab/train001/data/bremen
  [train001@jrl07 bremen]$ cp * ~
  ```

- Check your home directory for the Bremen datasets
  
  ```
  [train001@jrl07 bremen]$ cd ~
  [train001@jrl07 ~]$ ls -al
  total 1341824
  drwxr-x--- 13 train001 hpclab 32768 Jan 14 09:44 .
  drwxr-xr-x  302 root  sys  32768 Mar 25 2013 ..
  -rw-------  1 train001 hpclab  7547 Jan 14 08:28 .bash_history
  -rw-r--r--  1 train001 hpclab  18 Jan 8 08:58 .bash_logout
  -rw-r--r--  1 train001 hpclab  176 Jan 8 08:58 .bash_profile
  -rw-r--r--  1 train001 hpclab 124 Jan 8 08:58 .bashrc
  drwxr-xr-x  3 train001 hpclab  512 Jan 14 00:28 bin
  -rw-r--r--  1 train001 hpclab 1302382632 Jan 14 09:59 bremen.h5
  -rw-r--r--  1 train001 hpclab  72002416 Jan 14 09:59 bremenSmall.h5
  ```

- Notice binary content
  
  ```
  [train001@jrl07 ~]$ head bremen.h5
  @HDF
  ```
Hierarchical Data Format (HDF)

- HDF is a technology suite that enables the work with extremely large and complex data collections

- Simple ‘compound type’ example:
  - Array of data records with some descriptive information (5x3 dimension)
  - HDF5 data structure type with int(8); int(4); int(16); 2x3x2 array (float32)

‘HDF5 file is a container’ to organize data objects
HDF5 – Parallel I/O: Shared file

- Each process performs I/O to a single file
  - The file access is ‘shared’ across all processors involved
  - E.g. MPI/IO functions represent ‘collective operations’

- Scalability and Performance
  - ‘Data layout’ within the shared file is crucial to the performance
  - High number of processors can still create ‘contention’ for file systems

- Parallel I/O: shared file means that processes can access their ‘own portion’ of a single file
- Parallel I/O with a shared file like MPI/IO is a scalable and even standardized solution
HDF5 – Parallel I/O & File Systems

- Portable Operating System Interface for UNIX (POSIX) I/O
  - Family of standards to maintain OS compatibility, including I/O interfaces
  - E.g. read(), write(), open(), close(), ...(very old interface, some say ‘too old’)
- ‘Higher level I/O libraries’ HDF5 & NETCDF
  - Integrated into a parallel application
  - Built on top of MPI I/O for portability
  - Offers machine-independent data access and data formats

Hierarchical Data Format (HDF) is designed to store & organize large amounts of numerical data
Parallel Network Common Data Form (NETCDF) is designed to store & organize array-oriented data

[13] HDF Group
[14] Parallel NETCDF
I/O with Multiple Layers and Distinct Roles

- High-Level I/O Library
  - Maps application abstractions to a structured portable file format
  - E.g. HDF-5, Parallel NetCDF

- I/O Middleware
  - E.g. MPI I/O
  - Deals with organizing access by many processes

- Parallel Filesystem
  - Maintains logical space and provides efficient access to data
  - E.g. GPFS, Lustre, PVFS

Exercises – Bremen HDF5 Viewer
Review of Parallel DBSCAN Implementations

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

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

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

---

Exercises – Bremen Small HPDBSCAN Runs
HPC Environment – Modules Revisited

- **Module** environment tool
  - Avoids to manually setup environment information for every application
  - Simplifies shell initialization and lets users easily modify their environment

- **Module avail**
  - Lists all available modules on the HPC system (e.g. compilers, MPI, etc.)

- **Module spider**
  - Find modules in the installed set of modules and more information

- **Module load → needed before HPDBSCAN run**
  - Loads particular modules into the current work environment, E.g.:
    ```
    [train001@jrl12 ~]$ module load GCC
    
    Due to MODULEPATH changes, the following have been reloaded:
    1) binutils/.2.29
    
    The following have been reloaded with a version change:
    1) GCCcore/.5.4.0 => GCCcore/.7.2.0
    
    [train001@jrl12 ~]$ module load ParaStationMPI/5.2.0-1
    [train001@jrl12 ~]$ module load HDF5/1.8.19
    ```
#!/bin/bash
#SBATCH --job-name=HPDBSCAN
#SBATCH -o HPDBSCAN-%j.out
#SBATCH -e HPDBSCAN-%j.err
#SBATCH --nodes=2
#SBATCH --ntasks=4
#SBATCH --ntasks-per-node=4
#SBATCH --time=00:20:00
#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
BREMENSMALLDATA=/homea/hpclab/train001/bremenSmall.h5

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

srun $HPDBSCAN -m 100 -e 300 -t 12 $BREMENSMALLDATA

- 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>`

Note the tutorial reservation with `-reservation=ml-hpc-1` just valid for today on JURECA
JURECA HPC System – HPDBSCAN Job Submit

- Load module environment (once after login)

  
  ```
  [train001@jrl07 jsc_mpi]$ module load GCC

