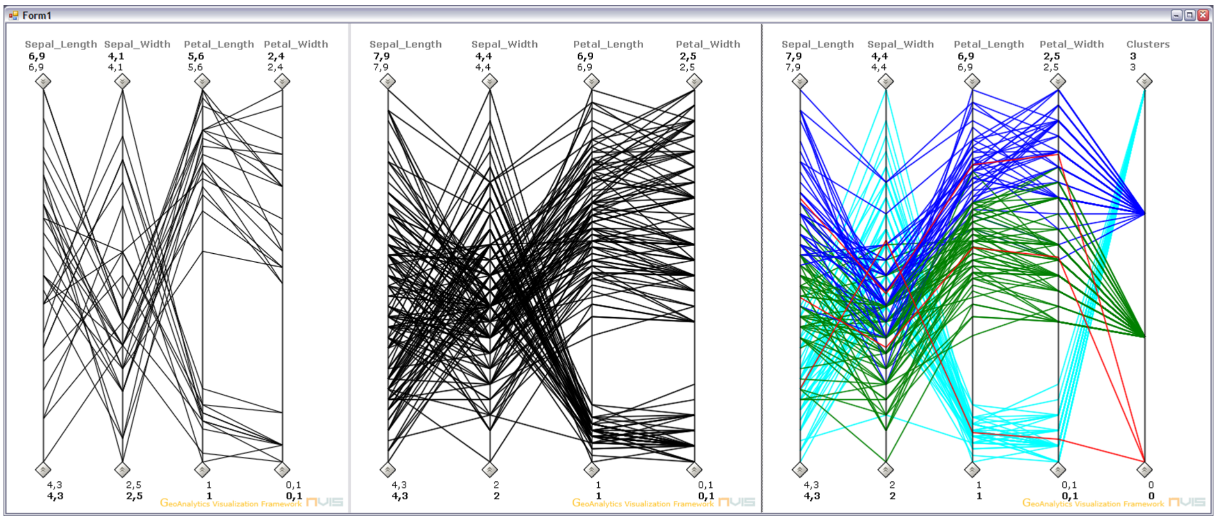
# TNM048 — Information Visualization
# 2. Visual Data Mining

January 23, 2007



## 1 Introduction

In this laboratory exercise you will use a combination of data mining algorithms and visualization techniques to analyse multivariate data sets. The data mining techniques that will be used are random downsampling and K-means clustering. The results of these techniques will be visualized using parallel coordinates.

The laboratory exercise should be performed individually or in a group of maximum two students. After completing all tasks, present your results to the laboratory assistant.

## 2 Getting Started

The data sets that will be used in this exercise are the cars data set (cars.xls) and the iris data set (iris.xls). In a multivariate data sets, a multivariate data item (or simply a data item) corresponds to a single row in the Excel spread sheet and a variable (or dimension) corresponds to a column.

**Task 1:**
Open the Excel files and familiarize yourself with the data sets.

In this lab a project is already created containing the following files:

- Form1.cs - The main file in the application.
- DownsamplingFilter.cs - The downsampling filter should be implemented in this file.
- KMeansFilter.cs - The K-Means filter should be implemented in this file.

The following classes will be used during this lab.

- LabExcelReader.ExcelReader
- Gav.Data.DataCube
- Gav.Data.Filter
- Gav.Data.ColorMap
- Gav.Data.SimpleColorMapPart
- Gav.Graphics.ParallelCoordinatesPlot

## 3 GAV Filter

To create a filter in GAV the abstract Filter class must be inherited. Inheriting this class forces the sub class to implement the *ProcessData()* method. The inherited functionality from the Filter class calls this method when the data needs to be processed. This can for example happen if the input to the filter has changed. In the ProcessData method the filter input is available through the *_input* variable. The following code references the input data array.

```
DataCube cube = _input.GetData();

float[, ,] inputData = cube.Data;
```

or simply

```
float[, ,] inputData = _input.GetData().Data;
```

The *GetData()* method returns a *DataCube*. The *DataCube* contains the data array which can be accessed through the *DataCube.Data* property. The output from the Filter is also a three dimensional array contained in a DataCube. The following code copies the input data to the output data of a filter.

```
protected override void ProcessData() {

    float[, ,] inputData = _input.GetData().Data;

    int sizeX = inputData.GetLength(0);

    int sizeY = inputData.GetLength(1);

    float[, ,] outputData = new float[sizeX, sizeY, 1];

    for (int i = 0; i < sizeX; i++) {
        for (int j = 0; j < sizeY; j++) {

            outputData[i, j, 0] = inputData[i, j, 0];

        }
    }

    _dataCube.Data = outputData;

}
```

## 4 Data Flow

In this lab we use three parallel coordinates components to visualize the data. One for the original data, one for the downsampled data and one for the K-Means clustered data. The visualization of the original data is used for comparison with the filtered data. Use the data flow model in Figure 1 when connecting the different components.

