Feature space partitioning based on the quantification of redundancy and synergy using interaction information

Aufteilung des Merkmalsraums basierend auf der Bewertung von Redundanz und Synergie unter Verwendung der Interaktionsinformation
Diplomarbeit von Amadeus Kalb aus Frankfurt am Main
Tag der Einreichung:

1. Gutachten: Dr.-Ing. Lars Gräning
2. Gutachten: Dipl.-Ing. Florian Damerow
3. Gutachten: Prof. Dr.-Ing. Jürgen Adamy

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich 18
Elektro- und Informationstechnik
Institut für
Automatisierungstechnik
und Mechatronik
Fachgebiet Regelungsmethoden
und Robotik

Feature space partitioning based on the quantification of redundancy and synergy using interaction information
Aufteilung des Merkmalsraums basierend auf der Bewertung von Redundanz und Synergie unter Verwendung der Interaktionsinformation

Vorgelegte Diplomarbeit von Amadeus Kalb aus Frankfurt am Main

1. Gutachten: Dr.-Ing. Lars Gräning
2. Gutachten: Dipl.-Ing. Florian Damerow
3. Gutachten: Prof. Dr.-Ing. Jürgen Adamy

Tag der Einreichung:

Erklärung zur Diplomarbeit

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 13th January 2016

(Amadeus Kalb)

Abstract

Data mining, machine learning and artificial intelligence have become important domains in computer science because they provide solutions to analyze and model technical systems. Real systems can be modeled using data that contains example relations of the system variables (data samples). Today, huge datasets are available that can overcharge the construction and learning of current computational models, while tending to contain details that are irrelevant or redundant. Therefore, feature selection methods have become a common method to remove less important variables from the data. The removal of variables allows to construct simpler system models that are expected to have better generalization capabilities, while requiring a smaller number of data samples to learn the system behavior.
However, all feature selection methods aim on removing those variables completely that are **on average** mostly irrelevant. Hence, a subset of variables is kept and the rest is lost. Consequently, a variable that is only useful in rare situations, will be completely removed.
Our approach aims to avoid this disadvantage. In order to use the variable in spite of that, a more complicated model can be used for the rare situations, where the variable is relevant. In the rest of the situations the variable is ignored and the **benefits** of a simpler model can take effect.
Assuming that the relevance of a variable depends on the location in the feature space, methods were developed that partition the feature space based on the quantification of local *Synergy* and *Redundancy* using the *Multivariate Interaction Information*. Ideally, in one partition *Redundancy* is dominating, which allows a local feature reduction and therefore a local simple model, while the other partition is dominated by local *Synergy* and requires the consideration of all features in a more complicated local model which is able to handle those relevant interdependencies. Depending on the data properties the overall generalization capability can be improved.

Acknowledgements

I wish to thank *Prof. Dr.-Ing. Jürgen Adamy*, *Prof. Dr. Bernhard Sendhoff*, *Andreas Richter*, *Dr.-Ing. Lars Gräning*, *Dipl.-Ing. Florian Damerow* for the great opportunity to write my diploma thesis in cooperation with the *Technische Universität Darmstadt (TUD)* and the *Honda Research Institute Europe GmbH (HRI-EU)*. In particular, I want to thank my supervisors *Dr.-Ing. Lars Gräning* and *Dipl.-Ing. Florian Damerow* for their dedicated, continuous and professional supervision and support. I also like to thank *Dipl.-Ing. Nikola Aulig* for the very helpful technical discussion we had. I am thankful to all the employees of *HRI-EU* for their cooperation and kindness and to the students for all the fantastic moments we shared. Especially, I want to thank *Viktor Losing*, *Benjamin Metka* and *Marius Anderie* for the friendly working atmosphere in our office. Lastly, I thank my parents, my girlfriend and my friends for their moral support.

Contents

1 Introduction

1.1 Motivation

Data mining, machine learning and artificial intelligence have become important domains of the computer science because they provide solutions to analyze and model technical systems. Real systems can be modeled using data that contains example relations of the system variables (data samples).
More data implies more details about the real system. However, today, due to the technological progress (e.g. internet), huge datasets are available that overcharge the construction and learning of current computational models. Huge datasets often contain details that are irrelevant or redundant. Therefore, feature selection method have become a common method to remove less important variables from the data.
The removal of variables allows to construct simpler system models that are expected to have better generalization capabilities, while requiring a smaller number of data samples to learn the system behavior.
However, all feature selection methods try to remove those variables completely that are **on average** mostly irrelevant. Hence, a subset of variables is kept and the rest is lost. Consequently, a variable that is only useful in rare situations, will be completely removed.
Our approach aims to avoid this disadvantage. In order to use the variable in spite of that, a more complicated model can be used for the rare situations, where the variable is relevant. In the rest of the situations the variable is ignored and the **benefits** of a simpler model can take effect.
Assuming that the relevance of a variable depends on the location in the feature space, the feature space can be partitioned and for every partition a corresponding local model can be used.

The main goal of this thesis is the partitioning of the feature space based on the quantification of *Synergy* and *Redundancy* using the *Interaction Information*.
The *Interaction Information* is a measure from information theory, which was already used in modern feature selection methods to identify the most relevant variables [2]. It is a measure for the additional information about a third variable which one is able to obtain through the interaction of two variables. The *Interaction Information* allows to discriminate between *Synergy* and a *Redundancy*. In general the meaning of the word *Synergy* is that a combined effect of two agents is greater than the sum of their separate effects. This description is also appropriate for our case:
In a *Synergy* case two variables do not only provide different information about the third variable, they are even able to produce new information about the third variable which was not contained in the separate variables. In contrast the general meaning of the word *Redundancy* is the state of being not or no longer needed or useful. In our case the equivalent is that two variables provide the same information about the third variable, thus only one variable is required.
In the optimal case the partitioning of the feature space leads to one feature subspace in which a *Synergy* between the variables is detected and one feature subspace in which a *Redundancy* is detected. Two different models are used for both feature subspaces. In the *Synergy* subspace, all variables are considered in the model, while a simpler model can be used in the *Redundancy* feature subspace due to the irrelevance of at least one variable.

A detailed research on the properties of the *Interaction Information* was required to reach the goal and provide three methods for the feature space partitioning based on the *Interaction Information*.

1.2 Overview

In Section 1.3 a few state-of-the-art feature selection methods based on information measures are discussed. Afterwards the *Information Theory Fundamentals* and in particular the *Interaction Information*, which are introduced in Section 2. The *Interaction Information* is an important measure, which allows the quantification of *Synergy* and *Redundancy* in data and in particular in local feature subspaces. Based

on the quantification of *Synergy* and *Redundancy* in local feature subspaces the partitioning methods that are introduced in Section 3.2 are able to partition the two dimensional feature space in two subspaces, where ideally *Synergy* is dominating in one feature subspace and *Redundancy* is dominating in the other feature subspace. In the *Synergy* subspace both features are required. Due to the *Synergy*, more information about the output variable can be gained by combining the knowledge of both features. Therefore, a predictive model that corresponds to that subspace should be able to consider both features in combination. In the *Redundancy* subspace only one feature is required. Due to the *Redundancy*, no extra information about the output variable can be gained from a feature if the other feature is known. Hence, a predictive model that corresponds to that subspace needs only one input, which allows a simpler model structure. Several predictive models are introduced in Section 3.3. In Section 4 the predictive capabilities of global models are compared with pairs of local models. A global model has two inputs in the entire features space. A pair of local models is used in combination with a partitioned feature space, where the local model with two inputs is used in the *Synergy* subspace and the local model with one input is used in the *Redundancy* or alternatively in a *Non-Interaction* subspace. Furthermore, the partitioning methods from Section 3.2 are tested on experimental data in order to compare the partitioning capability of the methods. In Section 5 the most important conclusions of the thesis are summarized and in Section 6 an outlook for future work is presented.

1.3 Related Work

In the past, many feature selection methods based on information measures were developed.
The task of a feature selection method is to find the best feature subset of a exponential large number $(2_f^N - 1)$ of possible feature subsets, where N_f is the number of features [2] [3]. In order to prevent that every possible subset has to be tested, common methods have different underlying assumptions.
One approach of feature selection is to rank the variables (features) by their direct relation to the output variable. At the end the features with the highest rank are selected for the subset.
The feature ranking can be performed under different criteria. For example, a feature can be ranked by its *Mutual Information* with the output variable, which is a measure for the information that a feature shares with the output (see Section 2.2 for more details).
The advantage of the feature ranking approach is its simplicity, only N_f scores and the sorting of the scores have to be computed ($\mathcal{O}(N_f)$). It is also scalable and robust against overfitting [9].
The assumption that the relevance of a feature can be calculated without considering the rest of the available features leads to the following disadvantages: High ranked features can be redundant, thus features in the subset may contain the same information about the output so that a lower ranked feature which is not redundant would be more useful in the subset than a higher ranked feature. Apart from that a feature ranking does not consider the circumstance that features which do not have a direct relation to the output variable can be related to the output variable if they are combined with other variables.

There are also feature selection methods that consider the feature interaction. An implicit consideration of the interaction can be performed by checking how much information about the output variable is lost if a specific feature is missing. Therefore, the *Mutual Information* of the output variable and all available features in combination are compared with the *Mutual Information* of the output variable and all features except a specific one in combination. Thus, features can be ranked by the resulting loss in case of their absence. However, if a feature is relevant in combination with all available features (which is the assumption) it can be less relevant in combination with a subset of features. Therefore, this method does not necessarily lead to the optimal subset.

An explicit consideration of the interaction can be performed by using the *Interaction Information*. The *Interaction Information* allows to check whether two features in combination are providing more information about the output variable than they provide separately. If that is the case, there is a strong *Synergy* between the features and the features should be used in combination and therefore both should

be kept in the feature subset.
If the two features in combination provide less information about the output than they provide separately, there is a strong *Redundancy* between the features and it is promising to use only one of them. However, there are different degrees of *Synergy* and *Redundancy*. The *Interaction Information* is described in detail in Section 2.2.

The *Interaction Gain Based Feature Selection (IGFS)* [2] uses the *Interaction Information* and *Mutual Information* at the same time, while the relevance of new features is always calculated by considering only the already selected features. Hence, the feature subset is determined stepwise, requiring a reranking of the remaining features each time. The ranking criterion has two components: The first one is a high *Mutual Information* of the output variable and the feature. The second one is a high *Interaction Information* with the already selected features and the output variable.

The *Minimum Redundancy - Maximum Relevance criterion (mRMR)* [2] [16] is a feature selection method that as well as the IGFS ranks the remaining features considering the already selected features. The criterion aims at two desired properties of a feature: The first property is a high *Mutual Information* of the output variable and the feature (maximum relevance) and the second property is a small *Mutual Information* of the feature and the already selected features (minimum redundancy).

The *Max-Dependency* feature selection [16] is also a method that selects features incremental. At first the feature with the highest *Mutual Information* with the output variable is taken. Afterwards in every further step the feature is selected that in combination with the already selected features has the highest *Mutual Information* with the output variable.

The *Conditional Mutual Information Maximization Criterion (CMIM)* [2] is another feature selection method that selects the next feature under consideration of the already selected features. At first for every possible combination of a remaining feature and an already selected feature the *Conditional Mutual Information* of the feature and the output variable is calculated given a feature that is already selected. Hence, for every remaining feature as many *Conditional Mutual Information* values were calculated as features are already selected. Then for every remaining feature only the smallest of its *Conditional Mutual Information* values is considered and the feature which has the highest minimum *Conditional Mutual Information* is selected.

However, all feature selection methods are used to decrease the number of features for one global model and therefore are considering the entire feature space at once.

Our feature space partitioning based on the *Interaction Information* is a preparation for a local feature selection. The feature space is partitioned in one subspace, where only one feature is relevant and a second subspace, where both features are relevant. A separate local feature selection for both feature subspaces is the base for two local sub models of different complexities.

In contrast to a decision tree [19], which also performs a local feature selection, the goal is to generate one complex local model and one simple local model. In contrast, the decision tree aims to partition the feature space in a way that that both local models have a similar complexity. A comparison can be found in the experiments section.

2 Information Theory Fundamentals

The fundamentals of modern communication and information theory goes back to Claude E. Shannon [22]. With his theory, Shannon provides the basis of the quantification of information of data, which is used in different domains like applied math, electrical engineering and computer science. The information theory provides measures to describe several kinds of information which are contained in data. These measures has successfully been used in data mining applications and feature selection methods (see Section 1.3). They are also required to study the interactions of variables which is one of the central points of this thesis.

2.1 Shannon Entropy

The *Shannon Entropy* is a measure for the uncertainty about a discrete random variable X if no previous knowledge about X is available.
For a discrete random variable X that can take n discrete values $x_1, ..., x_n$, the *Shannon Entropy* [22] is defined as

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 (p(x_i)), \tag{1}$$

where $p(x_i)$ is the probability of the event that X takes the value x_i. Here the base 2 of the logarithm was used. Entropies with a base 2 are measured in the unit bit. Using another base b would only mean a multiplication of the entropy in bit by a constant value $c_b = \log_b(2)$. The probability $p(x_i)$ takes values between 0 and 1. In this interval $log_2(p(x_i))$ is negative for all values if more than one event is possible ($p(x_i) < 1$) (see also Figure 64 on page 113). The sum of all probabilities $p(x_i)$ is 1. Hence, for more than one possible event, $p(x_i) < 1$, the term $-p(x_i) \cdot log_2(p(x_i))$ is always positive (see also Figure 65 on page 113). Because the entropy is the sum of these positive terms, the the entropy is always positive for $n > 1$.