  Due to MODULEPATH changes, the following have been reloaded:
  1) binutils/.2.29

  The following have been reloaded with a version change:
  1) GCCcore/.5.4.0 => GCCcore/.7.2.0
  ```

- Submit job via jobscript

  ```
  [train001@jrl07 jsc_mpi]$ sbatch submit-clustering-bremen.sh
  Submitted batch job 4629728
  ```

- Check job status (and cancel if needed)

  ```
  [train001@jrl07 hpdbscan]$ squeue -u train001
  JOBID PARTITION   NAME    USER   ST   TIME  NODES NODENAME(Reason)
  4629867    batch    HPDBSCAN train001   R   2:20 2 jrc[0672-0673]
  ```

  ```
  [train001@jrl07 hpdbscan]$ scancel 4629867
  ```

  ```
  [train001@jrl07 hpdbscan]$ squeue -u train001
  JOBID PARTITION   NAME    USER   ST   TIME  NODES NODENAME(Reason)
  4629867    batch    HPDBSCAN train001   CG   2:34 2 jrc[0672-0673]
  ```

  (scancel might take a second or two to take effect)
The outcome of the clustering process is written directly into the HDF5 file using cluster IDs and noise IDs.

Result...
65 Clusters
2973821 Cluster Points
26179 Noise Points
2953129 Core Points
Took: 59.111594s
HDFView Example – Bremen Output

- HDFView is a visual tool for browsing and editing HDF files
  - Tools is using a GUI thus needs ssh –X when log into JURECA

