RWTH Aachen University
Chair of Computer Science VI
Prof. Dr.-Ing. Hermann Ney

Seminar Data Mining in WS 2003 / 2004

Preprocessing and Visualization

Jonathan Diehl

Matrikelnummer 235087

January 19, 2004

Tutor: Daniel Keysers

Contents

1 Introduction

Imagine you are a manager of a major electronics corporation which ships its products worldwide. At the end of the year, it is your assignment to analyze the sales of the current year and set up a business plan for the next year. You most likely will be faced with huge amounts of sales data of your own products, as well as general market information. It gets even worse because some of the data might be corrupt or in strange unit systems, making it hard to understand any correlations. Since your time is limited, and every false decision might cost your company a fortune, you depend on good performance as well as on good quality of the analysis.

Lets assume you have successfully completed your task of creating the report and come up with a plan, you regard as suitable for the current situation. In order to present your plans to the executive board of the corporation, you need human readable means of displaying your data. Otherwise you might not be able to convince them at all, and therefore all your research has been useless.

Data preprocessing is the process of manipulating data prior to actual mining steps. It aims at improving the quality or the performance of data mining algorithms. Data preprocessing can be divided into several categories: Data cleaning, integration, transformation, reduction and compression. I will introduce several techniques which aid this process under these different goals.

Data visualization on the other hand provides methods to efficiently display data to the human mind. Visualization is mainly used in the context of data exploration, which is the combined human and computer analysis of data. I will introduce plotting techniques for different kinds of data, like single- and multidimensional data.

Beforehand I will give a quick summary of the theoretical background and conventions used later in this document.

2 Theoretical Fundamentals

This section will cover some aspects of linear algebra and statistics, which are necessary for some of the methods introduced in later sections. For additional introductory literature about linear algebra I recommend [4]. An excellent introduction to statistics can be found in [8].

2.1 Data Representation

Data always consists of a number (N) of *samples* and a number (M) of *attributes* or *variables*. Here $x_{n,m}$ represents the value of the n-th sample and the m-th attribute. The data set itself can be interpreted as a set $X := (X_1, \ldots, X_N)^{\mathrm{T}}$ of vectors $X_n := (x_{n,1}, \ldots, x_{n,M})$, with each vector containing all attribute values of one sample. By merging these vectors, we get a data matrix of size $N \times M$, having the data samples as rows and the attributes as columns.

Data classes are groups of samples, distinguished by at least one common attribute. The distinction can be a fixed value, a range or some other criterion like a cluster. Table 2.1 shows an example of value distinguished data classes.

Table 2.1: Example of Data Classes

gender	age	income
f	22	2013
m	24	1692
m	24	408
f	19	408
f	31	1792
m	28	4087
m	22	253

Data classes of above data for fixed variable gender=f:
$\{(f, 22, 2013), (f, 19, 408), (f, 31, 1792)\}$

2.2 Matrix Arithmetics

Every matrix A of size $N \times N$ can be seen as a function which maps an N-dimensional vector space onto itself. *Eigenvectors* are those vectors which can be mapped under A onto a multiple λ, called *eigenvalue*, of themselves.

$$A \cdot x = \lambda x \tag{2.1}$$

where $x \in \mathbb{C}^{N \times 1}$ is an eigenvector of A and λ is the respective eigenvalue. Eigenvectors can be calculated using the characteristic equation:

$$\det(A - \lambda I) = 0 \quad \text{where } I \text{ is the identity matrix} \tag{2.2}$$

Every symmetric real-valued matrix of size $N \times N$ has N real-valued eigenvectors.

2.3 Measures of Location

Measures of location tell us about the rough location of the data set in the universe. The most common measure of location is the *sample mean* which is defined as

$$\mu = \frac{1}{N} \sum_{n=1}^{N} X_n \tag{2.3}$$

The sample mean has the property that it is central, thus minimizing the sum of squared differences between it and the data values.

Another measure of location is the *median*, which is the value which has for each dimension an equal number of data points above and below it. The median is more resistant to outliers than the sample mean, meaning it will not change as much if few values are far off the average.

Similarly the *quartiles* represent different parts of the distribution of the data. The first quartile is the smallest value which is greater than one fourth of the data points. Likewise the third quartile is greater than three fourth of the data points.

The *mode* is the most common sample value in the data set. If there is more than one mode the data is called *multi-modal*.

2.4 Measures of Dispersion

Measures of dispersion can provide useful information about the shape of our data. They describe how far the data is spread over the universe. The most common used measure is the *variance* which is defined as

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} \left(X_n - \mu\right)^2 \tag{2.4}$$

The *standard deviation* is simply the square root of the variance.

The *range* of a data set is the difference between the largest and the smallest data point. For a multi-dimensional vector space the range is determined for each dimension separately. Likewise the *interquartile range* is the difference between the first and the third quartile.

2.5 Measures of Correlation

To compare two sets of data we can use *correlation analysis*. First we need to calculate the *covariance* of both sets (for this case of two attributes) by

$$Cov(i,j) = \frac{1}{N} \cdot \sum_{n=1}^{N} (x_{n,i} - \mu_i) \cdot (x_{n,j} - \mu_j) \tag{2.5}$$

with μ_m the mean of the m-th attribute.