If X_u is a uniformly distributed random variable, all probabilities $p(x_i)$ are equal ($p(x_i) = \frac{1}{n}$) and the entropy simplifies to

$$H(X_u) = -\sum_{i=1}^{n} p(x_i) \log_2(p(x_i)) = -n \cdot \frac{1}{n} \cdot \log_2\left(\frac{1}{n}\right) = -\log_2\left(\frac{1}{n}\right) = \log_2(n). \tag{2}$$

The value $\log_2(n)$ is also the number of bits that is required to discriminate between n cases. Uniformly distributed random variables X_u have the maximum entropy that a random variable can have for a given number n of discrete values that the variable can take. A random variable $X_{\bar{u}}$ that can take n values which is not uniformly distributed has a smaller entropy:

$$H(X_{\bar{u}}) < H(X_u), \tag{3}$$

where X_u and $X_{\bar{u}}$ can respectively take n discrete values.

Joint Entropy
The Shannon entropy of multiple variables in combination is called the *Joint Entropy* and is based on the joint probability distribution of the variables. The *Joint Entropy* of two and three discrete random variables is defined as follows:

$$H(X,Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) \log_2 \left(p(x_i, y_j)\right), \tag{4}$$

$$H(X,Y,Z) = -\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{q} p(x_i, y_j, x_k) \log_2 \left(p(x_i, y_j, x_k)\right), \tag{5}$$

where X, Y and Z are discrete random variables and n is the number of values that X can take , m is the number of values that Y can take and q is the number of values that Z can take. The *Joint Entropy* for more than three variables is defined in [6].

The *Joint Entropy* of the variables X and Y equals the entropy of the variable $V = (X,Y)$, where V can take at most $h = n \cdot m$ combinations $v_k = (x_i, y_j)$, $i = 1,...,n$, $j = 1,...,m$, $k = 1,...,h$ with the probabilities $p(v_k) = p(x_i, y_j)$.

The *Joint Entropy* of X and Y is at most $H(X) + H(Y)$, which is the case if X and Y are independent,

$$H(V) = H(X,Y) \leq H(X) + H(Y). \tag{6}$$

Conditional Entropy
The *Conditional Entropy* $H(Y|X)$ is the rest of uncertainty of the discrete random variable Y if the value of the discrete random variable X is already known. It is defined as

$$\begin{aligned}
H(Y|X) &= H(X,Y) - H(X) && (7)\\
&= \left(-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left(p(x_i,y_j)\right)\right) - \left(-\sum_{i=1}^{n} p(x_i)\log_2\left(p(x_i)\right)\right) && (8)\\
&= \left(-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left(p(x_i,y_j)\right)\right) - \left(-\sum_{i=1}^{n}\left(\sum_{j=1}^{m} p(x_i,y_j)\right)\log_2\left(p(x_i)\right)\right) && (9)\\
&= -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left(\frac{p(x_i)}{p(x_i,y_j)}\right). && (10)
\end{aligned}$$

If X and Y are independent, $H(Y|X)$ is equal to $H(Y)$.

2.2 Information Measures

The information measures describe the information that variables share among each other, where different conditions can be taken into account. They are defined by differences of several entropy measures. Information measures allow to quantify the interdependencies of variables.

Mutual Information
The *Mutual Information* or *Transinformation* describes the amount of information that two discrete random variables X and Y mutually share among each other. It is defined as

$$\begin{aligned}
I(X;Y) &= H(Y) - H(Y|X) && (11)\\
&= \left(-\sum_{j=1}^{m} p(y_j)\log_2\left(p(y_j)\right)\right) - \left(-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left(\frac{p(x_i)}{p(x_i,y_j)}\right)\right) && (12)\\
&= \left(-\sum_{j=1}^{m}\left(\sum_{i=1}^{n} p(x_i,y_j)\right)\log_2\left(p(y_j)\right)\right) - \left(-\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left(\frac{p(x_i)}{p(x_i,y_j)}\right)\right) && (13)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i,y_j)\log_2\left(\frac{p(x_i,y_j)}{p(x_i)\cdot p(y_j)}\right). && (14)
\end{aligned}$$

The *Mutual Information* $I(X;Y)$ is the difference between the uncertainty of Y and the uncertainty of Y if X is known (also see the left side of Figure 1).
Hence, knowing X, the uncertainty $H(Y)$ is reduced by $I(X;Y)$ and the remaining uncertainty is $H(Y|X)$.
The *Mutual Information* $I(X;Y)$ is always non-negative because of relation (6), which leads to

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \geq 0 \Longleftrightarrow H(X,Y) \leq H(X) + H(Y). \tag{15}$$

If X and Y are independent, they do not share any information among each other:

$$I(X;Y) = 0 \Longleftrightarrow X \text{ and } Y \text{ are independent.} \tag{16}$$

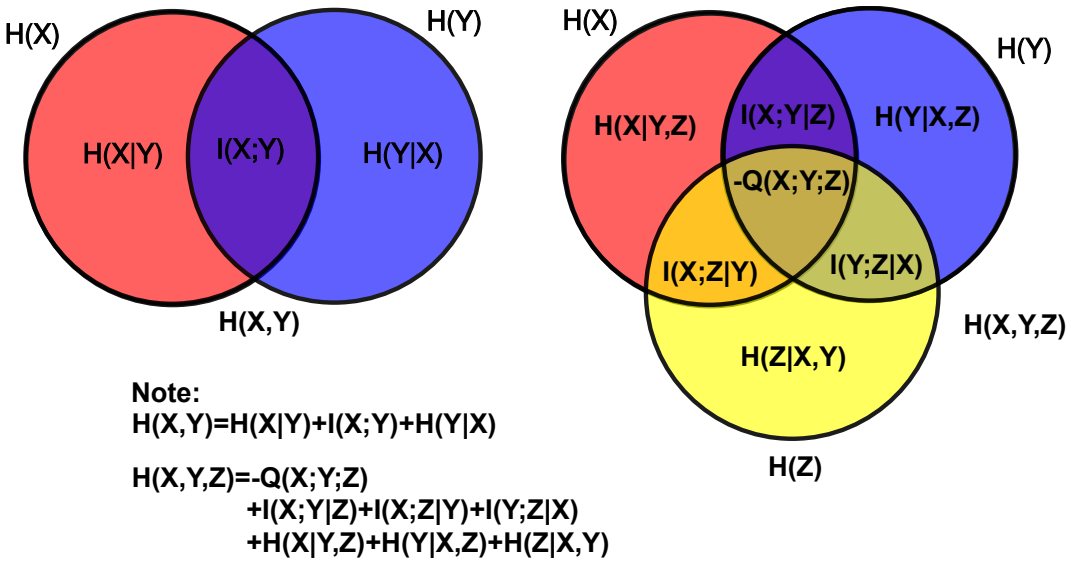

Figure 1: Venn diagrams of *Mutual Information I(X;Y)* and *Interaction Information Q(X;Y;Z)*[1]
Remark: The *Mutual Information* labeled with I and the *Entropies* labeled with H are always positive, while the area which is labeled with $-Q(X;Y;Z) = I(X;Y) - I(X;Y|Z)$ can represent a positive or a negative value. Even though the area of $I(X;Y)$ contains the area of $I(X;Y|Z)$, it is not possible to find out whether $I(X;Y|Z)$ or $I(X;Y)$ is greater by reading the diagram. But in the diagram the surface of $I(X;Y)$ is greater than the surface of $I(X;Y|Z)$, hence, the remaining surface is $-Q(X;Y;Z) = I(X;Y) - I(X;Y|Z)$ that is the reason why the *Interaction Information* area is labeled with $-Q(X;Y;Z)$ here, although the area is usually labeled with $Q(X;Y;Z)$ e. g. in [5] and [4].

Conditional Mutual Information
The *Conditional Mutual Information* describes the amount of information that X and Y share among each other if the value of Z is known.

$$I(X;Y|Z) = H(X,Y) + H(Y,Z) - H(X,Y,Z) - H(Z) \tag{17}$$

$$= \left(-\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q} p(x_i,y_j,x_k)\log_2\left(p(x_i,y_j)\right)\right) + \left(-\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q} p(x_i,y_j,x_k)\log_2\left(p(y_j,z_k)\right)\right)$$
$$- \left(-\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q} p(x_i,y_j,x_k)\log_2\left(p(x_i,y_j,x_k)\right)\right) - \left(-\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q} p(x_i,y_j,x_k)\log_2\left(p(z_k)\right)\right) \tag{18}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q} p(x_i,y_j,z_k)\log_2\left(\frac{p(z_k)\cdot p(x_i,y_j,z_k)}{p(x_i,z_k)\cdot p(y_j,z_k)}\right). \tag{19}$$

[1] The diagram is inspired by the venn diagrams from [5] and [4]

The *Conditional Mutual Information* $I(X;Y|Z)$ is always positive or zero and can be larger or smaller than the *Mutual Information* $I(X;Y)$, it is even possible that they are equal.

Interaction Information

The *Interaction Information* of three discrete random variables X, Y and Z describes the difference between the information that X and Y in combination share with Z and the sum of the information that X and Y separately share with Z. The *Interaction Information* [6][7][18][23][12] is also called *Multivariate Mutual Information* or *Interaction Gain* [2][11]. The *Interaction Information* for three discrete random variables is defined as.

$$Q(X;Y;Z) = I(X,Y;Z) - [I(X;Z) + I(Y;Z)], \tag{20}$$

where

$$I(X,Y;Z) = H(X,Y) + H(Z) - H(X,Y,Z), \tag{21}$$
$$I(X;Z) = H(X) + H(Z) - H(X,Z), \tag{22}$$
$$I(Y;Z) = H(Y) + H(Z) - H(Y,Z). \tag{23}$$

Thus, the *Interaction Information* can also be described by entropies:

$$\begin{aligned} Q(X;Y;Z) &= H(X,Y) + H(Z) - H(X,Y,Z) && (24)\\ &\quad - [H(X) + H(Z) - H(X,Z) + H(Y) + H(Z) - H(Y,Z)] && (25)\\ &= H(X,Y) + H(X,Z) + H(Y,Z) - H(X) - H(Y) - H(Z) - H(X,Y,Z) && (26)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q} p(x_i,y_j,z_k) \log_2 \left(\frac{p(x_i,y_j,z_k)\cdot p(x_i)\cdot p(y_j)\cdot p(z_k)}{p(x_i,y_j)\cdot p(x_i,z_k)\cdot p(y_j,z_k)} \right). && (27) \end{aligned}$$

The value of $Q(X;Y;Z)$ does not change if X, Y and Z are pairwise swapped.

Alternatively, the *Interaction Information* can be described by the difference between the information that X and Y share given Z and the information that X and Y share if Z is not given,

$$Q(X;Y;Z) = I(X;Y|Z) - I(X;Y). \tag{28}$$

This description is used to explain the *Interaction Information* in Figure 2.
The picture shows how the information about Z provided by X and Y separately can be compared with the information about Z that X and Y provide in combination.
If Z is known the information that is shared between X and Y can increase ($I(X;Y|Z) > I(X;Y)$) or decrease ($I(X;Y|Z) < I(X;Y)$), consequently, the *Interaction Information* is in contrast to the *Mutual Information* and the *Conditional Mutual Information* able to get negative.

Interpretation of the Interaction Information
For the interpretation of the *Interaction Information* we distinguish between four cases:

Synergy ($Q(X;Y;Z) > 0$)
A positive value of the *Interaction Information* indicates that we gain information about a third variable from the *Synergy* of the other two.
If $I(X,Y;Z) > I(X;Z) + I(Y;Z)$, X and Y in combination provide more information about Z than X and Y provide separately, $Q(X;Y;Z)$ becomes positive. Hence, if you want to predict for example the

variable Z, you gain more information about Z if you consider X and Y in combination.

Redundancy ($Q(X;Y;Z) < 0$)
A negative value of the *Interaction Information* indicates that two variables provide redundant information about a third variable.
If $I(X,Y;Z) < I(X;Z)+I(Y;Z)$, X and Y in combination contain less information about Z than X and Y separately contain, $Q(X;Y;Z)$ becomes negative. Thus, if you want to predict for example the variable Z, there is redundant information about Z contained in X and Y

Non-Interaction ($Q(X;Y;Z) = 0$)
If X is independent from Y and Z and also from the combination X,Y, the entropies can be replaced as follows:

$$H(X,Y,Z) = H(X) + H(Y,Z) \Longleftrightarrow X \text{ and } (Y,Z) \text{ are independent,} \tag{29}$$
$$H(X,Z) = H(X) + H(Z) \Longleftrightarrow X \text{ and } Z \text{ are independent,} \tag{30}$$
$$H(X,Y) = H(X) + H(Y) \Longleftrightarrow X \text{ and } Y \text{ are independent.} \tag{31}$$

As result the *Interaction Information* vanishes:

$$\begin{aligned} Q(X;Y;Z) &= H(X,Y) + H(X,Z) + H(Y,Z) - H(X) - H(Y) - H(Z) - H(X,Y,Z) \\ &= H(X) + H(Y) + H(X) + H(Z) + H(Y,Z) - H(X) - H(Y) - H(Z) - H(X) - H(Y,Z) \\ &= 0. \end{aligned} \tag{32}$$

A more obvious way to describe this case can be obtained after the variables X and Z have been exchanged in the equation:

$$\begin{aligned} Q(X;Y;Z) &= I(X,Y;Z) - [I(X;Z) + I(Y;Z)] \\ &= I(Z,Y;X) - [I(Z;X) + I(Y;X)] \end{aligned} \tag{33}$$

Because of the independencies, all summands of the *Interaction Information* are zero:

$$I(Z,Y;X) = 0 \Longleftrightarrow X \text{ and } (Y,Z) \text{ are independent,} \tag{34}$$
$$I(Z;X) = 0 \Longleftrightarrow X \text{ and } Z \text{ are independent,} \tag{35}$$
$$I(Y;X) = 0 \Longleftrightarrow X \text{ and } Y \text{ are independent,} \tag{36}$$
$$Q(X;Y;Z) = 0 \Longleftrightarrow X \text{ is independent from } X,\ Y \text{ and } (Y,Z). \tag{37}$$

In this case, X contains and complets no information of Y and Z. Therefore, X is not helpful in any sense if Y or Z has to be predicted.
The Non-Interaction case does not contain *Redundancy* or *Synergy*.