```
$ module load HDFView/2.14-Java-1.8.0_144
$ hdfview.sh
```
Point Cloud Viewer Example – Bremen Output

adminuser@linux-Bdjg:~$ ssh -X vsc42544@login.hpc.ugent.be
Last login: Wed Nov 22 16:16:28 2017 from 91.177.4.215

STEVIN HPC-UGent infrastructure status on Thu, 23 Nov 2017 02:15:01

| cluster - full - free - part - total - running - queued |
|----------|--------|--------|--------|---------|---------|---------|
|          | nodes  | nodes  | nodes  | jobs    | jobs    |
| delcatty | 157    | 0      | 8      | 159     | N/A     | N/A     |
| gelett   | 56     | 45     | 53     | 196     | N/A     | N/A     |
| phanpy   | 9      | 0      | 7      | 16      | N/A     | N/A     |
| rahcu    | 34     | 0      | 22     | 56      | N/A     | N/A     |
| swalot   | 110    | 0      | 18     | 128     | N/A     | N/A     |

For a full view of the current loads and queues see:
http://hpc.ugent.be/clusterstatus/
Updates on maintenance and unscheduled downtime can be found on

/usr/bin/xauth:  file /user/home/gent/vsc425/vsc42544/.Xauthority does not exist

[vsc42544@gligar03 Bremen]$ module load PCL/1.8.1-intel-2017b-Python-2.7.14
[vsc42544@gligar03 Bremen]$ pwd
/apps/gent/tutorials/machine_learning/clustering/Bremen
[vsc42544@gligar03 Bremen]$ ls -al
total 3431816
drwxr-xr-x 2 vsc40003 vsc40003 4096 Nov 22 22:39 .
drwxr-xr-x 5 vsc40003 vsc40003 4096 Nov 22 22:39 ..
-rw-r--r-- 1 vsc40003 vsc40003 382559971 Nov 22 22:39 bremenClustered.pcd
-rw-r--r-- 1 vsc40003 vsc40003 1302382632 Nov 22 14:07 bremen.h5.h5
-rw-r--r-- 1 vsc40003 vsc40003 72602416 Jan 13 2017 bremenSmall.h5.h5
[vsc42544@gligar03 Bremen]$ pcl_viewer bremenClustered.pcd
Point Cloud Viewer Example – Bremen Output (2)

- Use Strg and Mouse Wheel to Zoom and use numbers of keyboard for different visualizations
Exercises – Bremen HPDBSCAN Check Outputs
[Video] Point Clouds

[18] Point Based Rendering of the Kaiserpfalz in Kaiserswerth, YouTube Video
Lecture Bibliography
Lecture Bibliography (1)

- [1] LLView Tool,
- [2] An Introduction to Statistical Learning with Applications in R,
  Online: [http://www-bcf.usc.edu/~gareth/ISL/index.html](http://www-bcf.usc.edu/~gareth/ISL/index.html)
- [3] PANGAEA Data Collection, Data Publisher for Earth & Environmental Science,
  Online: [http://www.pangaea.de/](http://www.pangaea.de/)
- [5] Animation of the k-means algorithm using Matlab 2013, YouTube Video,
  Online: [http://www.youtube.com/watch?v=5FmnJVv73fU](http://www.youtube.com/watch?v=5FmnJVv73fU)
- [6] Statistical Computing with R Tool,
  Online: [http://www.r-project.org/](http://www.r-project.org/)
- [7] Rattle brochure,
  Online: [https://www.youtube.com/watch?v=5E097ZLE95g](https://www.youtube.com/watch?v=5E097ZLE95g)
- [10] YouTube Video, ‘Point Based Rendering of the Aachen Cathedral’
  Online: [https://www.youtube.com/watch?v=X_wyoroo4co](https://www.youtube.com/watch?v=X_wyoroo4co)
  Online: [http://hdl.handle.net/11304/6eacaa76-c275-11e4-ac7e-860aa0063d1f](http://hdl.handle.net/11304/6eacaa76-c275-11e4-ac7e-860aa0063d1f)


[18] YouTube Video, ‘Point Based Rendering of the Kaiserpfalz in Kaiserswerth’, Online: https://www.youtube.com/watch?v=KvDb58YvlvQ
Working with Vectors

- E.g. creating a vector using the concatenate function `c()`
  - Assigning values can be done using `<-` and `=`
  - Be careful with **overwriting** (see below, `x` was overwritten)

```r
> x <- c(1, 3, 2, 5)
> x
[1] 1 3 2 5
> x = c(1, 6, 2)
> x
[1] 1 6 2
> |
```

- E.g. number of elements using `length()`

```r
> length (x)
[1] 3
> |
```
Useful Working Commands: List and Remove Objects

- List all already defined objects (data and functions) with `ls()`

```r
> y = c(1,4,3)
> z = c(0,0,7)
> ls()
[1] "ds_140518213752"  "x"  "y"  "z"
> |
```

- Remove objects with `rm()`

```r
> ls()
[1] "ds_140518213752"  "x"  "y"  "z"
> rm (ds_140518213752)
> ls()
[1] "x"  "y"  "z"
> |
```
Working with Matrices

- E.g. creating a matrix using the function `matrix()`
  - Different versions exist, here we use the function with three parameters
  - It takes a number of inputs: matrix data, # rows, and # columns
  - You may specify parameter names: e.g. `nrow=2`
  - Default: filling columns, filling rows use parameter `byrow=TRUE`

```r
> x = matrix(c(1,2,3,4), nrow=2, ncol=2) > x = matrix(c(1,2,3,4), 2, 2, byrow=TRUE)
> x
   [,1] [,2]
[1,]  1  3
[2,]  2  4
> x = matrix(c(1,2,3,4), 2, 2)
> x
   [,1] [,2]
[1,]  1  3
[2,]  2  4
> |
```
Working with sqrt and power

- E.g. applying each element of a matrix with `sqrt()`
  - Remember: you not changing x

