

Task 1.1) OPTICS in high dimensional data

6 points

Download the Data Mining Tool WEKA and its documentation from

<http://www.cs.waikato.ac.nz/~ml/weka/>. Install the WEKA package and execute the tool.

Use the OPTICS algorithm implemented in **WEKA** to generate OPTICS plots for ARFF datasets given on the L2P website. There are 5 datasets (2d, 4d, 8d, 16d and 32d) in a ZIP file.

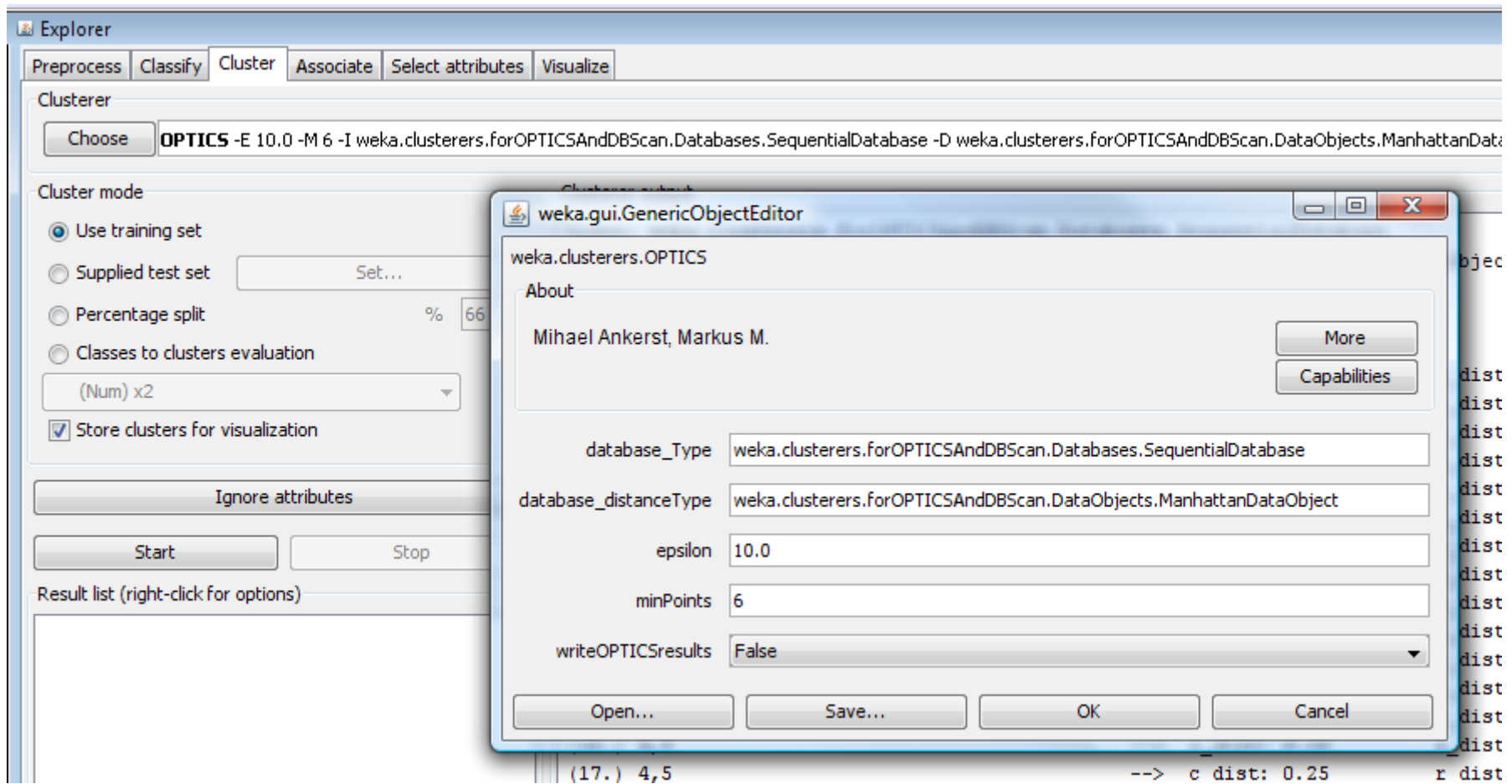
Use Manhattan Distance with MinPts=6 and $\epsilon = 10$ to compute an OPTICS plot for each of the given datasets.

Hand in the computed OPTICS plots and discuss the following questions:

- Can you detect hierarchical clusters?
- How has ϵ to be chosen to detect these clusters with the DBSCAN algorithm?
- Discuss the change in core and reachability distances over the data sets?

What are the reasons for the observed effects in the OPTICS plots?