Compensation ($Q(X;Y;Z) = 0$)
There is also the case possible that *Redundancy* and *Synergy* are canceling each other out. That can happen if data contain *Redundancy* and *Synergy* (an example can be found in the Appendix A).

In the Appendix A, the manual calculation of the *Interaction Information* is illustrated for four small datasets, which are examples for *Synergy*, *Redundancy*, *Non-Interaction* and *Compensation*.

Normalization of the Interaction Information
As seen in Figure 1 the maximum absolute value of the *Interaction Information* is bounded by the smallest entropy value of the involved variables,

$$|Q(X;Y;Z)| \leq \min(H(X),H(Y),H(Z)). \tag{38}$$

Therefore, a normalized measure of the *Interaction Information* can be created as follows:

$$-1 \leq \frac{Q(X;Y;Z)}{\min(H(X),H(Y),H(Z))} \leq 1. \tag{39}$$

The normalization allows a better evaluation of the *Interaction Information*, which also can be represented in percent (%). The boundaries can be interpreted in the following manner:

$$\frac{Q(X;Y;Z)}{\min(H(X),H(Y),H(Z))} = 100\% \text{ (pure Synergy)},$$
$$\frac{Q(X;Y;Z)}{\min(H(X),H(Y),H(Z))} = -100\% \text{ (pure Redundancy)}.$$

However, if the normalized value of $Q(X;Y;Z)$ should describe the relative ratio of information about the variable Z that is provided by the *Interaction Information* of X, Y and Z, the normalization has to be calculated in relation to $H(Z)$:

$$-c \leq \frac{Q(X;Y;Z)}{H(Z)} \leq c \text{ (Percentage of } H(Z) \text{ that is covered by the } \textit{Synergy} \text{ or } \textit{Redundancy} \text{ of } X \text{ and } Y),$$
$$c = \frac{\min(H(X),H(Y),H(Z))}{H(Z)},\ 0 \leq c \leq 1. \tag{40}$$

If $c < 1$ i. e. $\min(H(X),H(Y),H(Z)) < H(Z)$, $|\frac{Q(X;Y;Z)}{H(Z)}|$ can not achieve 100% because $H(X)$ or $H(Y)$ or both are smaller than $H(Z)$ and therefore $Q(X;Y;Z) < H(Z)$.

Interaction Information for more than three variables
The *Interaction Information* for more than three variables does not allow the identification of *Synergy* and *Redundancy* by observing whether it is positive or negative.
For example, if there are four variables, there can be a relation of the values $s = x \oplus y \oplus z$, which describes a *Synergy* and the relation $s = x = y = z$, which describes a *Redundancy*. In both cases the *Interaction Information* is $+1bit$ [24].

The definition of the generalized *Interaction Information* for n variables can be found in [4] and [6].

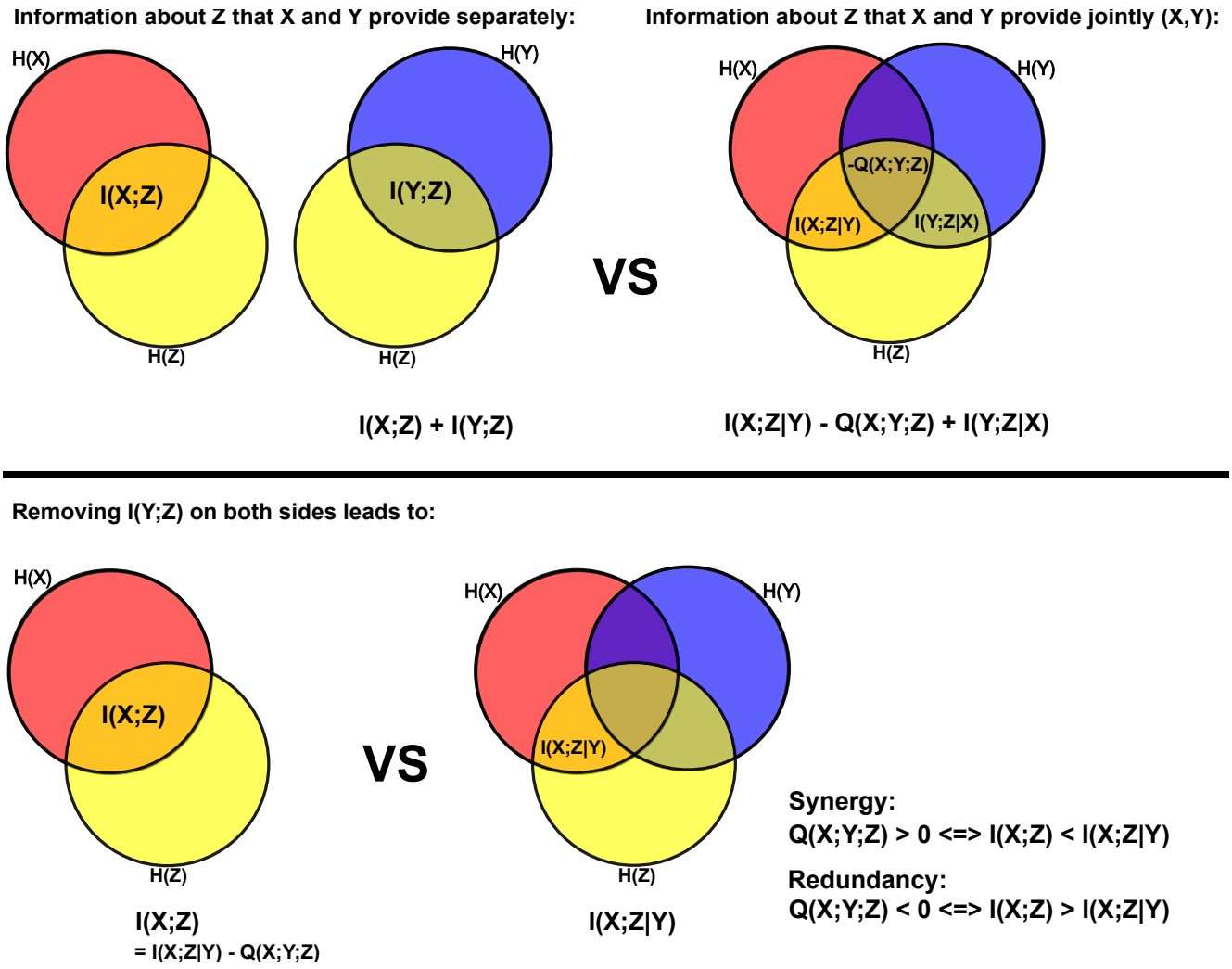

Figure 2: Venn diagrams: Explanation of the *Interaction Information* by comparing the *Mutual Information* with the *Conditional Mutual Information*

2.3 Data preparation, Conditions, Assumptions and Definitions

In order to use the information measures from Section 2.2 a suitable *Discrete Probability Density Function* of the variables is required. The purpose of this section is to discuss how a suitable *Discrete Probability Density Function* can be obtained from given data.

Data can consist of different variable types. We discriminate between the following types:

Categorical Variables (Classes)
Every value that a *categorical variable* can take stands for a category or class. The only relation of two different categories is that they are unequal.

Ordinal Variables (Ranks)
Every value that an *ordinal variable* can take stands for a rank or level. Therefore, it is known if one value is higher or smaller than another value, but not how great the difference of both is.
The values can also be understood as ordered categories.

Interval Variables (Numbers)
Every value that an *interval variable* can take stands for a rank while the distance between two consecu-

tive ranks has a certain defined distance. This distance is the minimal distance that two ranks or values can have. Therefore, two arbitrary values have a distance which is a multiple of the minimal distance. Numbers that are used in digital domains are usually described by interval variables. The accuracy of a variable type which is used to represent numbers is determined by the minimal distance.

Continuous Variables
Continuous variables can take infinite different values that are lying in a finite interval. Therefore, the accuracy of a variable is infinite and can not be described by a digital variable.

Discrete Variables
A *discrete variable* can take a finite number of different values that are lying in a finite interval. Hence, the accuracy of a variable is finite and can be described by a digital value e. g. by an *interval variable*. *Discrete variables* are used to approximate *continuous variables*. In the communications engineering domain an analog-to-digital-converter is used to quantize the continuous values by dividing the continuous space into intervals of equal size and replacing all values that are lying in an interval by the one value that is positioned in the middle of the interval.

Discrete Probability Density Function
The information theory formulas in Section 2 are based on the entropy formula. The entropy formula [equation (1)] describes the entropy of a random variable and is calculated from probabilities of the discrete values that the variable can take. Therefore, a *Discrete Probability Density Function* is a precondition to use the mentioned formulas. The entropy can also be calculated based on an exact continuous probability density function, which can not be reconstructed from data with a finite number of samples. A *Discrete Probability Density Function* can be calculated from the data as showed in the calculation examples in Appendix A.
The the number of parameters of the *Discrete Probability Density Function* of a discrete variable equals the number of diverse values that a variable can take. Every parameter stands for the probability that the corresponding variable takes a certain value. Consequently, the parameter number of the *Discrete Probability Density Function* increases with the accuracy of the discrete variable. Because the *Discrete Probability Density Function* is comparable with a model, which is trained by data samples, overfitting occurs if the number of samples is to low for the number of parameters.
For example, if the data contains 1,000 samples with values of a variable and the variable can take 10,000 values. Then there will be at least 9,000 possible values which did not occur in the data. Consequently, their probability of occurrence is assumed to be zero. If this is the case, the *Discrete Probability Density Function* is overfitted to the data. The number of parameters is too high compared to the number of data samples for reliable determinations of the parameters.
The smaller the number of parameters the better the reliability of the parameter determination for a given number of samples.

Quantization of variables that are already discrete
In order to reduce the number of parameters of the *Discrete Probability Density Function*, a discrete variable can be quantized to reduce the number of values that it can take. Therefore, the space is divided into intervals of equal size and all values that are lying in an interval are replaced by the value that is positioned in the middle of the interval. The number of intervals is the new number of values that the quantized variable can take and therefore the new number of parameters of the *Discrete Probability Density Function*. The number of examples of the variable should be much larger than the number of parameters in order to avoid overfitting.
With a simplified model of the *Discrete Probability Density Function* of a variable, less data samples of the variable are required for a reasonable approximation of the *Discrete Probability Density Function*. In order to determine the entropy of a variable, a reasonable approximation of the *Discrete Probability Density*

Function of the variable is required.

Comparison of the variable types for the entropy calculation
The entropy equation (1) treats the variables as *categorical variables* because it does not take into account if one value is higher than another one. However, the parameter number of the *Discrete Probability Density Function* of an interval or an ordinal variable can be reasonably reduced by a quantization. That means similar values are grouped to one value in order to reduce the number of values that are distinguished by the *Discrete Probability Density Function*. The number of values that *categorical variables* can take is usually not reducable by grouping similar categories in one category because there is no measure of similarity between the categories. However, the number of existing categories is supposed to be smaller than the number of values that interval variables can take because the categories are often defined manually by humans. Humans usually do not tend to discriminate between huge numbers of categories.

3 Feature space partitioning and learning methods

3.1 Local properties of the Interaction Information

In this section the influence of a feature space partitioning on the *Interaction Information* is discussed.

3.1.1 Synergy and Redundancy are local properties of the feature space

In this section the *Synergy* and *Redundancy* which were introduced in Section 2.2 are considered in a discrete feature space. The feature space consists of combinations of x_i and y_j values, while z_k describes the output values, where $i \in \{1, ..., n\}$, $j \in \{1, ..., m\}$, $k \in \{1, ..., q\}$.
The *Interaction Information* can be calculated from the entropies $H(X)$, $H(Y)$, $H(Z)$, $H(X,Y)$, $H(X,Z)$, $H(Y,Z)$ and $H(X,Y,Z)$. $H(X)$ can be determined from the *Discrete Probability Density Function* $p(x_i), \forall i \in \{1, ..., n\}$ and $H(X,Y)$ from $p(x_i, y_j), \forall i \in \{1, ..., n\}, j \in \{1, ..., m\}$ and so on.

Assume that the *Discrete Probability Density Functions* of X, Y, Z and all possible combinations of the variables are known. For example, the *Discrete Probability Density Functions* can be approximated based on given data samples.
Every *Discrete Probability Density Function* can be represented by a histogram. The *Discrete Joint Probability Density Functions* like $p(x_i, y_j)$ are not appropriate for a simple illustration, hence, Figure 3 depicts an arbitrary *Discrete Probability Density Function* $p(x_i)$ of X as an example, where the probability of every value x_i that X can take, is specified.

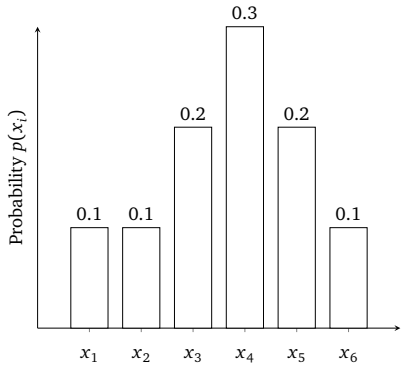

Figure 3: Historgram of an arbitrary *Discrete Probability Density Function* $p(x_i)$

Local Feature Spaces and Feature Space Partitioning
A local feature subspace, is a smaller part of the entire feature space. A local property is a property that can be determined for every considered local subspace, while the result depends on the areas that the chosen local subspace contains. There are two types of feature spaces that are considered here. The first type is a space, where a position consists of values of *interval variables*, therefore, a distance or similarity between two positions in the space can be considered. The more general first type is a space, where a position consists of values of *categorical variables*, hence, there is no meaningful consideration of distance or similarity between two positions possible.
The second space type is a space, where a similar relation between variables is expected in local areas, due the circumstance that values with small distances from each other have similar meanings. In that kind of spaces, it can be reasonable to form local subspaces that contain samples which have small distances from each other in the feature space.