For more than two attributes we can summarize all covariance information in the so-called *covariance matrix*:

$$\Sigma = \frac{1}{N} \cdot X^{\mathrm{T}} X \quad \text{with } X \text{ zero-centered data matrix} \tag{2.6}$$

Zero-centered means that all columns of the matrix add up to zero. This can be achieved by subtracting the sample mean of all the values.

In order to be able to analyze the correlation we sometimes need to normalize the covariance. This can be achieved by calculating the correlation coefficient as

$$\rho_{i,j} = \frac{Cov(i,j)}{\sigma_i \cdot \sigma_j} \tag{2.7}$$

with σ_m the standard deviation of the m-th attribute.

If the correlation coefficient is greater than 0 the attributes are positively correlated, meaning if the value of one attribute increases, the value of the other attribute also increases. The inverse is true if it is less than 0. The higher the absolute value is, the more the two attributes correlate. Maximum correlation is reached by a correlation coefficient of 1.

2.6 Modeling Data

The simplest way to create a model for a set of data is to have the model fit a straight line. This method is called *linear regression*.

$$\mathrm{f}(x) = a + b \cdot x \qquad \text{with } a, b \text{ coefficients} \tag{2.8}$$

For the attributes i and j of a data set X of size N the coefficients can be calculated by

$$b = \frac{Cov(i,j)}{\sigma_i^2} \tag{2.9}$$

$$a = \mu_j - b \cdot \mu_i \tag{2.10}$$

with σ_m^2 the variance and μ_m the mean of the m-th attribute.

Multiple linear regression is an extension of linear regression allowing us to model our data set as a linear function of multiple variables.

$$\mathrm{f}(x_1, \ldots, x_K) = a + \sum_{k=1}^{K} b_k \cdot x_k \qquad \text{with } a, b_k \text{ coefficients} \tag{2.11}$$

3 Visualization

In everyday life, we have to find patterns and structures all the time in the most complex of all data sets, the world. This is the motivation of data visualization, to display data in a way which utilizes the human's natural abilities to analyze complex structures. This approach obviously is very different from formal mathematical methods and although it is limited in application especially with large sets of data, it still plays an important role in data mining. Data exploration combines visualization with interaction by providing multiple, user-defined views of the data.

More information about data visualization can be found in [6].

3.1 One- or Two-Dimensional Plots

Visualizing low-dimensional data is in most cases quite intuitive. We simply display our data on a two-dimensional plane representing variables or other indicators with the axes. We also can map two-dimensional models of our data onto the same plot and therefore compare the model to the actual data points. Since low-dimensional plots are easy to read and understand they are widely used for the most versatile occasions. I will introduce the most important plot types here and then move on to more complicated plots which attempt to visualize multidimensional data.

3.1.1 Histograms and Kernel Method

The most common display of univariate data is also a very basic one. The *histogram* shows the number of values of the data set that lie in consecutive intervals. Figure 3.1 shows such a histogram over a discrete set of data. Histograms may be misleading for small sets of data. Small changes of the interval can result in major fluctuations of the diagram. However with large data sets these effects diminish.

These undesirable impacts can be encountered by *smoothing estimates* like the *kernel estimate*. This method smoothes the data by adapting each data point toward its immediate neighbors. It does so by fitting the kernel function onto every point of the histogram and combining all the functions by simply adding them up. This new density over the kernel function K with bandwidth h can be calculated as

$$f(x) = \frac{1}{N}\sum_{n=1}^{N} K\left(\frac{x - x_{m,n}}{h}\right) \qquad \text{for given data set } X \text{ and } m\text{-th attribute} \tag{3.1}$$

where $\int K(t)dt = 1$ is enforced to preserve the condition of a proper density function. Usually a smooth unimodal function peaking at zero like the Gaussian curve is chosen for K. The quality of this estimate depends less on K than on the value h: The greater h is the smoother the resulting curve will be. Figure 3.2 shows a kernel estimate of the previous histogram.

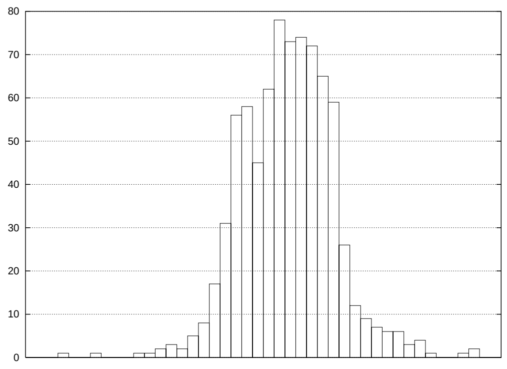


Figure 3.1: Histogram

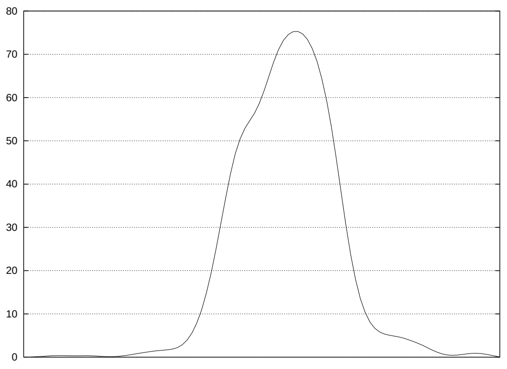


Figure 3.2: Kernel Function Estimate

3.1.2 Box Plots

The *box plot* is a tool to display summarized data. Basically any combination of measures can be displayed. Boxes are used to represent measures of dispersion, such as the interquartile range, and lines to indicate measures of location, such as the sample mean. It is also possible to arrange multiple indicators horizontally to compare different variables or to show the course of one variable over time. Figure 3.3 shows an example of a box plot.