Exercise 1.1

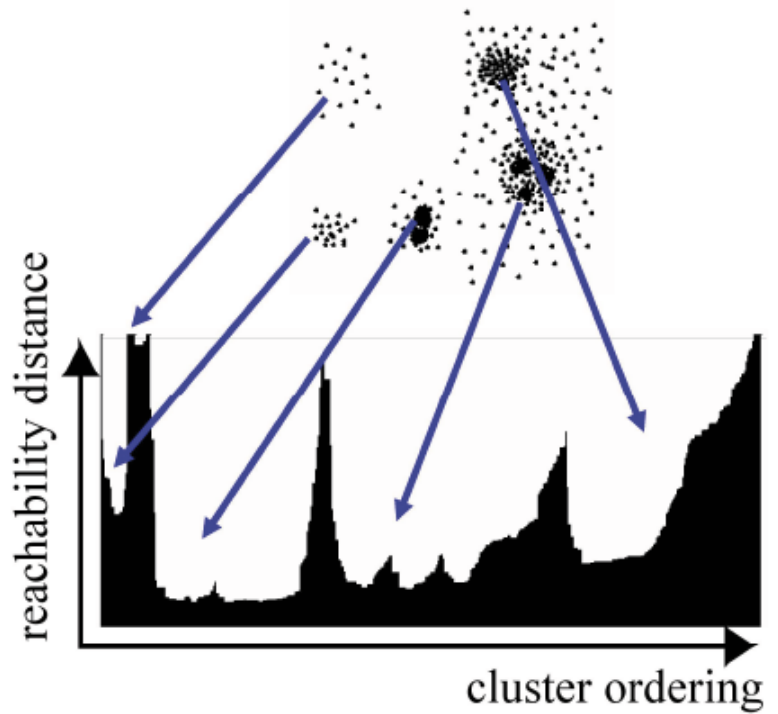
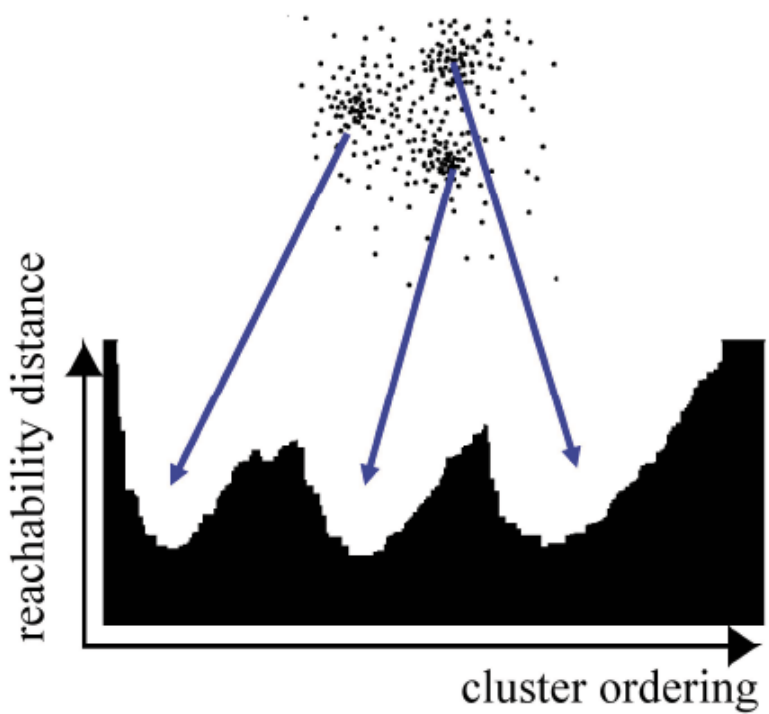