```
> x
[,1] [,2]
[1,] 1  2
[2,] 3  4
> sqrt(x)
[,1]      [,2]
[1,] 1.000000 1.414214
[2,] 1.732051 2.000000
```

- E.g. applying each element of a matrix with `power ^`
  - Raises each element of x to the power 2
  - Remember: you not changing x

```
> x
[,1] [,2]
[1,] 1  2
[2,] 3  4
> x^2
[,1] [,2]
[1,] 1  4
[2,] 9 16
```
Working with Random Variables

- E.g. creating 100 random normal variables with `rnorm()`
  
  ```r
  > x = rnorm(100)
  > y = rnorm(100)
  > plot(x, y)
  ```

- E.g. visualizing random variables in a simple diagram with `plot()`
Working with Datasets – Weather Patterns Example

- **library(rattle)**
  - Loads the Rattle package and the associated datasets into the memory

- **weatherAUS**
  - Loads in the weatherAUS dataset

- **names(weatherAUS)**
  - Shows the variables Names

- **nrow(weatherAUS)**
  - Displays the number of rows (observations on the longest variable)

- **ncol(weatherAUS)**
  - Displays the number of columns (variables)

- **head(weatherAUS)**
  - First six records of the dataset.

- **tail(weatherAUS)**
  - The last six rows of the dataset.

- **sample(weatherAUS)**
  - A snapshot of some of the data
Working with data: weatherAUS

- Load the already available Dataset
  - **weatherAUS**
  - Loads in the weatherAUS dataset
  - Dataset will be listed (lots of rows)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4151</td>
<td>13.9</td>
<td>No</td>
<td>1.2</td>
<td>Yes</td>
</tr>
<tr>
<td>4152</td>
<td>18.4</td>
<td>Yes</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4153</td>
<td>22.1</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4154</td>
<td>25.5</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4155</td>
<td>26.4</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4156</td>
<td>30.3</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4157</td>
<td>30.6</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4158</td>
<td>30.1</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4159</td>
<td>31.1</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4160</td>
<td>20.8</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4161</td>
<td>22.6</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4162</td>
<td>27.4</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4163</td>
<td>27.4</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4164</td>
<td>25.4</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4165</td>
<td>19.8</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>4166</td>
<td>23.3</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
</tbody>
</table>
Weather Patterns – Get a Feel for a Dataset (1)

- Look at the names of the variables
  - `names(weatherAUS)`
  - Shows the variables Names

```r
> names(weatherAUS)
[1]  "Date"     "Location"   "MinTemp"   "MaxTemp"
[5]  "Rainfall" "Evaporation" "Sunshine"  "WindGustDir"
[9] "WindGustSpeed" "WindDir9am" "WindDir3pm" "WindSpeed9am"
[13] "WindSpeed3pm" "Humidity9am" "Humidity3pm" "Pressure9am"
[17] "Pressure3pm" "Cloud9am"    "Cloud3pm"   "Temp9am"
[21] "Temp3pm"   "RainToday"   "RISK_MM"   "RainTomorrow"
```
Weather Patterns – Get a Feel for the Dataset (2)

- Look at the number of variables and the number of observations
  - `nrow(weatherAUS)`
  - Displays the number of rows (observations on the longest variable)

```r
> nrow(weatherAUS)
[1] 75136
```

- `ncol(weatherAUS)`
- Displays the number of columns (variables)

```r
> ncol(weatherAUS)
[1] 24
```
Weather Patterns – Get Knowledge about the Dataset (1)

- **Look at the Head**
  - `head(weatherAUS)`
  - Displays first six records of the dataset

```
> head(weatherAUS)

      Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir WindGustSpd
1 2008-12-01  Albury     13.4      22.9       0.6          NA       NA          W         44
2 2008-12-02  Albury      7.4      25.1       0.0          NA       NA         WNW         44
3 2008-12-03  Albury     12.9      25.7       0.0          NA       NA         WSW         46
4 2008-12-04  Albury     9.2       28.0       0.0          NA       NA          NE         24
5 2008-12-05  Albury     17.5      32.3       1.0          NA       NA          W         41
6 2008-12-06  Albury     14.6      29.7       0.2          NA       NA         WNW         56
```

- **WindDir9am WindDir3pm WindSpd9am WindSpd3pm Humidity9am Humidity3pm Pressure9am Pressure3pm**