3.1.3 Scatter Plots

Scatter plots visualize relationships between two variables. Each data point is simply mapped onto a two-dimensional plane where each axis represents one dimension of the

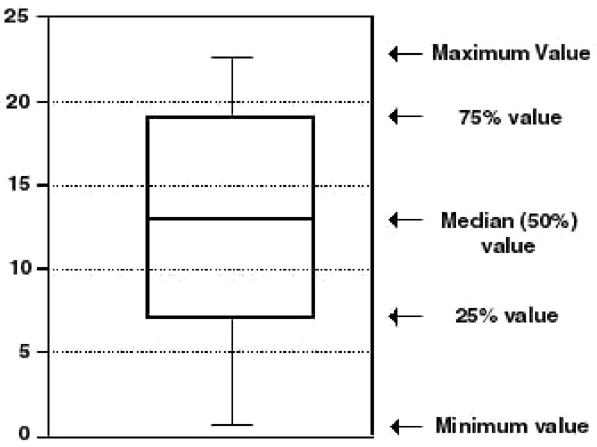


Figure 3.3: Box Plot from [3]

data. Using this tool we can easily determine whether two variables are correlated and how many outliers there are. Scatter plots can also be used to show the progression of a variable over time. In Figure 3.4 a scatter plot of a random data set is shown.

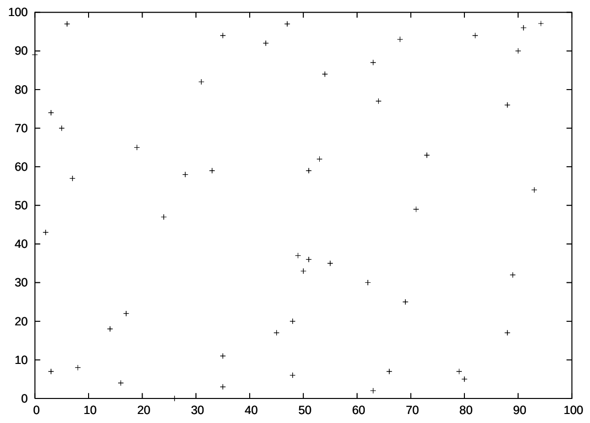


Figure 3.4: Scatter Plot

For large data sets or small discrete scales, the scatter plot becomes useless though, since much information is simply overprinted, meaning that multiple data points are printed at the same point in space. At the worst what remains of the scatter plot is a purely black rectangle. To counter this effect we need a tool which respects multiplicity of data points.

3.1.4 Contour Plots

Contour Plots work similar to scatter plots, although they do not display each data point separately but rather boundaries of accumulation classes of data. For this to work we need to construct a two-dimensional density estimate, for instance by using a generalization of the kernel method. In a way the contour plot is like a geographical map of our data indicating different heights of the two-dimensional density. Instead of contour lines color can be used just as well to distinguish between the different levels of accumulation. Figure 3.5 shows such a plot.

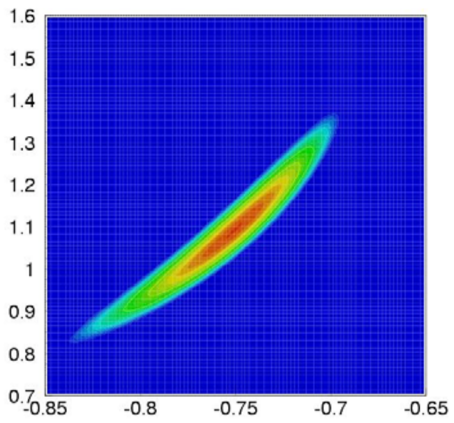


Figure 3.5: Contour Plot from [2]

3.2 Multi-Dimensional Plots

Since the output devices we use like paper or computer screens are flat, they cannot display multidimensional data efficiently. Therefore we need to project the data onto a two-dimensional plane in order to visualize it. Here I will introduce two methods which simply display the data from different points of view in multiple low-dimensional plots. In the next section I will then introduce methods to mathematically reduce high-dimensional data onto a low-dimensional coordinate system, which then can be plotted with any method described before.

3.2.1 Trellis Plots

Trellis plots use multiple bivariate plots to visualize multi-dimensional data. A particular pair of variables is fixed to be displayed while relationships to other variables are shown by producing a series of plots, conditioning the different levels of the classifying variables. Figure 3.6 shows an example of a trellis plot where the data is split into six classes according to the two classifying variables. The name trellis plot comes from the arrangement of the plots which looks like a trellis (grid).