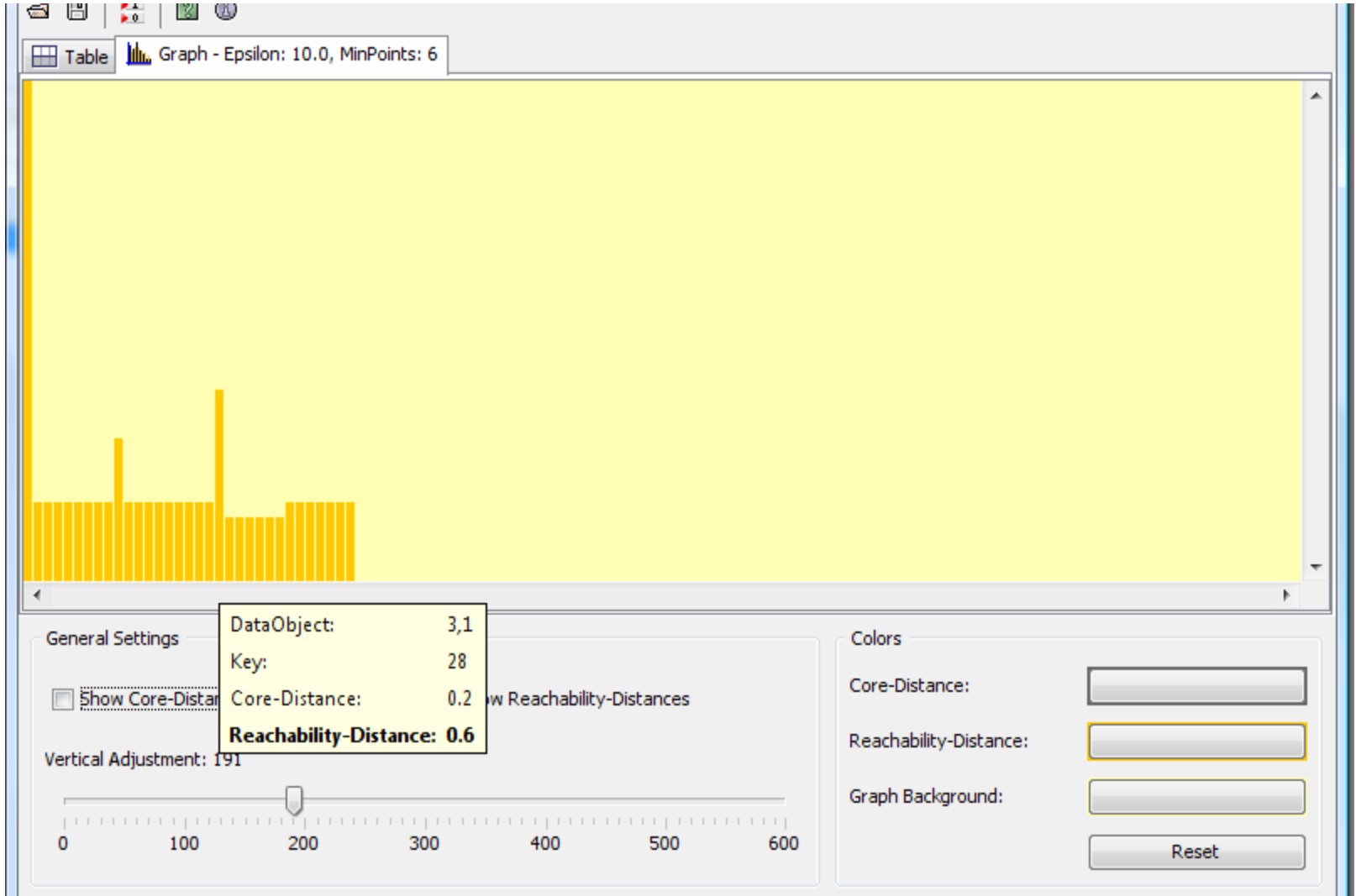
The screenshot shows the Weka Explorer interface with the 'Cluster' tab selected. A dialog box titled 'weka.gui.GenericObjectEditor' is open, displaying the configuration for the 'weka.clusterers.OPTICS' algorithm. The dialog includes an 'About' section with the authors 'Mihael Ankerst, Markus M.' and buttons for 'More' and 'Capabilities'. Below this, several parameters are listed in a table-like format:

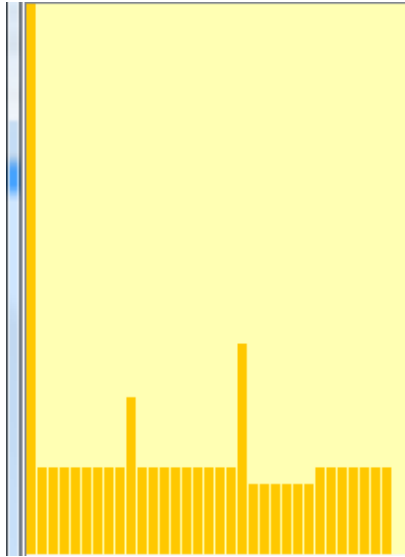
Parameter	Value
database_Type	weka.clusterers.forOPTICSAndDBScan.Databases.SequentialDatabase
database_distanceType	weka.clusterers.forOPTICSAndDBScan.DataObjects.ManhattanDataObject
epsilon	10.0
minPoints	6
writeOPTICSresults	False

At the bottom of the dialog are buttons for 'Open...', 'Save...', 'OK', and 'Cancel'. The background Explorer window shows the 'Clusterer' section with the 'OPTICS' algorithm selected and various options like 'Use training set' and 'Store clusters for visualization' checked.

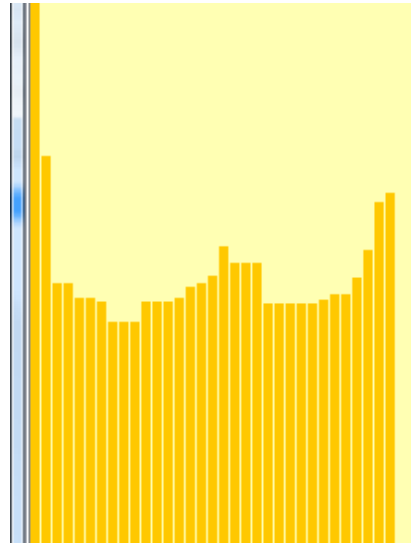
- represents the density-based clustering structure
- easy to analyze
- independent of the dimension of the data