```
> head(weatherAUS)

     WindDir9am WindDir3pm WindSpd9am WindSpd3pm Humidity9am Humidity3pm Pressure9am Pressure3pm
1           W         WNW          20         24          71          22     1007.7     1007.1
2          NNW        WSW          4          22          44          23     1010.6     1007.5
3            W        WSW          9          26          38          30     1007.6     1008.7
4           SE          E          11          9          45          16     1017.6     1012.5
5           ENE         NW          7          20          82          33     1010.8     1006.0
6           W           W          19          24          55          23     1009.2     1005.4
```

- **Cloud9am Cloud3pm Temp9am Temp3pm RainToday RISK MM RainTomorrow**

```
> head(weatherAUS)

      Cloud9am Cloud3pm Temp9am Temp3pm RainToday RISK MM RainTomorrow
1           NA         NA       16.9     21.8          NO          0.0            No
2           NA         NA       17.2     24.3          NO          0.0            No
3           NA         NA       21.0     23.2          NO          0.0            No
4           NA         NA       18.1     26.8          NO          1.0            No
5           NA         NA       17.8     29.7          NO          0.2            No
6           NA         NA       20.6     28.9          NO          0.0            No
```
Weather Patterns – Get Knowledge about the DataSet (2)

- Look at the Tail
  - `tail(weatherAUS)`
  - Displays last six records of the dataset

```r
> tail(weatherAUS)

    Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir WindGustSpeed WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressure9am Pressure3pm Cloud9am Cloud3pm Temp9am Temp3pm RainToday RISK_MH RainTomorrow
 75131 2013-07-25  Darwin   17.0   30.3     0     9.0   11.2       ESE         50
 75132 2013-07-26  Darwin   17.3   30.6     0    10.2   11.2       ESE         35
 75133 2013-07-27  Darwin   17.6   31.0     0     7.4   10.4       NW          30
 75134 2013-07-28  Darwin   18.9   30.6     0     5.8    7.8       NNW         20
 75135 2013-07-29  Darwin   17.6   32.3     0     4.4   10.1       ESE         37
 75136 2013-07-30  Darwin   18.5   33.0     0     4.4   10.9       ENE         50
     7
```
Weather Patterns – Get Knowledge about the DataSet (2)

3. Look at the Sample
   - \texttt{sample(weatherAUS)}
   - A snapshot of some of the data

\begin{verbatim}
4145  22  1  No  No  1025.6  3  41
4146  17  5  No  No  1019.4  2  65
4147  13  1  No  No  1020.6  2  30
4148  20  8  No  Yes  1019.1  7  35
4149  24  7  Yes  No  1016.9  4  56
4150  22  0  No  No  1021.8  1  46
4151  17  7  No  Yes  1020.2  2  54
4152  9   0  Yes  No  1026.0  0  24
4153  6   1  No  No  1026.7  0  24
4154  13  1  No  No  1026.2  0  39
4155  6   1  No  No  1024.5  NA  67
4156  17  NA  No  No  1020.9  NA  43
4157  9   NA  No  No  1017.9  NA  48
4158  7   7  No  No  1018.0  6  41
4159  17  0  No  No  1015.4  0  67
4160  28  1  No  No  1014.8  4  52
4161  7   0  No  No  1023.6  0  24
4162  13  0  No  No  1022.6  0  39
4163  11  NA  No  No  1019.7  NA  33
4164  9   NA  No  No  1019.2  NA  37
4165  15  NA  No  No  1019.1  NA  41
4166  24  1  No  No  1020.9  6  48
\end{verbatim}
Read tab data file (1)

- Read tab separated data from a real science project
  - Often different than UCI machine learning repository datasets
  - Example: measurement data with descriptive information first

```r
> data <- read.table("data.tab", sep="\t")