3.2.2 Parallel Coordinates Plots

A parallel coordinates plot consists of one y-axis and multiple x-axes, each representing one variable. The data points are displayed as lines, connecting the measured values of each

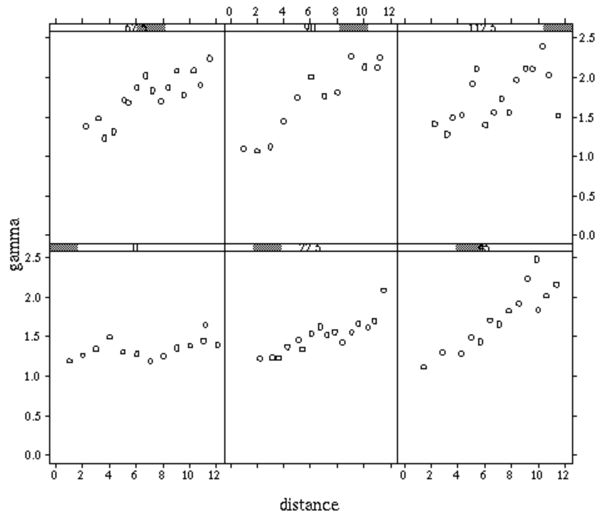


Figure 3.6: Trellis Plot from [5]

variable. Colored lines can be used to emphasize the distinction between the individual samples. Figure 3.7 shows a parallel coordinates plot of four variables.

There are more plot types which can show multi-dimensional data but their complexity quickly outweighs the purpose of visualization. Instead I will now concentrate on preprocessing steps to prepare the data adequately, either for visualization or further processing.

4 Preprocessing

Data preprocessing can greatly improve the quality of data mining results, no matter whether an algorithmic or a visual approach is considered. There are four different aspects of data preprocessing: Data cleaning, data integration, data transformation and data reduction. I will describe these aspects here and introduce the most common methods to realize each one of them.

For more information about data preprocessing refer to [7].

4.1 Data Cleaning

Data cleaning provides methods to cope with three common problems of real world data sets. The first problem is that data may be incomplete, meaning a considerable amount of values are missing. To solve this problem we have to look at means of how to best approximate the missing data. The second problem deals with errors in data values or so-called noisy data. In order to remove these errors there are several smoothing methods available. The third problem deals with data inconsistencies, which occur mostly when data is merged from different sources.

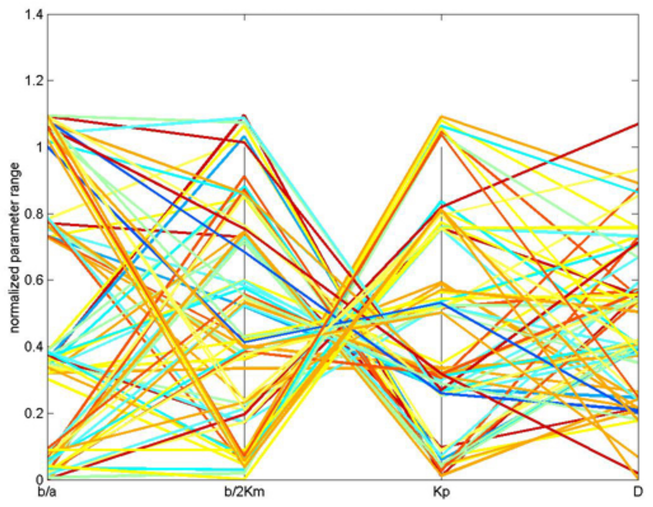


Figure 3.7: Parallel Coordinates Plot from [1]

4.1.1 Missing Values

In the real world collected data almost never is complete. Certain values might be missing for several reasons, most of all they were never recorded. Nevertheless many data mining algorithms require a complete set of data to run.

To solve this problem we obviously can just ignore all tuples with missing values. This method is not very efficient though because large amounts of valuable data might be thrown away.

A better way to approach the problem is to use statistical measures, as discussed in section 2. The most obvious measure to fill in for missing values is the attribute mean. If known classifications within the data set exist and we wish to include them in our analysis, we can improve our results further by calculating the mean values over each class separately.

A promising and hence very popular strategy is to derive the value from its correlation to other attributes. This is done by constructing a model of the attribute and its relationship to other attributes. One way to achieve this is by using regression, as introduced in section 2.6. By constructing a model of the attribute we can efficiently approximate any missing values. Another way of approximation is to construct a decision tree based on other attributes. Since decision trees are used in data mining algorithms themselves, I will leave this topic for the following papers to cover. Either method has a great chance to preserve any relationship between attributes.

4.1.2 Noisy Data

Similarly to the problem of missing values, real world data sets often have random error within their values. To remove these errors we smoothen the data. I will introduce two different techniques to cope with erroneous data.

Binning is a local smoothing method which means it consults only the immediate neighborhood of each data point for the smoothing operation. Therefore binning requires the data set to be ordinal and sorted. At first the data is partitioned into a number of bins, which make up the neighborhood of the data points inside. Bins may be equiwidth, equidistributed or just arbitrary, depending on the application. Then all points are smoothed according to the neighborhood or bin they are located in. In the end all bins are again merged, resulting in a smoothed data set. The operation used for smoothing the bin values may vary:

- *Smoothing by means* replaces all individual data points with the bin's mean value.
- *Smoothing by medians* is similar but replaces the data points with the bin's median.
- *Smoothing by bin boundaries* first identifies the boundary of the bin as its range and then replaces each data point with the closest boundary value.

The intensity of the smoothing relies mostly not on the smoothing operation but on the width of the bins. In general, the larger the width, the greater the effect of the smoothing will be. Table 4.1 shows an example of the partitioning and two operations. Binning can also be used as a method of data discretization.