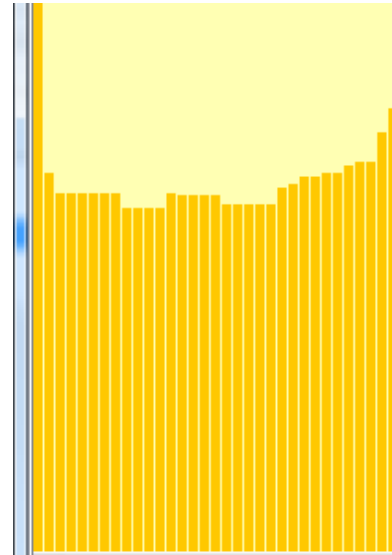




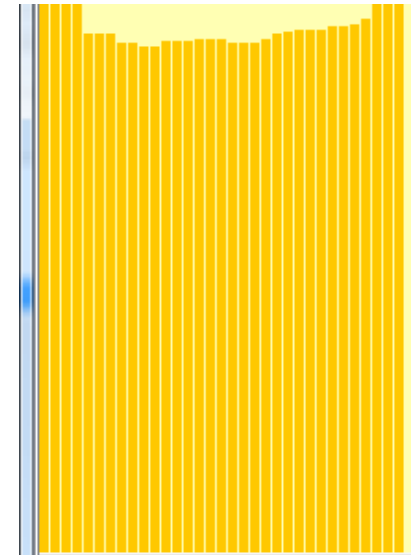
2D



4D

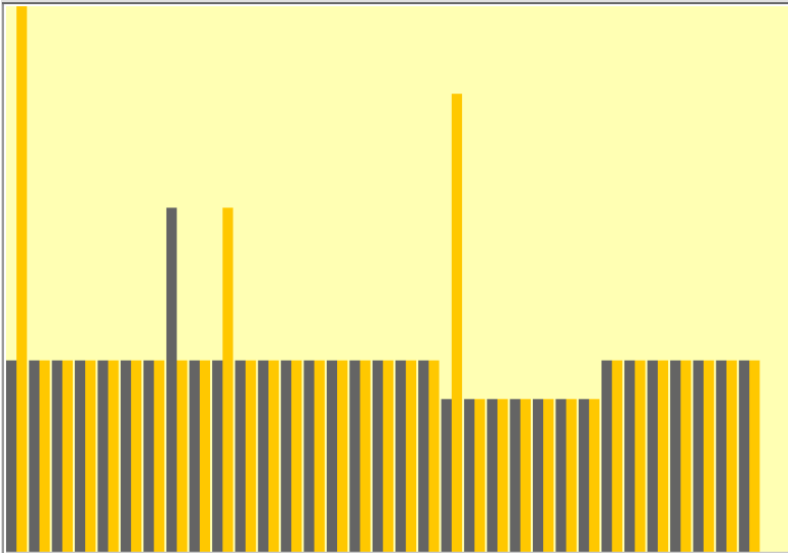


8D [similar to 16D]

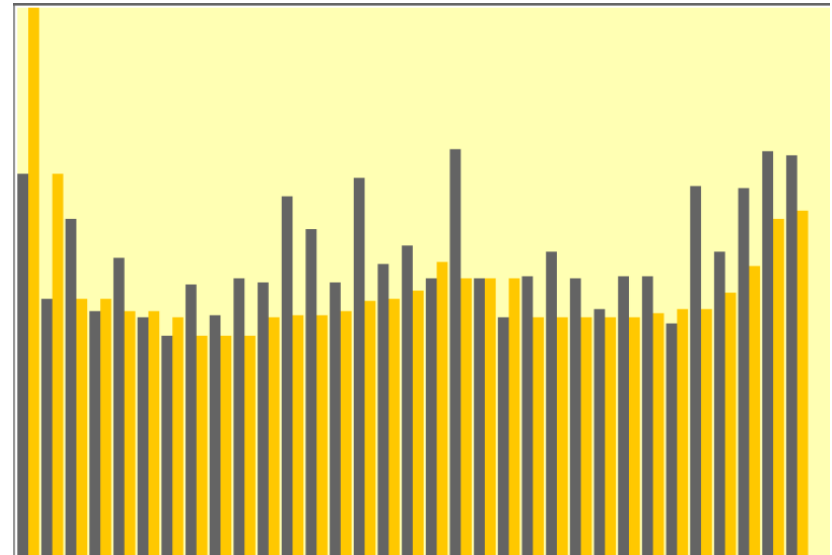


32D

- Can you detect hierarchical clusters?
 - How has ϵ to be chosen to detect these clusters with the DBSCAN algorithm?
1. Clusters \rightarrow yes, but no hierarchical clustering structure. With increasing dimensionality the reachability distances grow more and more alike, therefore the clustering structure becomes more and more obfuscated.
 2. Another consequence: with increasing dimensionality the range from which to choose a meaningful ϵ value becomes increasingly narrow.



2D



4D

- Can you detect hierarchical clusters?
 - How has ϵ to be chosen to detect these clusters with the DBSCAN algorithm?
 - Discuss the change in core and reachability distances over the data sets?
3. *With increasing dimensionality the distances grow, and more and more objects change from being core objects to being border objects or noise.*