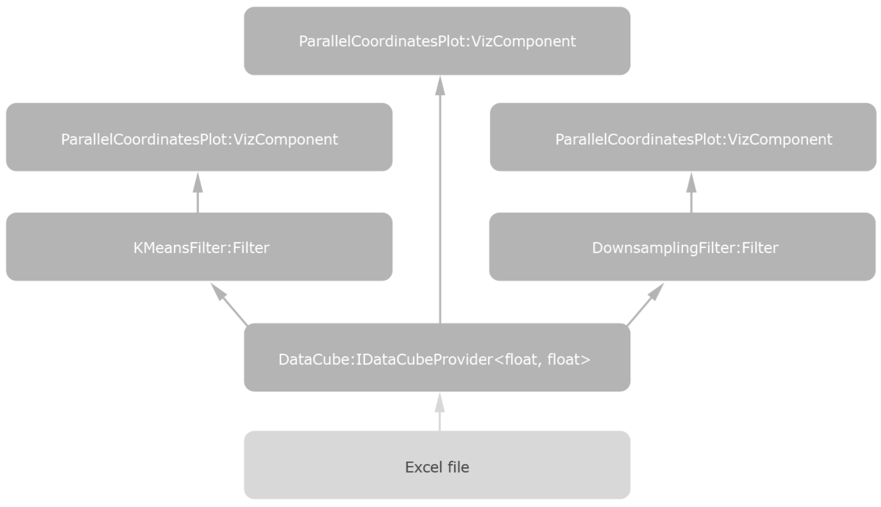

ParallelCoordinatesPlot:VizComponent
ParallelCoordinatesPlot:VizComponent
ParallelCoordinatesPlot:VizComponent
KMeansFilter:Filter
DownsamplingFilter:Filter
DataCube:IDataCubeProvider<float, float>
Excel file

Figure 1: The data flow model used in this lab.

## 5 Downsampling

A downsampling is performed by taking $M$ data items from an original data set containing $N$ data items ($M < N$). To preserve structures in the data, the $M$ data items are often taken randomly.

**Task 2:**
Implement a downsampling filter in the file DownsamplingFilter.cs that can take an arbitrary number of random data items from a data set.

**Task 3:**
Visualize the downsampled data sets in parallel coordinates. Compare different downsamplings with the original data sets and find out the minimum number of data items, $M$, needed in order to give a good representation of the original data sets.

## 6 K-means Clustering

The $K$-means algorithm is one of the most popular clustering algorithms and it is used in a wide range of applications. It is indented for data sets where all data items are of the quantitative type and the squared Euclidean distance

$$d(x, y) = \|x - y\|^2 \tag{1}$$

is used as the dissimilarity measure.
The $K$-means algorithm minimizes the criterion

$$\sum_{i=1}^{K} \sum_{C(i)=k} \|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2 \tag{2}$$

by assigning the $N$ data items in the data set to the $K$ clusters in such a way that the within-cluster variances, as defined by the average dissimilarity of the data items from the cluster means, are minimized. This is performed in two steps: (1) the criterion is minimized with respect to the cluster means, $\{\boldsymbol{m}_i, \ldots, \boldsymbol{m}_K\}$. (2) the criterion is minimized with respect to the cluster assignment, $C$. These two steps are described in detail below.

1. For a given cluster assignment, $C$, (the initial cluster assignment should be randomized) the total cluster variance is minimized according to equation 2. This is achieved by calculating the mean values of the assigned clusters.

2. Given a set of means, $\{\boldsymbol{m}_i, \ldots, \boldsymbol{m}_K\}$, equation 2 is minimized by assigning each data item to the closest cluster mean. That is

$$C(i) = \underset{1 \leq i \leq K}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2 \tag{3}$$

3. Steps 1 and 2 are iterated until the cluster assignments do not change.

**Task 4:**
Implement the $K$-means clustering algorithm in the file KMeansFilter.cs. The algorithm should support an arbitrary number of clusters.

To visualize the cluster means add $K$ number of rows to the output data array where $K$ is the number of clusters. Insert the cluster means in these rows. Add an extra column to the array. Use this column to store the cluster assignment for each data item (row). For the cluster means, insert the value -1.

To colour the data items, use a ColorMap with $\Psi$ number of parts, where $\Psi = K + 1$. Use the *SetEdgeValues* method on the ColorMap to set the correct edges. *Tip:* Use the SimpleColorMapPart.

**Task 5:**
Visualize the cluster means together with the original data sets. Use one colour for the cluster means and colour the data items according to cluster assignments.

**Task 6:** (optional)
To speed up the analysis, implement a button to re-calculate the clusters and a text field to specify the number of clusters to be used.

**Task 7:** (optional)
Visualize the clustered data in a 3D scatter plot with the previously used colour map.

**Task 8:**
Find the most appropriate number of clusters for each of the two data sets?