Looking at Figure 3 again, the values $x_1, ..., x_6$ can represent categories or numbers. However, categories are discussed here because conclusions about the information of categories can be transferred on the less

general case of numbers.

A partitioning of the feature space produces a new data sample subset for every feature subspace, where the only condition is that the samples which have the same input value combinations have to be in the same data sample subset because the partitioning can only discriminate samples in the feature space but not based on the output value. An arbitrary partitioning of the data sample set is performed by assigning all samples, where $x_i \in \{x_1, x_2, x_6\}$ to feature subspace 1 and all samples, where $x_i \in \{x_3, x_4, x_5\}$ to feature subspace 2. Figures 4 and 5 are depicting the histograms of X in both feature subspaces.

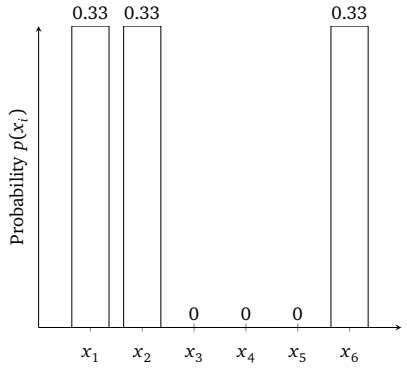

Figure 4: Historgram of the *Discrete Probability Density Function* $p(x_i)$ of feature subspace 1

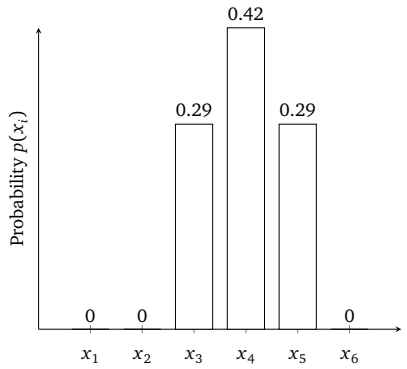

Figure 5: Historgram of the *Discrete Probability Density Function* $p(x_i)$ of feature subspace 2

The *Discrete Probability Density Functions* $p(x_i)$ of X in the local feature subspaces are and the entire feature space differ from each other. Hence, the entropy $H(X)$ is different in every local feature subspace.

Entropies
Entire feature space: $H(X) = -3 \cdot 0.1 \log_2(0.1) - 2 \cdot 0.2 \log_2(0.2) - 0.3 \log_2(0.3) = 2.4464 bit$
Feature subspace 1: $H(X_{sub1}) = -3 \cdot 0.3333 \log_2(0.3333) = 1.5849 bit$
Feature subspace 2: $H(X_{sub2}) = -2 \cdot 0.2857 \log_2(0.2857) - 0.4286 \log_2(0.4286) = 1.5566 bit$

The probability that a sample lies in feature subspace 1 is $p(W = w_1) = 0.3$ and the probability that a sample lies in feature subspace 2 is $p(W = w_2) = 0.7$. Hence, knowing the feature subspace provides an information that reduces the entropy $H(X)$.
The entropy of the assignment is: $H(W) = -0.3 \cdot log_2(0.3) - 0.7 \cdot log_2(0.7) = 0.8813 bit$

The exact reduction of the entropy can be determined using the following equations

$$H(X|Y) = H(X,Y) - H(Y) \text{and} \tag{41}$$
$$H(X|Y) = \sum_{y_i} p_Y(y_i) H(X|Y = y_i) \tag{42}$$

from [8].
If the value x is known, the value w of W is also known $H(W|X) = 0$. Using that knowledge in equations (41) and (42) leads to

$$\begin{aligned} H(W|X) &= H(W,X) - H(X) = 0 \Longleftrightarrow H(W,X) = H(X,W) = H(X), \\ H(X|W) &= H(X,W) - H(W) = H(X) - H(W) \\ &= p(W = w_1) \cdot H(X|W = w_1) + p(W = w_2) \cdot H(X|W = w_2) \\ &= p(w_1) \cdot H(X_{sub1}) + p(w_2) \cdot H(X_{sub2}). \end{aligned}$$

As a matter of fact the entropy $H(X)$ is reduced by the entropy $H(W)$ if subsets of the entire data set are considered $H(X) - H(W) = p(w_1) \cdot H(X_{sub1}) + p(w_2) \cdot H(X_{sub2})$.

Also the remaining entropies that occur in the *Interaction Information*

$$Q(X;Y;Z) = H(X,Y) + H(X,Z) + H(Y,Z) - H(X) - H(Y) - H(Z) - H(X,Y,Z)$$

are different in local feature subspaces due to the changes in their *Discrete Probability Density Functions* caused by the partitioning. Even the entropies that do not contain a X have another sample data set, therefore, they can change, too.
The experiments in Section 4.4.1 are showing how different kinds of partitionings affect the *Interaction Information* in local feature subspaces.

3.1.2 Information Decomposition: Information Maps

In order to create *Information Maps* that allow to depict the local shares of information measures, the information measures can be decomposed in their local summands.
As already shown in Section 2 the information measures can be determined from the marginal and joint probability distributions of X, Y and Z.

Mutual Information

$$I(X;Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) \log_2 \left(\frac{p(x_i, y_j)}{p(x_i) \cdot p(y_j)} \right) \tag{43}$$

Conditional Mutual Information

$$I(X;Y|Z) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{q} p(x_i, y_j, z_k) \log_2 \left(\frac{p(z_k) \cdot p(x_i, y_j, z_k)}{p(x_i, z_k) \cdot p(y_j, z_k)} \right) \tag{44}$$

Interaction Information

$$Q(X;Y;Z) = I(X;Y|Z) - I(X;Y) \tag{45}$$

$$Q(X;Y;Z) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{i,j} \tag{46}$$

$$a_{i,j} = -p(x_i,y_j)\log_2\left(\frac{p(x_i,y_j)}{p(x_i)\cdot p(y_j)}\right) + \sum_{k=1}^{q} p(x_i,y_j,z_k)\log_2\left(\frac{p(z_k)\cdot p(x_i,y_j,z_k)}{p(x_i,z_k)\cdot p(y_j,z_k)}\right) \tag{47}$$

These formulas are sums of components which correspond to local parts of the probability distributions of the variables. Hence, conclusions about influences of the local probabilities can be made. However, we have to keep in mind, that a local probability is influenced by data samples from all locations, because it is calculated relatively to the probabilities at other locations. Therefore, a partitioning of the feature space would change the local probabilities in the local feature subspaces and simultaneously the local components.

***Mutual Information* share at position** (x,y):

$$i(x,y) = p(x,y)\log_2\left(\frac{p(x,y)}{p(x)\cdot p(y)}\right) \tag{48}$$

***Conditional Mutual Information* share at position** (x,y):

$$i_z(x,y) = \sum_{k=1}^{q} p(x,y,z_k)\log_2\left(\frac{p(z_k)\cdot p(x,y,z_k)}{p(x,z_k)\cdot p(y,z_k)}\right) \tag{49}$$

***Interaction Information* share at position** (x,y):

$$q_z(x,y) = i_z(x,y) - i(x,y) \tag{50}$$

where the local values can be calculated for all discrete locations (x,y), $x \in \{x_1,...,x_n\}, y \in \{y_1,...,y_m\}$ in the feature space.
The values at all positions (x,y) in the feature space together can be drawn on a map that we call *Information Map* or in particular *Mutual Information Map*, *Conditional Mutual Information Map* and *Interaction Information Map*.
Interesting examples of the *Information Maps* and a few more remarks can be found in Section 4.2.1.

Because the *Interaction Information Map* provides very useful information about the local influences of the *Interaction Information* it is used in the partitioning *Method 3: Map Reinforcement*, which provides a good partitioning capability, while being the fastest of the three developed methods (see Section 3.2.3).

Utilization of a partitioned feature space
The partitioned feature space should have a feature subspace, where more information about the output variable can be gained by combining the knowledge of both features and a second feature subspace, where no extra information about the output variable can be gained from a feature if the other feature is known. In that case two local models of different complexity can be used. One complex local model that considers interactions of two inputs and one simple local model that has only one input.
Some predictive models are able to remove irrelevant features autonomous, if that is the case, two identical models can be used as local models, where one of the models simplifies its structure while training. However, many models are not able to remove features, therefore, the following rules for a local feature selection can be used in order to two local models with a different number of inputs correctly:

Local feature selection rules for a local feature subspace:
The following rules can be applied for every feature subspace, where $0 < \tau_1, \tau_2 \leq 1$ are thresholds. The higher the thresholds the more the rules tend to remove features. Small values of τ_1 and τ_2 are reasonable.
1) If $\frac{Q(X;Y;Z)}{H(Z)} > \tau_1$, both features are relevant due to a *Synergy*.
2) If $-\frac{Q(X;Y;Z)}{H(Z)} > 1 - \tau_1$, both features provide redundant information about the output variables, thus, one feature can be removed. If $I(X;Z) > I(Y;Z)$, Y should be removed, else X should be removed.
3) If $|\frac{Q(X;Y;Z)}{H(Z)}| < \tau_1$ and $\frac{I(X;Z)}{H(Z)} < \tau_2$, the interaction of X and Y as well as the information that X provides about the output variable are negligible. X can be removed.
4) If $|\frac{Q(X;Y;Z)}{H(Z)}| < \tau_1$ and $\frac{I(Y;Z)}{H(Z)} < \tau_2$, the interaction of X and Y as well as the information that Y provides about the output variable are negligible. Y can be removed.
5) If $|\frac{Q(X;Y;Z)}{H(Z)}| < \tau_1$ and $\frac{I(X;Z)}{H(Z)}, \frac{I(Y;Z)}{H(Z)} > \tau_2$ the interaction of the features X and Y is negligible but both features provide unequal information about the output variable. No feature can be removed. The model does not require a consideration of an interaction (e.g. Naive Bayesian Classifier).

The partitioning methods aim in particular on finding feature subspaces, where *Synergy* or *Redundany* is dominant.
However, the partitioning criterion can be modified in order to find feature spaces, where *Non-Interaction* is dominating by minimizing the information measures from rule 3) or 4).
It would also be possible to consider the information measures of both feature subspaces at the same time in order to maximize the *Synergy* in one feature subspace and maximize the *Redundancy* in the other feature subspace but a part of the *Synergy* could be lost using that approach.

Introduction that refers to all partitioning methods
The following methods are designed to partition the 2D feature space into two feature subspaces. All methods decompose the feature space in N rectangular sectors of equal size. A few examples for sector decompositions of the feature space are illustrated in Figure 6-9. In order to define a sector size, the number of neighboring values, that are covered in x-direction and in y-direction by one sector, is defined. The smallest sector size (1x1) covers only one discrete x-y-value combination. The size of the sectors defines the number of required sectors to cover the feature space and thereby the complexity of the sector assignment problem. Every sector is either assigned to the feature subspace 1 or to the feature subspace 0. All assignments can be described by a binary number that consists of as many bits as sectors are used. The goal of the partitioning methods is for example to find the feature subspace where the corresponding subdataset reaches the highest *Interaction Information*. The methods can also be modified to find feature subspaces using other optimization criteria. The number of possible sector assignment combinations is $2^{(N-1)}$ if at least one of the N sectors is assigned to every feature subspace. As well as the feature selection problem this problem is a NP-hard problem, where all combinations has to be

tested to be sure, that a global optimum was found. Therefore, if the number of sectors N is small, all $2^{(N-1)}$ sectors assignment combinations can be tested and the best combination can be determined by comparing the results.
The following methods are designed for high numbers of N. For high numbers of N the search for the best combination is performed, while making assumptions that reduce the search space to decrease the search time. That means that usually only a subset of possible combinations was tested until the methods stop. Therefore, the methods do not guarantee to provide the globally optimal result.

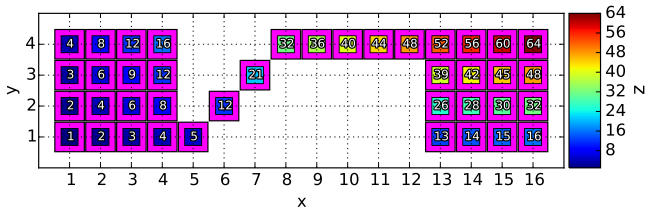

Figure 6: Decomposition of the feature space in sectors with a size of 1x1. Every rectangle in magenta is one sector.

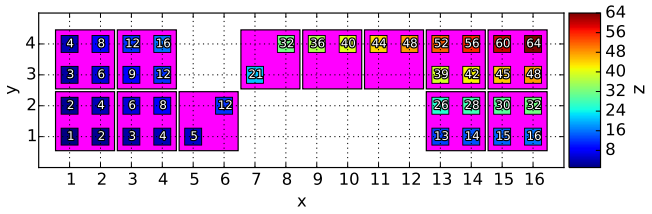

Figure 7: Decomposition of the feature space in sectors with a size of 2x2. Every rectangle in magenta is one sector.

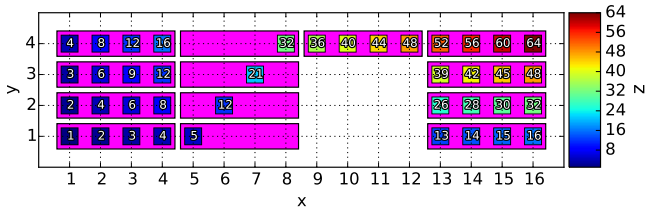

Figure 8: Decomposition of the feature space in sectors with a size of 4x1. Every rectangle in magenta is one sector.

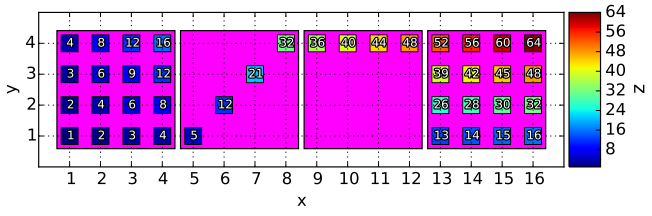

Figure 9: Decomposition of the feature space in sectors with a size of 4x4. Every rectangle in magenta is one sector.