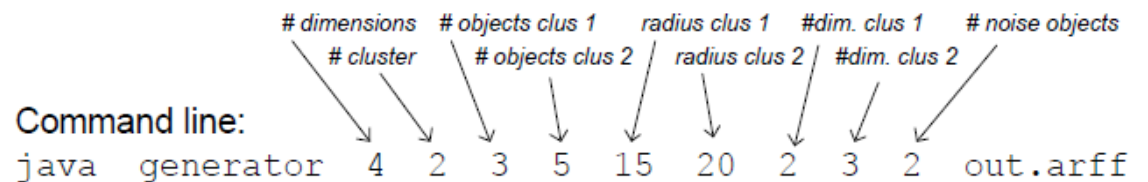
Exercise 1.2) Data generator

12 points

Implement a meaningful Java class ‘generator.java’ to generate a clustered data file in the *ARFF* format. The method shall generate a dataset of specified dimensionality d , where each dimension has the value range $[0,100]$. Furthermore the data shall have k hidden clusters, where the objects of each cluster are uniformly distributed within a specified radius. Additionally the generator shall have the option to generate outliers: objects that are uniformly distributed in the data space (range $[0,100]$). The generator shall be able to handle *subspace* clusters. Thus, for each cluster a certain dimensionality (i.e. the number of *relevant* dimensions for the cluster) can be specified. In the relevant dimensions the objects of a cluster are distributed within the given radius while in the non-relevant dimensions the objects are uniformly distributed in the data space (range $[0,100]$). For each cluster a different set of relevant dimensions is possible. State the relevant dimensions of each cluster in the header of the generated ARFF-File (see example below).

The following parameters are to be specified in the command line:

- number d of dimensions
- number k of clusters
- number of objects in the cluster (for each cluster)
- radius for the cluster *in the relevant dimensions* (for each cluster)
- dimensionality of the cluster (for each cluster)
- number of noise objects
- arff-output-file





```
this.filename = filename;

//generate cluster dimensions (subspaces)
generate_dims();
//generate clusters
generate_data();
//generate noise uniformly (they may lie in clusters by chance)
generate_noise(); // cf. last Exercise!

save_file();
}

/**
 * randomly generate relevant dimensions
 */
private void generate_dims()
{
    real_cluster_dims = new HashMap<Integer, SortedSet<Integer>>();
    for(int j=0; j<clusters; j++)
    {
        SortedSet<Integer> back = new TreeSet<Integer>();
        for(int i=0; i<cluster_dims[j]; i++)
        {
            int dim;
            //randomly pick a dimension
            do {
                dim = (int)Math.floor(random(0, dims));
            } while(back.contains(dim));
            back.add(dim);
        }
        real_cluster_dims.put(j, back);
    }
}
```



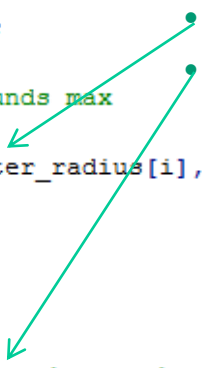

Choose this distance in the following

```
/**
 * computes euclidean distance based on relevant dimensions
 */
private double getDistance(int cluster, double[] point1, double[] point2)
{
    double distance = 0;
    for(int i=0; i<dims; i++)
    {
        if(real_cluster_dims.get(cluster).contains(i))
        {
            distance = distance + Math.pow((point1[i] - point2[i]), 2);
        }
    }
    return Math.sqrt(distance);
}

/**
 * computes euclidean distance based on all dimensions
 */
private double getDistance_old(double[] point1, double[] point2)
{
    double distance = 0;
    for (int i = 0; i < point1.length; i++)
    {
        distance = distance + Math.pow((point1[i] - point2[i]), 2);
    }
    return Math.sqrt(distance);
}
```

```
/**
 * Generates cluster centers, such that they do not overlap
 */
public void generate_cluster_centers()
{
    for(int i=0; i<clusters; i++)
    {
        boolean overlap = true;
        double[] new_center=new double[dims];
        int round=0;
        while(overlap && round<100) //100 rounds max
        {
            new_center = random_point(0+cluster_radius[i], 100-cluster_radius[i]);
            if(i>0)
            {
                boolean all=true;
                for(int j=0; j<i; j++)
                {
                    //distance to other centers has to be larger than the sum over their radii
                    if(getDistance_old(new_center, cluster_center[j])<cluster_radius[i]+cluster_radius[j])
                    {
                        all = false;
                    }
                }
                if(all) overlap=false;
                round++;
            }
            else{overlap=false;}
        }
        cluster_center[i]=new_center;
    }
}
```

- 2 things to take care of:
- Distance centers to the space borders
 - Avoid overlap of clusters



if(getDistance_old(new_center, cluster_center[j])<cluster_radius[i]+cluster_radius[j])



```
*/
public void generate_data()
{
    generate_cluster_centers(); // cf. last Exercise!
    int objectID=0;
    for(int i=0; i<clusters; i++)
    {
        // hyperCUBE approximation of cluster
        double[] lower = new double[dims];
        double[] upper = new double[dims];
        for(int k=0; k<dims; k++)
        {
            // bound random number in each relevant dimension by the radius
            if(real_cluster_dims.get(i).contains(k)) //if relevant then objects inside radius
            {
                lower[k] = cluster_center[i][k]-cluster_radius[i];
                upper[k] = cluster_center[i][k]+cluster_radius[i];
            }
            else //if irrelevant then objects randomly in whole dataspace
            {
                lower[k] = 0;
                upper[k] = 100;
            }
        }

        for(int j=0; j<cluster_size[i]; j++)
        {
            double[] point = new double[dims];
            boolean validPoint=false;
            while(!validPoint)
            {
                // generate point randomly located in hyperCUBE
            }
        }
    }
}
```



```
for(int j=0; j<cluster_size[i]; j++)
{
    double[] point = new double[dims];
    boolean validPoint=false;
    while(!validPoint)
    {
        // generate point randomly located in hyperCUBE
        point = random_point_radius(lower, upper);
        //new distance estimation based on relevant dimensions
        if(getDistance(i, point, cluster_center[i])<cluster_radius[i])
        {
            // point is also located in hyperSPHERE
            validPoint=true;
        }
    }
    data[objectID]=point;
    objectID++;
}
}
```