Table 4.1: Example of Binning

	partitioning	smoothing by means	smoothing by bin boundaries
bin 1	2, 4, 5, 6, 9, 10	6, 6, 6, 6, 6, 6	2, 2, 2, 2, 10, 10
bin 2	12, 16, 17, 19	16, 16, 16, 16	12, 19, 19, 19
bin 3	23, 26, 27, 28	26, 26, 26, 26	23, 28, 28, 28
bin 4	31, 33, 35	33, 33, 33	31, 31, 35

Regression is a global method of smoothing where we attempt to find a function which best fits the data set as a whole. Representing data as a mathematical equation helps smoothing out noise and also greatly reduces the amount of data to be stored. I have already introduced some regression models in section 2.6. Figure 4.1 shows a linear regression model of sample data.

The drawback of this method is that it is not always possible to compute a precise regression model of the given data. Therefore the model has to be verified that it accurately represents the data. This verification can be done by correlation analysis or by visualizing and comparing the model to the data itself.

4.2 Data Integration

The problem of *data integration* occurs whenever the target data is extracted from multiple data stores. Here three main problems arise: The data must be merged, redundancies must be removed and value conflicts must be resolved.

To merge data from different sources equivalent attributes must be matched up. This problem is called the *entity identification problem.* To help solve this problem some databases

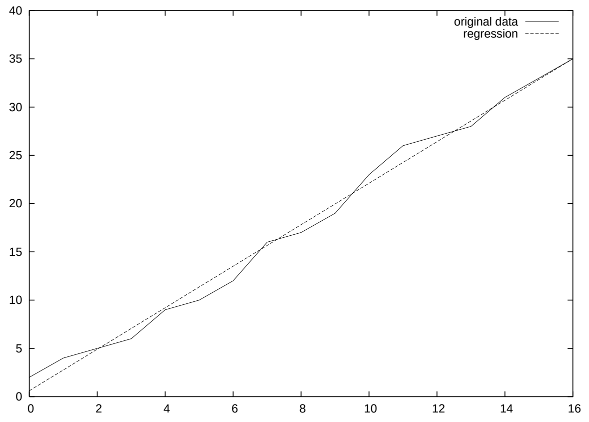


Figure 4.1: Linear Regression

carry meta data, which is data about the data. By using and comparing this meta data attributes can be matched up and errors in integration avoided.

Data redundancy occurs if attributes can be derived from the data itself. Redundancy causes unnecessarily large sets of data and therefore can slow down the data mining process. To detect redundancies we can use correlation analysis as introduced in section 2.6. If any pair of attributes correlate we can simply drop one of these attributes.

Value conflicts may be the result of differences in representation, scaling or encoding of multiple data sources. For instance one source has a weight attribute stored in metric units while another source has the same attribute stored in British imperial units. Data transformation, as discussed in the next section, provides some methods to cope with this problem.

4.3 Data Transformation

For data mining algorithms to work efficiently the input data often has to be in a certain format. In *data transformation* the data is transformed to meet these requirements.

The most important technique in data transformation is *normalization* which rescales data values to fit into a specified range such as 0.0 to 1.0. Normalization can speed up the learning phase of certain data mining algorithms and prevent attributes with large ranges from outweighing attributes with small ranges. On the other hand normalization can alter the data quite a bit and should not be used carelessly. I will introduce two commonly used normalization methods.

Min-max normalization simply transforms the data linearly to the new range as

$$x'_{n,m} = \frac{x_{n,m} - min_m}{max_m - min_m} \cdot (max'_m - min'_m) + min'_m$$

with $[min_m, max_m]$ the original and $[min'_m, max'_m]$ the target range of the m-th attribute of the data. Min-max normalization will encounter an error if future input falls outside the original data range of the attribute.

Z-score normalization makes use of the arithmetic mean μ_m and standard deviation σ_m to normalize the data.

$$x'_{n,m} = \frac{x_{n,m} - \mu_m}{\sigma_m}$$

The advantage of this method is that we do not need to provide the range of the attribute and therefore ensure that all future data will be accepted.

Another data transformation technique is *attribute construction* where new attributes are constructed from given data. These new attributes usually represent operations on the original attributes like the sum or the product. Constructed attributes provide fast access to summarized data and therefore help speed up the actual mining process.

4.4 Data Reduction

Data mining on huge databases is likely to take a very long time, making the analysis practically impossible. By applying data reduction techniques to the data we obtain a much smaller representation of the original data, while maintaining its integrity and therefore the analytical results we will receive. In this section, I will introduce five different approaches to data reduction.

4.4.1 Data Cube Aggregation

Imagine the electronics corporation introduced in the beginning of this paper. Say we have gathered information of all the sales of different item types and different branches over the last few years. We now wish to organize our data in a way which allows fast access to summaries of the data, like annual sales of a certain year. A *data cube* provides a method to store this kind of information efficiently.

To construct a data cube the following steps are executed:

1. Choose the attribute to be displayed in value. For our example we would choose sales amount.

2. Choose a number of attributes according to which the data should be aggregated. These will be the dimensions of the cube. For us this will be time, item type and branch.

3. Choose the level of abstraction for the aggregation and determine data classes accordingly. In our example the time attribute is split into years $(1997, 1998, 1999)$. The other attributes are split similarly into classes.

4. For all possible combination of classes of the dimensions calculate the aggregated value of the displayed attribute. Figure 4.2 shows a fully constructed data cube of our example.