3.2.1 Method 1: Genetic Algorithm based feature space partitioning

Method 1 evaluates definite combinatorical sector assignments while the optimization. In contrast, the optimization process of the other methods considers relaxed sector assignments.

A *Genetic Algorithm* (GA) is based on mechanisms that were observed in the biological evolution. Every genotype is a representation of the corresponding phenotype. There is a mapping from certain genotype to the corresponding phenotype known, but not vice versa. The genotype can be manipulated in order to optimize the phenotype in terms of maximizing the fitness of the phenotype. Mechanisms like *Selection*, *Recombination* and *Mutation* that are known from the biological evolution are used to find a genotype that produces a phenotype with the requested condition. The requested condition is represented by a fitness function, that has to be maximized while the optimization.

For the feature space partitioning task the sector assignment to the feature subspaces is the genotype and phenotype consists of subdatasets of the resulting feature subspaces and the corresponding information measures.
The genotypes in GAs are traditionally represented by bit vectors. Therefore, the sector assignment to the subspaces, which is already represented by a bit vector, is a suitable genotype for the GA. In addition, small changes of the genotype lead to small changes in the phenotype, which is an intended characteristic for the genotype to phenotype mapping in a GA. Hence, the more bits in the genotype are inverted the more the subdatasets of the phenotype will change.

The fitness function depends on the properties that are required for the feature subspaces. For example, if a feature subspace is searched where the *Synergy* is maximal, the Q-value of one considered feature space can be used as fitness function. Consequently, the GA would search for the sector assignment that leads to the highest Q-value in the considered feature subspace.

Figure 10 shows a flowchart of a GA, where the main elements of a GA are depicted. In GAs the *Recombination* is the dominant part of the population modification, while the *Mutation* is secondary [1]. The blocks that are shown in the flowchart where implemented as follows:

Parameters
The following parameters have to be set before the GA can be started:
T: Maximum number of generations (number of iterations)
$\mu = \lambda$: Number of individuals per generation (size of the population)
b: Number of bits that a genotype has (number of considered sectors in the feature space)
S: Selection pressure $1 \leq S \leq 2$ ($S = 2$: Highest pressure, $S = 1$: No pressure)
$k = n_{point}$: Number of random crossover points while the recombination
p_m: Probability for every bit of a genotype that it is inverted while the mutation
e: Number of best parents that are kept for the next generation ($e > 0$: Elitism)

1.) Generate the initial population
In order to generate the initial population, μ bit vector $\mathbf{x_i}$, $i = 1, ..., \mu$ with a length of b are randomly generated. This population can be represented by a $\mu \times b$ matrix with bits entries that consists of the $\mathbf{x_i}$ vectors.

Note: The number of sectors b should only consider the sectors that contain data samples. The assignment of the empty sectors would not take effect on the *Interaction Information*, but it would unnecessarily increase the degree of freedom. For the case that test samples fall in sectors that were empty in the training data, the empty sectors have to be assigned to a feature subspace. For example all empty

sectors can be assigned to the feature subspace with the more complex model that considers feature interactions. Alternatively, the empty sectors can be assigned to the same feature subspace to that the most of their neighboring sectors are assigned to.

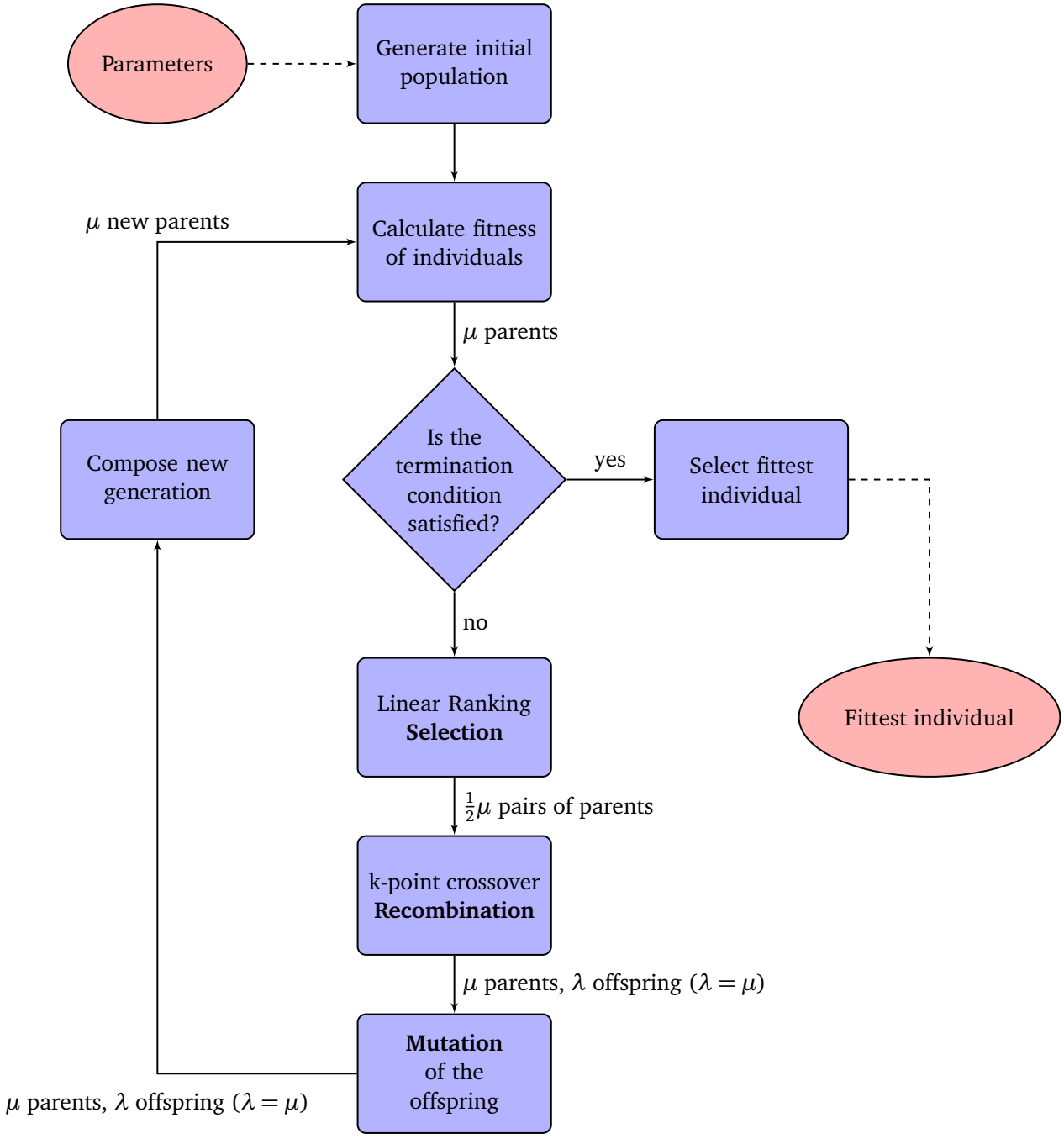

Figure 10: Flowchart of the Genetic Algorithm

2.) Calculate fitness of the individuals

In order to calculate the fitness of a individual i with the genotype $\mathbf{x_i}$, the phenotype has to be created from the individual's genotype. The genotype is our sector assignment and therefore the feature space partitioning. For every sector there is a bit that assigns the sector to one of the two subspaces. The feature subspace 0 consist of all sectors with an assignment bit that is set to 0 and in the same manner the feature subspace 1 consist of all sectors with an assignment bit that is set to 1. If the partitioning

was performed, the original dataset can be divided in the dataset $\mathbf{D_{0,i}}$ of subspace 0 and dataset $\mathbf{D_{1,i}}$ of subspace 1.

The fitness of an individual i is the fitness of the corresponding phenotype that where created from the genotype $\mathbf{x_i}$.
Here are some example fitness functions f, which are supposed to be maximized from generation to generation:

1.) Goal: Maximum *Synergy* in feature subspace 1:

$$f(i) = f(\mathbf{D_{0,i}}, \mathbf{D_{1,i}}, \mathbf{x_i}) = Q(\mathbf{D_{1,i}}) = Q_{1,i} \tag{51}$$

2.) Goal: Maximum *Redundancy* in feature subspace 1:

$$f(i) = f(\mathbf{D_{0,i}}, \mathbf{D_{1,i}}, \mathbf{x_i}) = -Q(\mathbf{D_{1,i}}) = -Q_{1,i} \tag{52}$$

More fitness function examples can be found in the Appendix E on page 121.

3.) Termination condition
If the termination condition is satisfied, the GA is stoped and the best individual of the current generation is returned by the GA. In many cases a good termination condition is that the fitness function achieves a given value in order to guarantee a certain quality of the result.
In the feature space partitioning application, it is not easy to evaluate the potential for improvement i. e. it is not clear whether a certain value can be achieved. Therefore, the termination condition is that the maximum number of generations T is achieved. Alternatively, a maximum number of generations, where no progress was observed, can be defined.

4.) Linear Ranking Selection
For the *Linear Ranking Selection* [1] the individuals are ranked by their fitness $f(i)$. The individual $i = h$ with the highest fitness value has the rank $r_h = 1$ and the individual $i = l$ with the lowest fitness has the rank $r_l = \mu$. Using the ranks r_i of the individuals $i = 1, ..., \mu$, the selection probability of a individual p_i can be calculated with the following function:

$$p_i = \frac{1}{\mu} \cdot \left(2 - S + 2 \cdot (S-1) \cdot \frac{\mu - r_i}{\mu - 1}\right),\ 1 \leq S \leq 2,\ i = 1, ..., \mu \tag{53}$$

$$p_i|_{S=2} = \frac{2}{\mu} \cdot \frac{\mu - r_i}{\mu - 1} \tag{54}$$

$$p_i|_{S=1} = \frac{1}{\mu} \tag{55}$$

where S is the selection pressure. p_i increases linear with r_i. If the selection pressure is high ($S = 2$) the selection probability p_i of the individual with the lowest fitness is zero, while the p_i of the individual with the highest fitness is twice as high as the p_i of the middle ranked individual. If the selection pressure is low ($S = 1$) the p_i of all individuals is the same. For the middle ranked individual $p_i = \frac{1}{\mu}$ holds true for every S (see Figure 11).

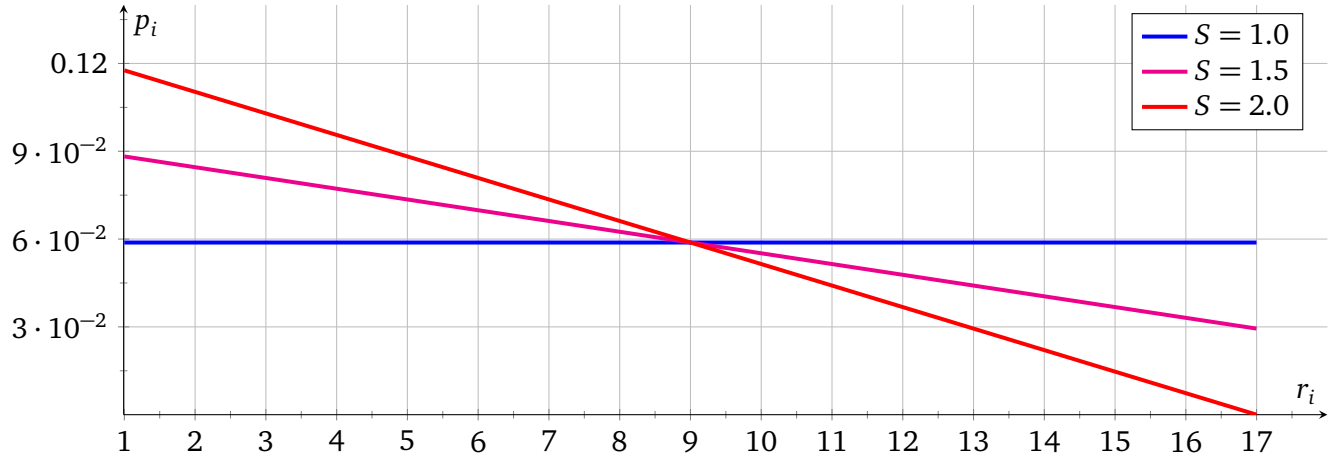

Figure 11: Graph of the selection probability p_i from equation 53 for $\mu = 17$ and different values of S

In order to prepare the recombination of the individuals, $\frac{1}{2}\mu$ pairs of individuals are selected as parents for the new generation.
The can be done by filling a list with μ selected individuals, where the selection probability of an individual i is p_i. Hence, the list of selected individuals can contain duplicates and some of the previous individuals can be missing. The expected average fitness in the list of selected individuals is higher than the average fitness in the current generation, if $S > 1$.
The $\frac{1}{2}\mu$ pairs of parents are created by assigning to every second individual the previous individual in the list.

5.) k-point Crossover Recombination
Every pair of parents produces two offspring by using the *k-point Crossover Recombination*, which mixes the bit vectors (genotypes) with the length b of the parents. For every pair k random values $z_1, ..., z_k$ are picked out of the pool $1, 2, ..., b$ in "drawing without replacement" manner. The values are sorted from the smallest z_1 to the highest z_k. The bit vectors are segmented, using the values $z_1, ..., z_k$. Every z_j contains the bit number of a bit which is the last bit of a segment of the bit vector. Hence, the bit vectors of both parents is divided in k segments. The k-point crossover is performed by exchanging every second segment of the parents in order to create two new bit vectors of the offspring.

Example:
Given:
$b = 8$, $k = 2$: $z_1 = 2$, $z_2 = 6$
Parent 1: 11011010
Parent 2: 10101010
Crossover:
Parent 1: 11|**0110**|10
Parent 2: 10|**1010**|10
Descendant 1: 11|**1010**|10
Descendant 2: 10|**0110**|10

If $\frac{1}{2}\mu$ recombinations were performed, μ new offspring are created.