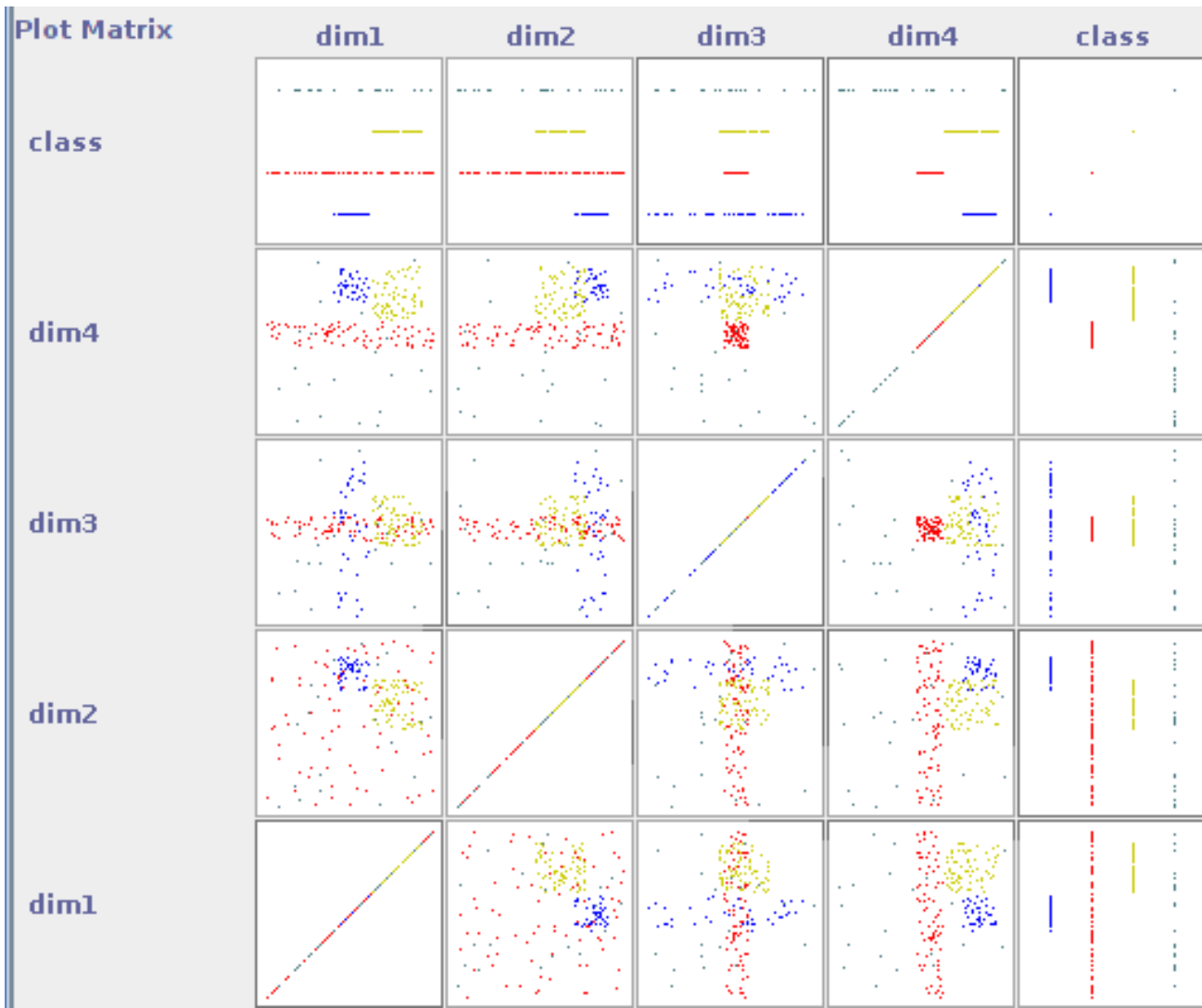
Main Idea:

Generate points randomly in hypercube of side length $2 \cdot \text{radius}$ centered at the center and then check whether it lies also in the sphere.