Error in scan(file, what, nmax, sep, dec, quote, skip, nlines, na.strings, :
  Zeile 24 hatte keine 2 Elemente

> data <- read.table("data.tab", sep="\t")
```

- Addressing the error:
  - Check if data.tab file may have descriptive information/comments
  - Descriptive information is sometimes put in front of the real data set (e.g. metadata = explaining where data was measured, by whom, etc.)
  - Removing metadata works if file is properly tab separated
  - What other surprised we encounter when we load the data?
Read tab data file (2)

- Look at the names of the variables/attributes
  ```
  > names(data)
  [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9" "V10" "V11" "V12" "V13" "V14" "V15" "V16"
  ```

- Look at one variable/attribute value only
  ```
  > data$V8
  ```

/* DATA DESCRIPTION:
Citation: Hall, Per (2012): Koljofjord cabled observatory RDCP data. Sweden (2012-03). Department of Chemistry, University of Gothenburg, Unpublished dataset #779644
Project(s): In situ monitoring of oxygen depletion in hypoxic ecosystems of coastal and open seas and land-locked water bodies (HYPOX) (URI: http://www.hypox.net/)
Coverage: LATITUDE: 58.228250 * LONGITUDE: 11.574000
   DATE/TIME START: 2012-03-01T00:13:16 * DATE/TIME END:
   2012-03-31T23:43:17
   MINIMUM DEPTH, water: 5.0 m * MAXIMUM DEPTH, water: 35.0
|
... Size: 168428 data points
*/

  Vbat [V]  Depth water [m]  CV hor [cm/s]  Direction
  [deg] CV vert [cm/s]  Sig str [dB]  Std dev [±]
2012-03-01T00:13:16
  5.000 9.311 339.974 1.791 -41.973 5.921
2012-03-01T00:13:16
  6.000 9.090 345.673 1.976 -42.196 6.497
Display Head of the data

```r
> head(data, n=100)

<table>
<thead>
<tr>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
<th>V6</th>
<th>V7</th>
<th>V8</th>
<th>V9</th>
<th>V10</th>
<th>V11</th>
<th>V12</th>
<th>V13</th>
<th>V14</th>
<th>V15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2012-03-01T00:00:00</td>
<td>0.01</td>
<td>0.32</td>
<td>35.5</td>
<td>2.01</td>
<td>330.9</td>
<td>3.01</td>
<td>3.01</td>
<td>3.01</td>
<td>3.01</td>
<td>3.01</td>
<td>3.01</td>
<td>3.01</td>
<td>3.01</td>
</tr>
<tr>
<td>2</td>
<td>2012-03-01T00:01:00</td>
<td>0.02</td>
<td>0.32</td>
<td>35.5</td>
<td>2.02</td>
<td>330.9</td>
<td>3.02</td>
<td>3.02</td>
<td>3.02</td>
<td>3.02</td>
<td>3.02</td>
<td>3.02</td>
<td>3.02</td>
<td>3.02</td>
</tr>
<tr>
<td>3</td>
<td>2012-03-01T00:02:00</td>
<td>0.03</td>
<td>0.32</td>
<td>35.5</td>
<td>2.03</td>
<td>330.9</td>
<td>3.03</td>
<td>3.03</td>
<td>3.03</td>
<td>3.03</td>
<td>3.03</td>
<td>3.03</td>
<td>3.03</td>
<td>3.03</td>
</tr>
<tr>
<td>4</td>
<td>2012-03-01T00:03:00</td>
<td>0.04</td>
<td>0.32</td>
<td>35.5</td>
<td>2.04</td>
<td>330.9</td>
<td>3.04</td>
<td>3.04</td>
<td>3.04</td>
<td>3.04</td>
<td>3.04</td>
<td>3.04</td>
<td>3.04</td>
<td>3.04</td>
</tr>
<tr>
<td>5</td>
<td>2012-03-01T00:04:00</td>
<td>0.05</td>
<td>0.32</td>
<td>35.5</td>
<td>2.05</td>
<td>330.9</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
<td>3.05</td>
</tr>
</tbody>
</table>
```

Lecture 3 – Unsupervised Clustering and Applications