6.) Mutation of the offspring
The mutation is secondary in a GA [1]. Hence, the change of the population should be dominated by the recombination, not by the mutation. The mutation can be performed by inverting each bit of ev-

ery descendant with a probability of p_m, which is a parameter of the GA that should be chosen very small.

7.) Compose the new generation
The parameter e contains the number of best ranked parents that are kept for the next generation. Because the new generation should consist of μ individuals, there is only the space for the $(\mu - e)$ best ranked offspring left. If $e = 0$ all parents are replaced by all offspring.
The case $e > 0$ is called elitism, which is used to guarantee that the best individual, that was created in the present generations, can not be lost. In order to use elitism, the fitness value of all new μ offspring, as already was shown in *2.) Calculate fitness of the individuals*, has to be calculated to identify the $(\mu - e)$ fittest offspring. The elitism leads to a convergence of the optimization, but it has the disadvantage that the optimization process is dominated by a few individuals. The consequence is a higher risk of a fast convergence into a local fitness maximum [1]. At the end the new generation with μ individuals consists of e parents from the last generation and $(\mu - e)$ offspring from the last generation.

Advantages of the Genetic Algorithm for feature space partitioning:
1.) The optimization criterion can easily be changed by changing the fitness function.
2.) The complexity of the optimization can be scaled by the number of used sectors.
3.) Recombination of approved results instead of random mutation (see old method in Appendix D).
4.) Elitism is optional.
5.) Competition between several parallel results.
6.) Slight mutations can help to leave local optima.

Disadvantages of the Genetic Algorithm for feature space partitioning:
1.) Due to the random elements of the algorithm, the results can differ from run to run.
2.) Compared to methods 2 and 3 the required computation time is enormous ($T \cdot \mu \cdot N$ calculations of the *Interaction Information* are required).

In this section the goal is to find a solution for the feature space partitioning using a mathematical approach that has no random components.
In the previous descriptions the feature space partitioning can be represented by a decomposition of the whole feature space in sectors and an assignment of the sectors into two feature subspaces.
Therefore, the assignments to the feature subspaces are binary that means a sector is either completely contained in a feature subspace or completely excluded.
Consequently, there is no gradient of the *Interaction Information* $Q(X;Y;Z)$ because the parameters of the assignment have only two states.
The information in every data sample that is not assigned to the considered feature subspace is completely lost, hence there is no information that allows to measure how the data sample would influence the *Interaction Information* if it was included in the feature subspace.
Apart from that, it is not possible to measure how the *Interaction Information* would change, if some of the data samples in the considered feature subspace were removed from the feature subspace.

Original problem
The given data consist of h data samples that are lying distributed in the entire feature space. The goal is to find a feature subspace, where the subset of data samples are lying that lead to a maximum *Interaction Information* Q in the feature subspace.
The entire feature space is decomposed in sectors of equal size and all data samples that are lying in one sector are assigned at once to the feature space if the sector is assigned to the feature subspace. The use of sectors reduces the number of data sample assignment combinations.
For every data sample there is an assignment value $w_l, l = 1,...,h$. The feature subspace 1, contains all data samples $\boldsymbol{s}_l$ which have the assignment value $w_l = 1, l = 1,...,h$, while the feature subspace 0 contains all samples $\boldsymbol{s}_l$ which have the assignment value $w_l = 0, l = 1,...,h$. The values $w_l, l = 1,..,h$ have to be equal if they are lying in the same sector.

Relaxation
A relaxation of the conditions in the optimization problem allows to transfer the NP-hard problem into a problem that can be solved in polynomial time. The relaxed problem can be solved using linear optimization techniques. The kind of relaxation that is used in the following descriptions is inspired by the "'Linear programming relaxation'" [25] and the "'Lagrangian relaxation'" [13].

The constraint that the assignment values $w_l, l = 1,..,h$ have to be 0 or 1 is relaxed. Hence, $w_l, l = 1,..,h$ can take all numbers between 0 and 1. The *Interaction Information* $Q(\widetilde{X};\widetilde{Y};\widetilde{Z})$ with weighted frequency of occurrence of the data samples using a weighting vector $\boldsymbol{w}$ can be determined as follows

$$w_l \in \{0,1\} \textit{ (without relaxation)}, \tag{56}$$
$$0 \leq w_l \leq 1 \textit{ (with relaxation)}, \tag{57}$$
$$l = 1,..,h.$$

Thus, $w_l = 0.7$ (for example) is an allowed case and means that sector l is 70% part of the feature subspace 1 and 30% part of the feature subspace 0.
The linear programming relaxation allows to use common optimization techniques like the *gradient descent algorithm*.
As well as in the original problem the values $w_l, l = 1,..,h$ have to be equal if they are lying in the same sector.

In order to obtain a realizable partitioning, the constraint from equation (56) has to be satisfied again at the final result.
The easiest way to bring the constraint back into the result is the following quantization:

$$\text{If } 0 \leq w_l \leq 0.5 \text{ set } w_l = 0, \tag{58}$$
$$\text{If } 0.5 \leq w_l \leq 1 \text{ set } w_l = 1, \tag{59}$$
$$\forall l = 1, .., h.$$

However, the more the values w_l, $l = 1, .., h$ are close to 0.5 the more the quantized final result differs from the optimized result before.
In order to push the values close to 0 or 1 while the optimization process, the constraint is gradual included into the later optimization steps using a penalty function $f_p(\boldsymbol{w})$.
The value of the penalty function increases the more the constraints are violated.

$$f_p(\boldsymbol{w}) = \frac{2^c}{h} \sum_{l=1}^{h} (\min((1 - w_l), w_1))^c, \; 1 \leq c \leq 2, \tag{60}$$
$$0 \leq f_p \leq 1. \tag{61}$$

If $c > 1$ is chosen, the constraint violations are exponentially weighted.

If the *Interaction Information* $Q(\widetilde{X}; \widetilde{Y}; \widetilde{Z})$ in the feature subspace 1 has to be maximized, the cost function $C(\boldsymbol{w}, t)$ with included penalty function $f_p(\boldsymbol{w})$ can be written as follows:

$$C(\boldsymbol{w}, t) = -Q(\widetilde{X}; \widetilde{Y}; \widetilde{Z}) \cdot (1 - r(t) \cdot f_p(\boldsymbol{w})), \tag{62}$$
$$r(t) = \left(\frac{t}{T}\right)^{c_r}, \tag{63}$$
$$1 \leq c_r \leq 10, \tag{64}$$

where t is the current optimization step and T the maximum number of optimization steps. The higher the value c_r the later the constraint takes a significant effect in the optimization process.

Due to the penalty function $f_p(\boldsymbol{w})$ minimization while the optimization, the final quantization (see (58) and (59)) will not change the result as much as before. Better final results can be expected if already the optimization itself is able to take the constraints into account, before the quantization is applied.

***Interaction Information* with weighted frequency of occurrence of the data samples**
In order to calculate the *Interaction Information* $Q(\widetilde{X}; \widetilde{Y}; \widetilde{Z})$ with weighted frequency of occurrence of the data samples using a weighting vector $\boldsymbol{w}$, the probabilities in the following formular are dependent from the weighting vector $\boldsymbol{w}$.
Note: All components that are marked with a tilde ($\widetilde{...}$), are changing due to the consideration of the weighting vector $\boldsymbol{w}$.

$$\widetilde{Q} = Q(\widetilde{X}; \widetilde{Y}; \widetilde{Z}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{q} \widetilde{a}_{i,j,k}, \tag{65}$$
$$\widetilde{a}_{i,j,k} = \widetilde{p}_{i,j,k} \left(\log_2(\widetilde{p}_{i,j,k}) - \log_2(\widetilde{p}_{i,j}) - \log_2(\widetilde{p}_{i,k}) - \log_2(\widetilde{p}_{j,k}) + \log_2(\widetilde{p}_i) + \log_2(\widetilde{p}_j) + \log_2(\widetilde{p}_k) \right) \tag{66}$$

The weighting vector consists of h weights because h samples are given.

$$\boldsymbol{w} = (w_1, w_2, ..., w_h)^T, \;\; 0 \leq w_\nu \leq 1 \; \forall \; \nu = 1, ..., h. \tag{67}$$

Table 1 shows for every weight the corresponding data sample that is weighted in its frequency of occurrence.

Table 1: Data samples with weighted frequency of occurrence

Weighted data samples			
Frequency	Data samples		
w_1	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
w_2	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_l	$s_{l,1}$	$s_{l,2}$	$s_{l,3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_h	$s_{h,1}$	$s_{h,2}$	$s_{h,3}$

The following functions are designed to allow a consideration of the data samples in the calculation of the probabilities that are required to calculate the *Interaction Information* $Q(\widetilde{X}; \widetilde{Y}; \widetilde{Z})$.

$$\delta(x) = \begin{cases} 0, \; x \neq 0 \\ 1, \; x = 0 \end{cases} \tag{68}$$

$$d_{l,(i,j,k)} = \delta(s_{l,1} - x_i) \cdot \delta(s_{l,2} - y_j) \cdot \delta(s_{l,3} - z_k) \tag{69}$$
$$d_{l,(i,j)} = \delta(s_{l,1} - x_i) \cdot \delta(s_{l,2} - y_j) \tag{70}$$
$$d_{l,(i,k)} = \delta(s_{l,1} - x_i) \cdot \delta(s_{l,3} - z_k) \tag{71}$$
$$d_{l,(j,k)} = \delta(s_{l,2} - y_j) \cdot \delta(s_{l,3} - z_k) \tag{72}$$
$$d_{l,(i)} = \delta(s_{l,1} - x_i) \tag{73}$$
$$d_{l,(j)} = \delta(s_{l,2} - y_j) \tag{74}$$
$$d_{l,(k)} = \delta(s_{l,3} - z_k) \tag{75}$$
$$\forall \; i = 1, ..., n, \; j = 1, ..., m, \; k = 1, ..., q, l = 1, ..., h$$

$$\widetilde{p}_{i,j,k}(w_1, ..., w_h) = \frac{1}{S_w} \sum_{\nu=1}^{h} w_\nu \cdot d_{\nu,(i,j,k)} \tag{76}$$

$$\widetilde{p}_{i,j}(w_1, ..., w_h) = \frac{1}{S_w} \sum_{\nu=1}^{h} w_\nu \cdot d_{\nu,(i,j)} \tag{77}$$

$$\widetilde{p}_{i,k}(w_1, ..., w_h) = \frac{1}{S_w} \sum_{\nu=1}^{h} w_\nu \cdot d_{\nu,(i,k)} \tag{78}$$

$$\widetilde{p}_{j,k}(w_1, ..., w_h) = \frac{1}{S_w} \sum_{\nu=1}^{h} w_\nu \cdot d_{\nu,(j,k)} \tag{79}$$

$$\widetilde{p}_i(w_1,...,w_h) = \frac{1}{S_w}\sum_{v=1}^{h} w_v \cdot d_{v,(i)} \tag{80}$$

$$\widetilde{p}_j(w_1,...,w_h) = \frac{1}{S_w}\sum_{v=1}^{h} w_v \cdot d_{v,(j)} \tag{81}$$

$$\widetilde{p}_k(w_1,...,w_h) = \frac{1}{S_w}\sum_{v=1}^{h} w_v \cdot d_{v,(k)} \tag{82}$$

$$S_w = \sum_{v=1}^{h} w_v \tag{83}$$

$$\forall\ i = 1,...,n,\ j = 1,...,m,\ k = 1,...,q, l = 1,...,h$$

Relaxed sector assignments $\hat{\boldsymbol{w}}$

A relaxed sector assignment means an identical, relaxed assignment of all samples that are contained in the same sector of the feature space. Hence, a new weighting vector $\hat{\boldsymbol{w}}$ is introduced to describe the sector assignments.

$$\boldsymbol{w} = (w_1,...,w_h) \text{ (sample assignments)}, \tag{84}$$

$$\hat{\boldsymbol{w}} = (\hat{w}_1,...,\hat{w}_N) \text{ (sector assignments)}, \tag{85}$$

$$N \leq h, \tag{86}$$

$$w_l = \hat{w}_{s(l)}, \tag{87}$$

$$l = 1,...,h,$$

where $s(l)$ is the sector which contains the sample l in the feature space. The greater the size of the sectors the smaller is the number of sectors and therewith the number of weights in $\hat{\boldsymbol{w}}$ compared to the number of weights in $\boldsymbol{w}$. If sectors are used, the weighting vectors $\hat{\boldsymbol{w}}$ and $\boldsymbol{w}$ can be transformed into each other ($\hat{\boldsymbol{w}} \Leftrightarrow \boldsymbol{w}$). They contain the same information if the sectors are already defined. Thus, all functions that are dependent from $\boldsymbol{w}$ can also be written in dependence from $\hat{\boldsymbol{w}}$.
There are three main reasons why sector assignments are used instead of sample assignments:

1) Samples that are at the same location in the quantized feature space are summarized and share one assignment weight. Which is reasonable because a feature space partitioning is not able to separate samples which are lying at the same location in the feature space.
2) The complexity of the assignment problem is reduced due to a smaller number of assignment weights.
3) If the feature space partitioning and training of the local models was performed, a prediction on test samples that did not occur in the training can be desired. These test samples can be assigned to a feature subspace if they are lying in an assigned sector.

3.2.2 Method 2: Gradient descent algorithm using relaxed sector assignments

The *Interaction Information* can be derived by the frequency of occurrence of every data sample. The complete and exact calculation of

$$\nabla Q(\widetilde{X};\widetilde{Y};\widetilde{Z}) = \begin{pmatrix} \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_1} \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_2} \\ \vdots \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_h} \end{pmatrix} \tag{88}$$

can be found in the appendix C.