For each level of abstraction a separate data cube can be constructed. The cube with the lowest possible level of abstraction is called the *base cuboid*, the one with the highest level

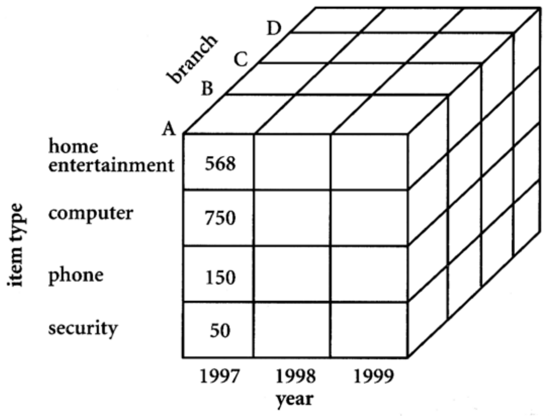


Figure 4.2: Data Cube over three dimensions from [7]

is called the *apex cuboid*. The higher the level of abstraction the more reduced the data set will be.

Data cubes provide efficient organization of summarized information which is the foundation of on-line analytical processing as used in data exploration.

4.4.2 Attribute Subset Selection

By removing attributes the data set will immediately shrink in size. *Attribute subset selection* aims at removing as many of the original attributes as possible, while maintaining its integrity with respect to a given class or concept. In other words, attribute subset selection tries to conserve all the information needed to classify the remaining data accurately.

To determine which attributes should be removed there are four methods mainly used:

1. *Stepwise forward selection* starts with an empty set of attributes and iteratively adds the remaining attributes to its set until a given threshold is met. An often used threshold is to meet a certain probability that the data is identified correctly using only the selected subset of attributes.

2. *Stepwise backward elimination* starts with all attributes and removes iteratively the worst of the set again until a threshold is met.

3. *Combination of forward selection and backward elimination* simply combines both methods by adding the best attribute to the set and removing the worst from the remaining.

4. *Decision tree induction* constructs a decision tree over the given attributes and derives the most significant ones from it.

To determine the significance of a specific attribute there are several measures available. Since this topic is discussed in detail when it comes to decision trees, I will only introduce one method called *information gain*.

The information gain of an attribute is a measure for its relevance in the classification process. Suppose in a data set of N samples there are K different classes which need to be identified. For $N_1 \dots N_K$, the number of samples of the the classes of a training set, the expected information needed for its classification can be calculated as

$$I(N_1, \dots, N_K) = -\sum_{k=1}^{K} \frac{N_k}{N} \cdot \log_2 \left(\frac{N_k}{N} \right) \tag{4.1}$$

This information measure is called the *entropy* of the data set. Now we want to know how the entropy changes if we partition the data according to the i-th attribute into data classes $C_1, \dots, C_L$. The entropy of the data after partitioning can be calculated as

$$E_i = \sum_{l=1}^{L} \frac{N_{1,l} + \dots + N_{K,l}}{N} \cdot I(N_{1,l}, \dots, N_{K,l}) \tag{4.2}$$

where $N_{k,l}$ is the number of samples from the partition C_l within the k-th class to be identified. From 4.1 and 4.2 the information gain can simply be calculated as

$$Gain_i = I(N_1, \dots, N_K) - E_i \tag{4.3}$$

The higher the gain is, the more significant the attribute will be.

4.4.3 Principal Components Analysis

Principal components analysis also reduces the dimensionality of the data set, but unlike attribute subset selection the data is projected onto a new set of variables, called *principal components.* These principal components are a linear combination of the original attributes. The method aims at maximizing the variability along the new coordinate system, thus conserving as much of the variance of the original data with as few dimensions as possible. Figure 4.3 shows both principal component axes of a two-dimensional ellipse-shaped data set.

It turns out that the principal component vectors are equal to the eigenvectors of the *scatter matrix* $S := X^{\mathrm{T}}X$ of a zero-centered data set X. To show this we first determine the variance (s^2) of the projected data and then maximize it.

$$\begin{aligned} s^2 &= \Big(Xa\Big)^{\mathrm{T}} \cdot \Big(Xa\Big) && (4.4) \\ &= a^{\mathrm{T}} X^{\mathrm{T}} X a && (4.5) \\ &= a^{\mathrm{T}} S a && (4.6) \end{aligned}$$

with a being the projection vector for the first principal component. Without enforcing some kind of restriction the projection vector could increase in size without limit, rendering the maximization problem unsolvable. Therefore we impose the normalization constraint $a^{\mathrm{T}}a = 1$ by subtracting it, multiplied by the so called Lagrange multiplier (λ), from our maximization problem.

$$\mathrm{F}(a) = a^{\mathrm{T}} S a - \lambda(a^{\mathrm{T}} a - 1) \tag{4.7}$$

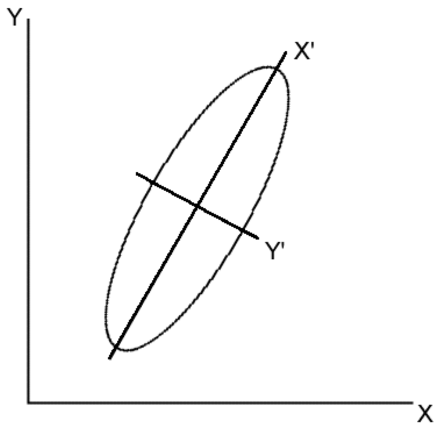


Figure 4.3: Principal Components Analysis from [9]

The derivation of this term leads to

$$\mathrm{F}'(a) = 2Sa - 2\lambda a = 0 \tag{4.8}$$

This can be reduced to the familiar eigenvalue form of

$$(S - \lambda I)a = 0 \tag{4.9}$$

with I being the identity matrix. Thus the first principal component, having the highest variance, is the eigenvector associated with the largest eigenvalue. Likeweise the remaining eigenvectors make up the rest of the principal components ordered by the size of the associated eigenvalues. By projecting the data onto the first K principal components with $K < M$ the original number of attributes, we reduce the data set in size while loosing only little of its variability. In practice a small number of principal components is sufficient to get a good presentation of the original data.