→ Bad idea for high-dimensionalities



```
/**
 * distributes noise randomly
 */
public void generate_noise()
{
    for(int i=0; i<noise; i++)
    {
        int round = 0;
        boolean inCluster=true;
        double[] point = new double[ dims ];
        while(inCluster && round<100) //100 rounds max
        {
            inCluster=false;
            point = random_point(0, 100);
            for(int j=0; j<clusters; j++)
            {
                //we do not want the noise to lie inside of clusters
                if(getDistance(point, cluster_center[j])<cluster_radius[j])
                {
                    inCluster=true;
                }
                if(inCluster) round++;
            }
        }
        data[objs-1-i] = point;
    }
}
```

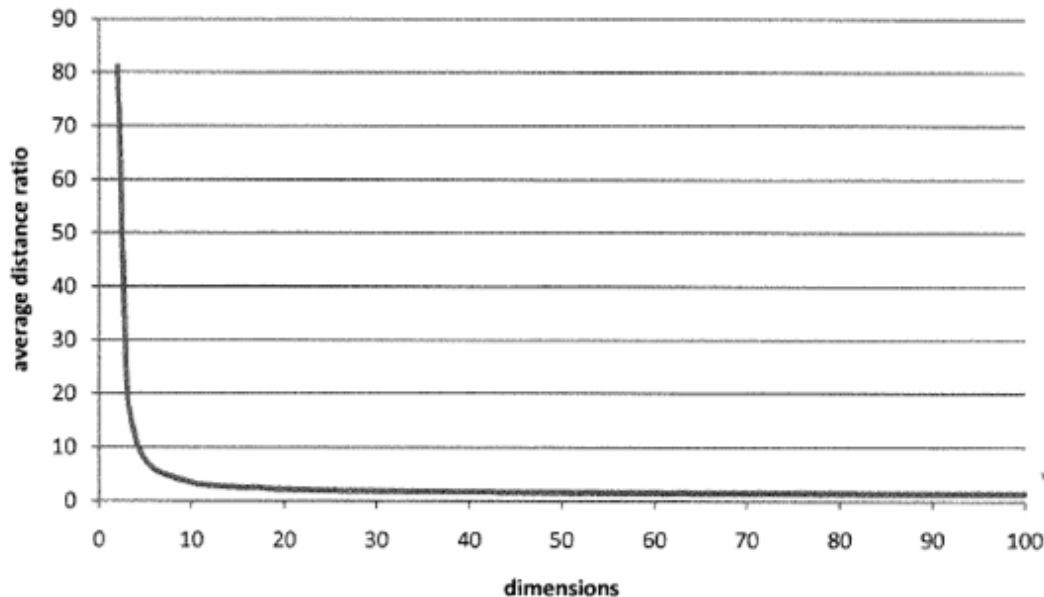


Task 1.3) Analysis of high dimensional data

8 points

- a) With the data generator of Task 3.1 create a sequence of datasets with increasing dimensionality D . Use the following parameter settings “ D 2 100 100 20 20 2 2 50 sequenceD.arff” for generating the data and vary D from 2 to 100 (step size 1).

Determine for each object the ratio “farthest-neighbor-distance”/“nearest-neighbor-distance” by using the Euclidean distance and calculate the average ratio for all objects (of the same dataset). Plot the average ratio for the sequence of datasets with increasing dimensionality. What conclusions can be drawn from this result with respect to the empty space problem/curse of dimensionality? Do you get the same results by using the Manhattan-Distance or the Maximum-Metric instead of the Euclidean distance?

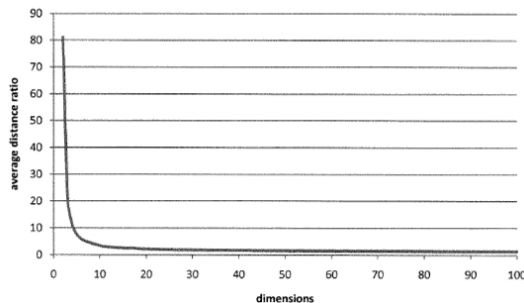


Task 1.3) Analysis of high dimensional data

8 points

- a) With the data generator of Task 3.1 create a sequence of datasets with increasing dimensionality D . Use the following parameter settings “ D 2 100 100 20 20 2 2 50 sequenceD.arff” for generating the data and vary D from 2 to 100 (step size 1).

Determine for each object the ratio “farthest-neighbor-distance”/“nearest-neighbor-distance” by using the Euclidean distance and calculate the average ratio for all objects (of the same dataset). Plot the average ratio for the sequence of datasets with increasing dimensionality. What conclusions can be drawn from this result with respect to the empty space problem/curse of dimensionality? Do you get the same results by using the Manhattan-Distance or the Maximum-Metric instead of the Euclidean distance?



Results are similar for the three different distances!