Alternatively, an approximation of the derivation can be calculated using the *Difference Quotient*.

$$\frac{df(x)}{dx} \approx \frac{f(x+\Delta x)-f(x)}{\Delta x} \tag{89}$$

The *Difference Quotient* can be used to approximate the following derivations:

$$\nabla Q(\widetilde{X};\widetilde{Y};\widetilde{Z}) = \begin{pmatrix} \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_1} \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_2} \\ \vdots \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_h} \end{pmatrix} \tag{90}$$

$$\frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_l} \approx \frac{Q_{\Delta w,l} - Q_0}{\Delta w} \tag{91}$$

$$Q_{\Delta w,l} = Q(\widetilde{X};\widetilde{Y};\widetilde{Z})|_{\boldsymbol{w}=(w_1,.,w_l+\Delta w,..,w_h)^T} \tag{92}$$

$$Q_0 = Q(\widetilde{X};\widetilde{Y};\widetilde{Z})|_{\boldsymbol{w}=(w_1,...,w_h)^T} \tag{93}$$

$$\Delta w = 0.01 \tag{94}$$

$$l = 1,...,h$$

As shown in Figure 12 the *Interaction Information* curves are not jumping if the weights $w_l,\ h = 1,...,h$ are changing in the interval [0,1] and the curves are smooth. Therefore, a very good approximation can be expected using $\Delta w = 0.01 << 1$.
The main advantages of using the *Difference Quotient* are the easy implementation and the possibility to replace the function that has to be derived. For example, if a subspace with a *Non-Interaction* is searched the *Interaction Information* has to be combined with *Mutual Information* functions. Using the *Difference Quotient* the exact analytical derivation of a function is not required.

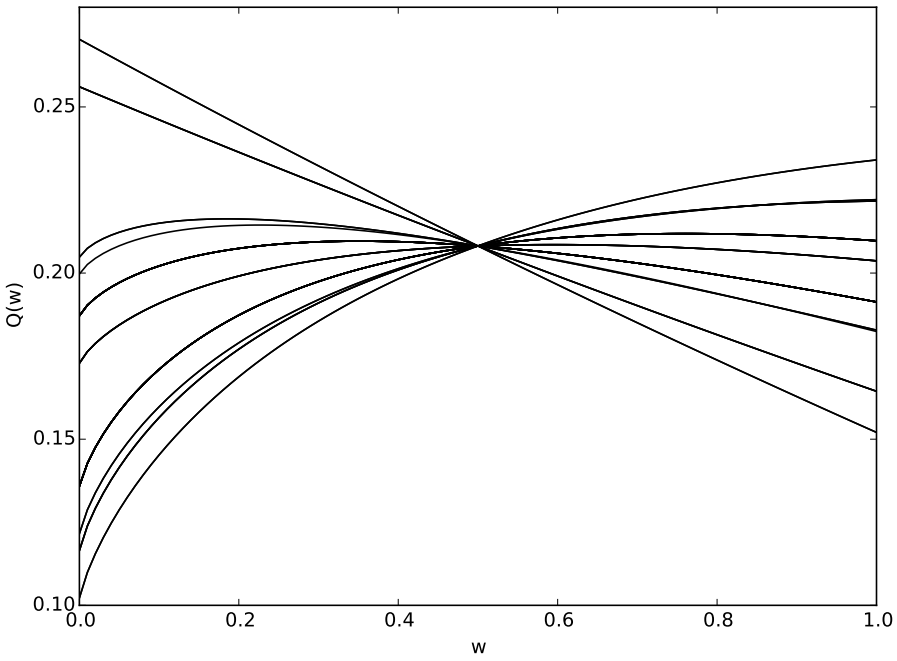

Figure 12: *Interaction Information* $Q(\widetilde{X};\widetilde{Y};\widetilde{Z})$ of Dataset 1, where one value from the weighting vector $\boldsymbol{w}$ is changed. Every curve stands for one weight w_l, $l = 1,...,h$ that changes, while the rest of the weights is constantly equal to 0.5. ***Note****: Some of the curves are identical.*

Derivation by relaxed sector assignments $\hat{w}$

$$\nabla Q(\widetilde{X};\widetilde{Y};\widetilde{Z}) = \begin{pmatrix} \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial \hat{w}_1} \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial \hat{w}_2} \\ \vdots \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial \hat{w}_N} \end{pmatrix} \tag{95}$$

$$\frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial \hat{w}_s} \approx \frac{Q_{\Delta w,s} - Q_0}{\Delta w} \tag{96}$$

$$Q_{\Delta w,s} = Q(\widetilde{X};\widetilde{Y};\widetilde{Z})|_{\hat{\boldsymbol{w}}=(\hat{w}_1,.,\hat{w}_s+\Delta w,..,\hat{w}_N)^T}, \tag{97}$$

$$Q_0 = Q(\widetilde{X};\widetilde{Y};\widetilde{Z})|_{\hat{\boldsymbol{w}}=(\hat{w}_1,...,\hat{w}_N)^T} \tag{98}$$

$$\Delta w = 0.01, \tag{99}$$

$$s = 1,...,N,$$

where N is the number of sectors. And $\hat{w}_s + \Delta w$ means that all weights $w_l, l = 1,...,h$ that are lying in sector s are increased by Δw (see equation (97)).

Derivation of the cost function

If the goal is to find a feature subspace with the maximum *Interaction Information*, the following cost

function is used:

$$C(\hat{\boldsymbol{w}}, t) = -Q(\widetilde{X}; \widetilde{Y}; \widetilde{Z}) \cdot (1 - r(t) \cdot f_p(\hat{\boldsymbol{w}})) \tag{100}$$

$$\nabla C(\hat{\boldsymbol{w}}, t) = \begin{pmatrix} \frac{\partial C(\hat{\boldsymbol{w}}, t)}{\partial \hat{w}_1} \\ \frac{\partial C(\hat{\boldsymbol{w}}, t)}{\partial \hat{w}_2} \\ \vdots \\ \frac{\partial C(\hat{\boldsymbol{w}})}{\partial \hat{w}_N} \end{pmatrix} \tag{101}$$

$$\frac{\partial C(\hat{\boldsymbol{w}})}{\partial \hat{w}_s} \approx \frac{C_{\Delta w,s} - C_0}{\Delta w} \tag{102}$$

$$C_{\Delta w,s} = C(\hat{\boldsymbol{w}}, t)|_{\hat{\boldsymbol{w}}=(\hat{w}_1,..,\hat{w}_s+\Delta w,..,\hat{w}_N)^T} \tag{103}$$

$$C_0 = C(\hat{\boldsymbol{w}}, t)|_{\hat{\boldsymbol{w}}=(\hat{w}_1,...,\hat{w}_N)^T} \tag{104}$$

$$\Delta w = 0.01 \tag{105}$$

$$s = 1, ..., N$$

If the goal is to find a feature subspace with the minimum *Interaction Information*, the cost function can be replaced:

$$C(\hat{\boldsymbol{w}}, t) = Q(\widetilde{X}; \widetilde{Y}; \widetilde{Z}) \cdot (1 - r(t) \cdot f_p(\hat{\boldsymbol{w}})) \tag{106}$$

Optimization process
The initialization starts with balanced assignment to both feature subspaces.

$$\hat{\boldsymbol{w}}(t = 0) = (0.5, ..., 0.5)^T \tag{107}$$

For a given number of iterations the following optimization step is performed:

$$\hat{\boldsymbol{w}}(t + 1) = \hat{\boldsymbol{w}}(t) - \gamma \cdot sgn(\nabla C(\hat{\boldsymbol{w}}, t)) \tag{108}$$

$$sgn(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \tag{109}$$

At the end of every step the values of $\hat{w}$ that are not lying in the interval [0,1] are moved to their nearest boundary (0 or 1).
If the values are already equal to 0 or 1 before the optimization step, the entries of $\nabla C(\hat{\boldsymbol{w}}, t)$ that would push them out of the interval [0,1] are set to zero.
The value γ is the step width of the optimization step.

At the end the quantization step is performed (see equations (58) and (59)).

Note: The optimization step can also be done without a sgn-function, however while performing tests the version of the method with a sgn-function provided better results.
Using the sgn-function is a common method to avoid the influence of the gradient on the width of the optimization steps [1].
The interval where the values of $\hat{\boldsymbol{w}}$ are moving is [0,1], hence a step width of $\gamma = 0.1$ seems to be a very high value, because it is 10% of the entire space. However, at the end (in quantization step) it is only important whether a value lies above or below the threshold 0.5, therefore, no finer steps are required.

3.2.3 Method 3: Interaction Information Map Reinforcement using relaxed sector assignments

Method 3 is very similar to method 2. The difference is that the gradient

$$\nabla Q(\widetilde{X};\widetilde{Y};\widetilde{Z}) = \begin{pmatrix} \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial \hat{w}_1} \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial \hat{w}_2} \\ \vdots \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial \hat{w}_N} \end{pmatrix} \tag{110}$$

$$\tag{111}$$

in method 2 is replaced by the sum of the local Interaction Information Map components $\widetilde{q}_z(x,y)$ that are lying in the same sector (see equation (50)).

$$\widetilde{q}_s = \sum_x \sum_y \delta_s(x,y) \cdot \widetilde{q}_z(x,y), \tag{112}$$

$$x \in \{x_1,...,x_n\}, y \in \{y_1,...,y_m\},$$

$$\delta_s(x,y) = \begin{cases} 1, & (x,y) \text{ lies in sector s,} \\ 0, & \text{otherwise} \end{cases}, \tag{113}$$

$$\forall\, s = 1,...,N,$$

$$\widetilde{\mathbf{q}}_{map} = (\widetilde{q}_1,...,\widetilde{q}_N)^T, \tag{114}$$

where $\widetilde{q}_1,...,\widetilde{q}_N$ are the influences of the sectors on the *Interaction Information* for the current vector $\hat{\boldsymbol{w}}(t)$.

Optimization step of method 3:

$$\hat{\boldsymbol{w}}(t+1) = \hat{\boldsymbol{w}}(t) - \gamma \cdot \widetilde{\mathbf{q}}_{map} \tag{115}$$

As well as in method 2, the idea of this method is to reinforce the sectors that have on average a positive influence on the *Interaction Information*, while the rest of the sectors is gradually removed from the feature subspace 1.
It is important to perform the map reinforcement iteratively because the map can change due to changing $\hat{\boldsymbol{w}}(t)$. The map describes the influence on the *Interaction Information* of every sector for a particular state of the vector $\hat{\boldsymbol{w}}(t)$.
If the goal is to find a feature subspace with a minimum *Interaction Information*, $\widetilde{\mathbf{q}}_{map}$ can be replaced by $(-\widetilde{\mathbf{q}}_{map})$.
Note: This method does not include a penalty function.

Optional improvement 1: $\hat{\boldsymbol{w}}(t)$ normalization
After the optimization step 115 an additional normalization step on $\hat{\boldsymbol{w}}(t)$ can be performed:

$$\hat{w}_{s,norm} = \frac{\hat{w}_s - \hat{w}_{min} + \hat{w}_{norm,min}}{\hat{w}_{max} - \hat{w}_{min} + w_{norm,min}} \tag{116}$$

$$\hat{w}_{min} = \min\{\hat{w}_1,...,\hat{w}_N\} \tag{117}$$

$$\hat{w}_{max} = \max\{\hat{w}_1,...,\hat{w}_N\} \tag{118}$$

$$w_{norm,min} = 0.01 \tag{119}$$

$$\forall\, s = 1,...,N$$

$$\hat{\boldsymbol{w}}_{norm} = (\hat{w}_{1,norm},...,\hat{w}_{N,norm})^T \tag{120}$$

After the normalization the value distribution of $\hat{\boldsymbol{w}}(t+1)$ is stretched or compressed to fit into the interval $[w_{norm,min}, 1]$. Instead of limiting $\hat{w}_s$ to 1, while the optimization step, higher values are allowed. Hence, a sector which already has an assignment value $\hat{w}_s$=1 is able to become even more dominant in relation to the other sectors by exceeding the value 1.
Then the normalization brings the values back in the range $[w_{norm,min},1]$ by keeping a similar relation between the values. For the value of the *Interaction Information* only the relative relation of the values in $\hat{\boldsymbol{w}}$ is important.
The small value $0 < \hat{w}_{s,norm} << 1$ avoids that sectors are completely removed from the competition. If a sector had the value $\hat{w}_s = 0$, it would not be considered for the *Interaction Information Map*, therefore, the influence of the sector in the current state could not be considered.

Note: The normalization step led to much better optimization results while the partitioning experiments. Therefore, this improvement is recommended. The computation cost is negligible.

Optional improvement 2: Search the best threshold for the quantization
Because no penalty function is used in method 3, the vector $\hat{\boldsymbol{w}}$ will also contain values near to 0.5 at the end of the optimization process.

Therefore, small changes of $\hat{\boldsymbol{w}}$ can change the assignments of sectors because the quantization step replaces values higher than 0.5 by 1 and the rest of the values by 0.
Because method 3 sometimes tends to take less sectors for the feature subspace 1 than reasonable, the threshold 0.5 is not always the best choice. A smaller threshold would be an option. A saver solution is to test a set of n_{thr} thresholds and calculate the *Interaction Information* for every threshold. At the end the threshold with the best result is kept.
The thresholds $\tau_\xi, \xi = 1, ..., n_{thr}$ can be distributed equidistant over the interval [0,1]:

$$\tau_\xi = \frac{\xi}{n_{thr}+1} \tag{121}$$

$$\forall\ \xi = 1, ..., n_{thr} \tag{122}$$

Those threshold distributions that are more dense near 0.5 could be more sophisticated, however the equidistant distribution works well.
The threshold search decreases the importance of a fine parameter tuning.
The search for the best threshold led to better optimization results while the partitioning experiments. Therefore, this improvement is recommended. It costs n_{thr} extra calculations of the *Interaction Information*.