Principal components analysis can be used for visualization by projecting the multivariate data onto a two-dimensional plane, which then can easily be plotted.

4.4.4 Multidimensional Scaling

The principal components approach discussed in the previous section has its limits in applicability though. It requires the data to lie in a linear subspace of the original universe to function properly. So what if this subspace is not given, if the data is aligned as a multi-dimensional curve for instance? Principal components analysis might not to be able to detect the underlying structure of the data at all. To cope with situations like these I will introduce another method called *multidimensional scaling*. This method projects the original data onto a plane, preserving the distances between all data points as good as possible.

There are various methods of multidimensional scaling differing mainly in the way they define distances. I will concentrate on the most basic approach using Euclidean distances to map the data to a two-dimensional plane. The quality of the representation (Y) of the original data (X) is measured as the sum of squared differences between the original distances (δ) and the mapped distances (d):

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} (\delta_{i_j} - d_{i_j})^2 \tag{4.10}$$

To solve this minimization problem we will construct a matrix $B = YY^{\mathrm{T}}$ of size $N \times N$ from which we can factorize the optimal representation Y. Starting with the squared distances of Y we deduce

$$\begin{aligned} d_{i,j}^2 &= \|Y_i - Y_j\|^2 && (4.11) \\ &= Y_i^{\mathrm{T}} Y_i - 2Y_i^{\mathrm{T}} Y_j + Y_j^{\mathrm{T}} Y_j && (4.12) \\ &= b_{i,i} - 2b_{i,j} + b_{j,j} && (4.13) \end{aligned}$$

with $b_{i,j}$ the value of the i-th column and the j-th row of B. By inverting equation 4.13, we can compute the elements of B. If we then use the distances of our original data set X and factorize B using the two eigenvectors with the largest eigenvalues, we will obtain a two-dimensional representation of the data which satisfies 4.10.

The inversion requires the introduction of a constraint though since Euclidean distances are invariant to shifting and rotation of the data. We can use zero mean normalization assuming that $\sum_{n=1}^{N} x_{n,m} = 0$ for all $m = 1, \ldots, M$, which leads to $\sum_{i=1}^{N} b_{i,j} = \sum_{i=1}^{N} b_{j,i} = 0$ for all $j = 1, \ldots, N$. By summing 4.13 first over i, then over j and finally over i and j we obtain

$$\sum_{i=1}^{N} d_{i,j}^2 = \mathrm{tr}(B) + n \cdot b_{j,j} \tag{4.14}$$

$$\sum_{j=1}^{N} d_{i,j}^2 = \mathrm{tr}(B) + n \cdot b_{i,i} \tag{4.15}$$

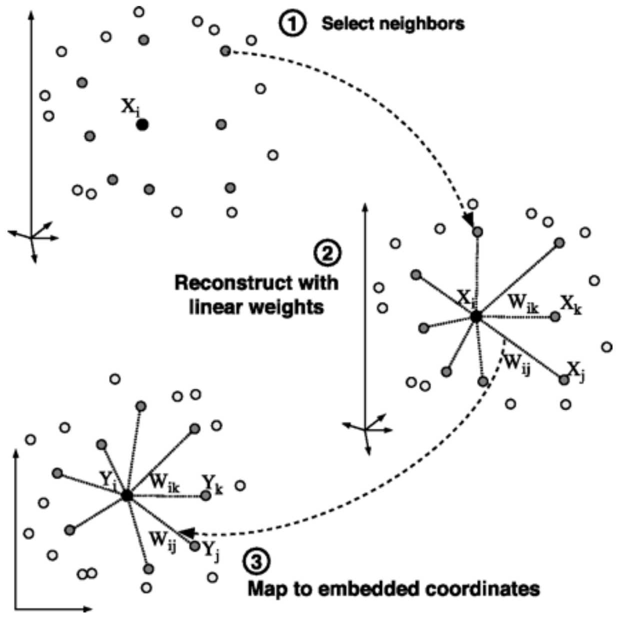
$$\sum_{i=1}^{N} \sum_{j=1}^{N} d_{i,j}^2 = 2n \cdot \mathrm{tr}(B) \tag{4.16}$$

with $\mathrm{tr}(B) = \sum_{n=1}^{N} b_{n,n}$ the trace of B. By fitting 4.16 into 4.14 and 4.15 we can express $b_{i,i}$ and $b_{j,j}$ in terms of $d_{i,j}$. Plugging these into 4.13 we can compute $b_{i,j}$ in terms of $d_{i,j}$ only, thus compute B.

This process is called *principal coordinates* method. It yields an optimal solution of the minimization problem 4.10 if B can arise as a product YY^{T}. There are other ways to obtain a likewise representation which do not depend on this requirement. For more information please refer to chapter three of [6].