Observation: distance ratio decreases with growing dimensionality

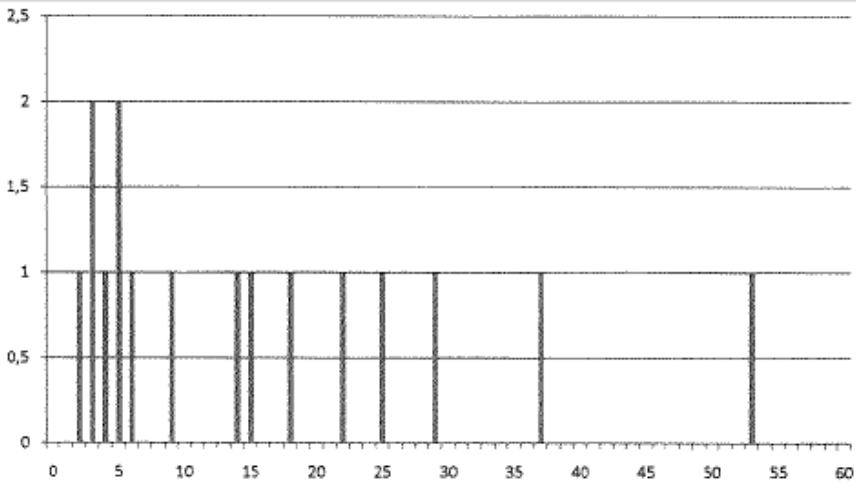
→ Distances grow more and more alike

→ Distance functions are less expressive in HD spaces

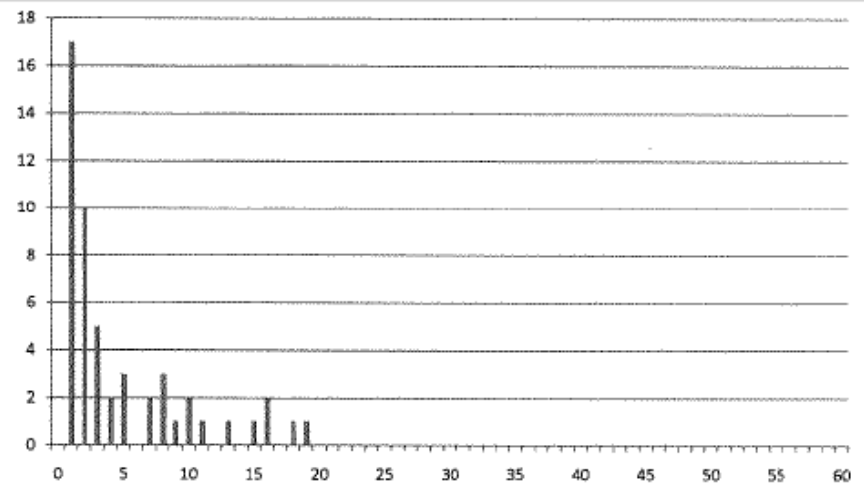
→ Clustering algorithms or general DM techniques based on distance functions will not work well (for discriminating between close and far away objects)



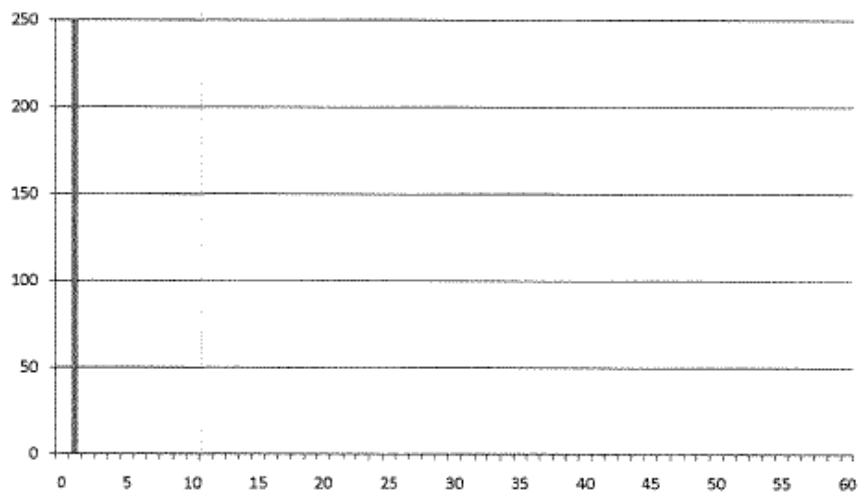
- b) Use the same sequence of datasets as in the previous task. Let us assume the data space is partitioned into a regular grid (cf. Slide 22, Chapter 1.2) with 4 partitions per dimension. Generate for each data set a histogram (bar chart) that counts the number of cells covering 1 object, 2 objects, 3 objects, ..., 250 objects. How do the histograms change by increasing the dimensionality of the data? What are your observations? Plot exemplarily the histograms for the dimensions $D=2,3,4,5,10,25,50,100$.



(a) $D = 2$



(b) $D = 3$



c) Let U_d be a d -dimensional hypersphere with the radius 1 and the volume Vol_d . Determine the radius r_d of the d -dimensional hypersphere X_d that covers the doubled volume. Provide a closed-form expression for r_d , give the limit of the function for $d \rightarrow \infty$, and plot the values of r_d in the range $d \in [1 \dots 50]$.

What conclusions can be drawn from these results with respect to the empty space problem/course of dimensionality?

First of all, the volume of a d -dimensional sphere with radius r is given by

$$\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)} r^d.$$

Then,

$$Vol_d = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2} + 1)},$$

and the equation for r_d is

$$\begin{aligned} \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2} + 1)} r_d^d &= 2 \cdot Vol_d \\ &= 2 \cdot \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2} + 1)}. \end{aligned}$$

Therefore,

$$r_d = \sqrt[d]{2}.$$

Then, clearly,

$$\lim_{d \rightarrow \infty} (r_d) = 1.$$