4.4.5 Locally Linear Embedding

Experience shows that even the most complex data usually follows some low-dimensional, nonlinear manifold. For classification and comparison tasks it is sufficient to regard only this manifold instead of the whole data set. For instance all faces follow a similar shape, and only small details are sufficient for distinction between different faces. *Locally linear embedding* provides a method to determine such a manifold of the data by analyzing its local correlations. While multidimensional scaling, as introduced in the previous section, minimizes distance errors of the whole data set, locally linear embedding only considers local patches of the data. Figure 4.4 summarizes the algorithm.



① Select neighbors
X_i
② Reconstruct with linear weights
X_i
W_ik
X_k
W_ij
X_j
Y_i
W_ik
Y_k
W_ij
Y_j
③ Map to embedded coordinates

Figure 4.4: Locally Linear Embedding Algorithm from [10]

The quality of the embedding is measured using the squared differences between the data points and their projections:

$$\epsilon(W) = \sum_{n=1}^{N} \|X_n - \sum_{i=1}^{N} W_{n,i} \cdot X_i\|^2 \tag{4.17}$$

with W the projection weights matrix. To compute W, two constraints are in effect: Only neighbors of the data point are considered for the reconstruction, leaving $W_{n,i} = 0$ for all X_i not in the neighborhood of X_n and the weight matrix itself is normalized ($\sum_{i=1}^{N} W_{n,i} = 1$). Since we assume the data to follow some low-dimensional manifold, there must be a linear mapping which maps the high-dimensional data onto this manifold. This mapping must consist of rotations, rescalings and translations of the data points, which are, because of the constraints, intrinsic properties of our weight matrix. We therefore expect W to reconstruct an accurate representation of the data not only in the high-dimensional space but also in the low-dimensional space of the underlying manifold. Mathematically the algorithm performs as following:

Input: Data set X

1. For $n = 1, \ldots, N$:

```
    (a) For $i = 1, \ldots, N$: Compute distance between $X_n$ and $X_i$
    (b) Set $Neigh_n = \{X_i \mid X_i \text{ is one of the } K \text{ closest neighbors of } X_n\}$
2. For $n = 1, \ldots, N$:
    (a) Create matrix $Z_n$ consisting of all neighbors $X_i \in Neigh_n$
    (b) Subtract $X_n$ from every column in $Z_n$
    (c) Compute scatter matrix $S_n = Z_n^{\mathrm{T}} Z_n$
    (d) Solve $S_n \cdot w_n = 1$ for $w_n$
    (e) Set $W_{n,i} = \begin{cases} w_{n,i} & i \in Neigh_n \\ 0 & \text{otherwise} \end{cases}$
3. (a) Create sparse matrix $M = (I - W)^{\mathrm{T}} \cdot (I - W)$ with $I$ the identity matrix of size $N$
    (b) Find $L$ eigenvectors of $M$ corresponding to the smallest $L$ eigenvalues greater than zero
    (c) Create matrix $Y$ with $L$ eigenvectors as columns
```

Output: Embedded data set Y

Figure 4.4 shows how a three-dimensional swiss roll is transformed into a plane using this algorithm.

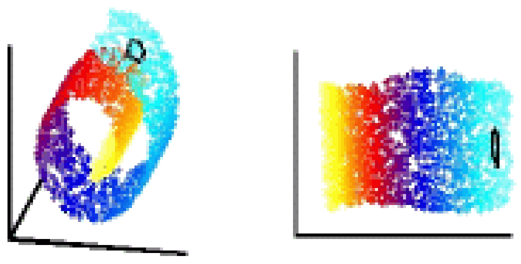


Figure 4.5: Locally Linear Embedding: Swiss Roll from [10]

5 Summary

Data visualization allows the human to read and interpret data. For this purpose various graphical plots are used for different scenarios. In general the more dimensions and the more data samples there are the more difficult the interpretation will be. For complex sets of data it is recommended to use data preprocessing steps prior to visualization.

Data preprocessing aims at preparing data for further processing steps or for visualization. There are several aspects of data preprocessing, each trying to cope with a different problem of real world data sets. Data cleaning provides methods to improve the coherence and quality of the data. Data integration deals with integrating data from different sources. Data transformation manipulates the data for uniformity. Data reduction aims at finding smaller representations of the data, which still provide the same information.

References

[1] Parallel coordinate graphics using matlab. Technical report, Cornell University, Department of Neurobiology and Behavior, 2001.

[2] Kaons at the tevatron, pi pi e e plots. Technical report, Fermi National Accelerator Laboratory, 2002.

[3] Monitoring and assessing water quality. Technical report, U.S. Environmental Protection Agency, 2003.

[4] A. Beutelspacher. *Lineare Algebra*. Vieweg Verlag, Wiesbaden, Germany, 2001.

[5] Iain Currie. First impressions of video conferencing. Technical report, CTI Statistics, 1997.

[6] H. Manila D. Hand and P. Smyth. *Principles of Data Mining*. MIT Press, Cambridge, MA, 2001.

[7] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, San Diego, CA, 2001.

[8] U. Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Vieweg Verlag, Wiesbaden, Germany, 2003.

[9] Brian A. Rock. An introduction to chemometrics - general areas. Technical report, ACS Rubber Division, 1997.

[10] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science Magazine*, 290(5500), Decembre 2000.