Advantages of method 2 and 3:
1.) The optimization criterion can easily be changed by changing the cost function.
2.) The complexity of the optimization can be scaled by the number of used sectors.
3.) Relaxation of the weights allows the consideration of all sectors at every optimization step and the use of more optimization techniques (e.g. gradient descent).
4.) The approaches will always provide the same feature space partitioning for the same data. No random components are included. Which makes it easier to evaluate the partitioning capability.
5.) The optimization is very fast compared to method 1.

Advantage of method 2 over 3:
1.) A penalty function can be used.

Advantage of method 3 over 2:
1.) Threshold search can be used.

2.) Faster, only $T + n_{thr}$ instead of $T \cdot (N + 1)$ calculations of the *Interaction Information* are required.

Note: T is the number of iterations.

Disadvantages of method 2 and 3:
1.) The optimization is not performed under the real constraints, therefore a quantization of the result in the final step is required, which potentially degrade the optimized score.
2.) Due to a loss of random components, the provided result will always be the same local optimum, which is possibly not the global optimum.

3.3 Classifiers and Regressors

Classifiers and Regressors are the models which are trained using training data. They are used to predict an output value for a given input value combination. The most important property of a suitable model is its generalization capability. If a model has a good generalization capability, it is able to predict reasonable output values for a new test input value combination that was not contained in the training data. Hence, a model should be able to recognize relations between the variables in the data instead of simply memorize the examples in the training data. If the relevant relations in the training data are less complex than the model, the model starts to learn irrelevant relations e.g. relation exceptions that are not correct in general. The learning of irrelevant relations from the training data that are not transmittable to the test data is called *overfitting*. An *overfitted* model has a poor generalization capability. Hence, if *overfitting* can be prevented by using simpler models, while the models are complex enough to learn the important relations, a simple model should be to preferred.

A classifier is designed to predict one of a given set of classes as output value, while a regressor is able to predict continuous numbers as output values.

3.3.1 Bayesian Classification

Bayesian Classifiers are used to classify unseen observations to one of the probable class category (also called class labels) [14].

That means that a model of a *Bayesian Classifier*, which was trained with Data (observations from a system), can calculate the probability of every output value for a given combination of input values. The output value with the highest probability is chosen as actual prediction of the output value *(probabilistic inference)*. The accuracy of the calculated probability distribution of the output value (or class) given the input values is higher the more the assumptions that were made in the chosen model hold true in the given data.

The simplest model is the model of the *Naive Bayesian Classifier*, where the number k of dependent features given the class is assumed to be zero ($k = 0$). In contrast the model of the *Full Bayesian Classifier* is a very complex one because a conditional dependence of all features given the class is assumed ($k = N - 1$). To cover the complete *feature dependence spectrum* ($k = 0, ..., N - 1$) of considered dependencies, the *Limited Dependence Classifier* [20] was developed, where k is one of the parameters that is given to the model, before the training.
The dependencies assumed in a model can be represented by a *Bayesian Network* which is a type of a *Probabilistic Graphical Model*, where the dependent variables are connected (e.g. as shown in Figure 13).

The set of output values of a *Bayesian Classification* model is restricted to the set of output values (classes) that already occurred in the training data because all variables are treated as *Categorical Variables* (see Section 2.3), therefore, no similarity of values is considered and no regression is possible.
Because of the deterministic training of the model, there is no random component in the prediction. Hence, the evaluation of different data subsets (feature subspaces) can be compared in a deterministic way.

3.3.2 Bayesian Networks

A *Bayesian Network* is a type of a *Probabilistic Graphical Model*, which can be used to visualize (conditional) dependencies in an underlying model of a certain *Bayesian Classifier*. In a *Bayesian Network* the variables are represented by nodes and the (conditional) dependencies between the variables are represented by arrows. An arrow represents a dependence and a condition.
For example, the *Bayesian Network* could consider three features X_1, X_2, X_3 and an output variable C. Then the joint probability of all variables $P(C, X_1, X_2, X_3)$ can be described using the *chain rule of conditional probability*.

$$P(C, X_1, X_2, X_3) = P(C) \cdot P(X_1|C) \cdot P(X_2|X_1, C) \cdot P(X_3|X_1, X_2, C) \tag{123}$$

$$P(C, X_1, X_2, X_3) = P(C) \cdot \frac{P(X_1, C)}{P(C)} \cdot \frac{P(X_1, X_2, C)}{P(X_1, C)} \cdot \frac{P(X_1, X_2, X_3, C)}{P(X_1, X_2, C)} \tag{124}$$

The corresponding *Bayesian Network* is shown in Figure 13.

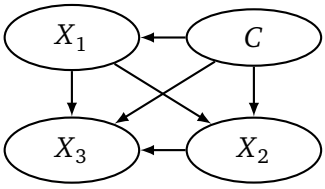

Figure 13: *Bayesian Network* example, where no dependency between X_1, X_2, X_3 and C is excluded (*Full Bayesian Network*).

The dependencies of a arbitrary variable X_A to other variables in the *Bayesian Network* is considered under the condition that the values of the variables that are pointing on the variable X_A, are already known.
If the variables in the equation are exchanged, the arrow directions in the *Bayesian Network* are changing (see Figure 14).

$$P(C, X_1, X_2, X_3) = P(X_2) \cdot P(X_1|X_2) \cdot P(C|X_1, X_2) \cdot P(X_3|X_1, X_2, C) \tag{125}$$

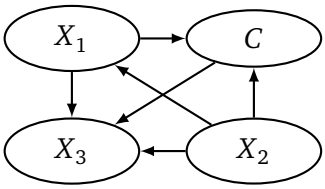

Figure 14: *Bayesian Network* example, where no dependency between X_1, X_2, X_3 and C is excluded.

Hence, the joint probability $P(C, X_1, X_2, X_3)$ can be calculated correctly for different considered conditional probabilities as shown in Figures 13 and 14 and equations (123) and (125).

The different simplified models of the *Bayesian Classifiers* are based on removals of arrows due to assumptions about conditional independencies of certain variables.

Example for conditional independence
Two persons are deciding to go together to one of three of their favorit restaurants, where the restaurants offer completely different meals. Both persons are ordering a meal respectively, while they do not care what the other one is ordering. In that case the ordered meals of the two persons are not independent

from each other because the meals have to be from the same restaurant. Under the condition that the restaurant is already known, the ordered meals of both persons are independent.

The effect of a conditional independence on the *Bayesian Network* is shown in Figure 15, where X_2 and X_3 are independent if the value of X_1 is known. The equation (123) simplifies to equation (126) because of the conditional independence that causes $P(X_3|X_1,X_2,C) = P(X_3|X_1,C)$.

$$\begin{aligned} P(C,X_1,X_2,X_3) &= P(C)\cdot P(X_1|C)\cdot P(X_2|X_1,C)\cdot P(X_3|X_1,\cancel{X_2},C) \\ &= P(C)\cdot P(X_1|C)\cdot P(X_2|X_1,C)\cdot P(X_3|X_1,C) && (126) \\ &= P(C)\cdot \frac{P(X_1,C)}{P(C)}\cdot \frac{P(X_1,X_2,C)}{P(X_1,C)}\cdot \frac{P(X_1,X_3,C)}{P(X_1,C)} && (127) \\ &= P(X_1,X_2,C)\cdot \frac{P(X_1,X_3,C)}{P(X_1,C)} && (128) \end{aligned}$$

The corresponding *Bayesian Network* is shown in Figure 15.

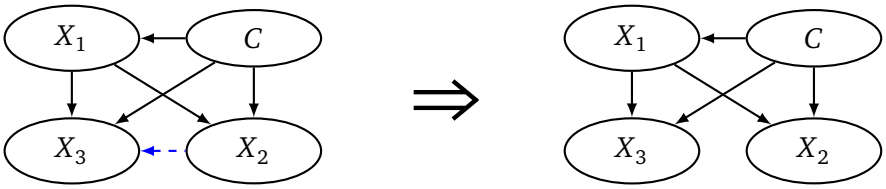

Figure 15: *Bayesian Network* example, where X_2 and X_3 are independent if the values of X_1 and C are known. The blue, dashed arrow in the network on the left side has been removed in the network on the right side due to the conditional independence of X_2 and X_3 given X_1 and C.

The benefit of the simplification from $P(X_3|X_1,X_2,C) = \frac{P(X_1,X_2,X_3,C)}{P(X_1,X_2,C)}$ to $P(X_3|X_1,C) = \frac{P(X_1,X_3,C)}{P(X_1,C)}$ is that the joint probability distribution $P(X_1,X_2,X_3,C)$ has much more value combinations than the joint probability distributions of the resulting replacement $\frac{P(X_1,X_2,C)\cdot P(X_1,X_3,C)}{P(X_1,C)}$ in equation (128) if every random variable can take at least three values. Consequently, the simplification decreases the number of parameters that has to be learned in the model. The training becomes faster. The risk of overfitting is reduced and therefore the number of required observations in the training data is reduced, too.
Hence, the removal of arrows (conditional dependencies) in the *Bayesian Network* is essential in the following *Bayesian Classifiers*.

The exact parameter reduction due to the removal of **only one** arrow in the example above can be calculated as follows:
n_1: Number of values that X_1 can take.
n_2: Number of values that X_2 can take.
n_3: Number of values that X_3 can take.
n_c: Number of values that C can take.
N_f: Number of parameters that are required to calculate $P(C,X_1,X_2,X_3)$ in a *Full Bayesian Network* as shown in Figure 13
N_s: Number of parameters that are required to calculate $P(C,X_1,X_2,X_3) = \frac{P(X_1,X_2,C)\cdot P(X_1,X_3,C)}{P(X_1,C)}$ in a *Bayesian Network* with a *Conditional Independence* as shown in Figure 15

$$N_f = n_1\cdot n_2\cdot n_3\cdot n_c \quad (129)$$
$$N_s = n_1\cdot n_2\cdot n_c + n_1\cdot n_3\cdot n_c + n_1\cdot n_c \quad (130)$$

Examples with $n_1 = n_2 = n_3 = n_c = n$:

$$N_f = n^4 \tag{131}$$

$$N_s = 2 \cdot n^3 + n^2 \tag{132}$$

n	N_f	N_s	$\frac{N_s - N_f}{N_f}$
2	16	20	+25.00%
3	81	63	-22.22%
4	256	144	-43.75%
5	625	275	-56.00%
6	1296	468	-63.89%
7	2401	735	-69.39%
8	4096	1088	-73.44%
9	6561	1539	-76.54%
10	10000	2100	-79.00%

Therefore, if every variable can take $n = 10$ different values, the removal of **only one** arrow reduces the number of required model parameters by 79%.

Conditional Dependence

Not only *Conditional Independence*, but also *Conditional Dependence* can exist. For example, a pure *Synergy* describes a *Conditional Dependence*, where Z is only dependent from Y if X is known (e. g. $Z = X \oplus Y$). The *Interaction Information* can also be written as $Q(X;Y;Z) = I(Y;Z|X) - I(Y;Z)$. Hence, if there is no dependence between X and Y, but a *Conditional Dependence*, a pairwise *Synergy* of X, Y and Z exists.

$$\begin{aligned}
Z &= X \oplus Y \quad \textit{(XOR: Pure Synergy)} \\
H(Z) &= 1bit \quad \textit{(Entropy)} \\
I(Z;Y) &= 0bit \quad \textit{(Independence without condition)} \\
I(Z;Y|X) &= 1bit \quad \textit{(Conditional Dependence)} \\
Q(X;Y;Z) &= I(Y;Z|X) - I(Y;Z) = 1bit - 0bit = 1bit \quad \textit{(Interaction Information} > 0\textit{)} \\
\frac{Q(X;Y;Z)}{H(Z)} &= \frac{1bit}{1bit} = 100\% \quad \textit{(Pure Synergy)}
\end{aligned} \tag{133}$$

This is good to know since those arrows between features which are representing a high conditional mutual information $I(X_i, X_j|C)$ are preferred in the *Limited Dependence Classifier*.

In contrast, a pure *Redundancy* describes a *Conditional Independence*, where Z is only independent from Y if X is known.

$$\begin{aligned}
Z = X = Y &\in \{0,1\} \quad \textit{(Equality: Pure Redundancy)} \\
H(Z) &= 1bit \quad \textit{(Entropy)} \\
I(Y;Z) &= 1bit \quad \textit{(Dependence without condition)} \\
I(Y;Z|X) &= 0bit \quad \textit{(Conditional Independence)} \\
Q(X;Y;Z) &= I(Y;Z|X) - I(Y;Z) = 0bit - 1bit = -1bit \quad \textit{(Interaction Information} < 0\textit{)} \\
\frac{Q(X;Y;Z)}{H(Z)} &= \frac{-1bit}{1bit} = -100\% \quad \textit{(Pure Redundancy)}
\end{aligned} \tag{134}$$

Note: $I(Y;Z|X)$ becomes zero due to the following equations:

$$I(Y;Z|X) = H(Z|X) - H(Z|Y,X) \tag{135}$$
$$H(Z|Y,X) = H(Z|X) = H(Z|Y) \Leftrightarrow X = Y \tag{136}$$
$$I(Y;Z|X) = 0 \Leftrightarrow X = Y$$

3.3.3 Relation between the Conditional Mutual Information and the structure connections of a Bayesian Network

The *Conditional Mutual Information* $I(X_i;X_j|\Pi_{X_i}\backslash X_j)$ of two features X_i and X_j given the parents $\Pi_{X_i}\backslash X_j$ measures the requirement of an arrow from X_j to X_j, where $i,j \in \{1,...,N\}$. If $I(X_i;X_j|\Pi_{X_i}\backslash X_j) = 0$ holds true, an arrow from X_j to X_i can be removed without losing accuracy.

Note: Equivalently an arrow from C to X_i can be removed if $I(X_i;C|\Pi_{X_i}\backslash C) = 0$ holds true. However, the arrows from C to X_i are usually kept.
The set $\Pi_{X_i}\backslash X_j$ is the set of parents of X_i (denoted as Π_{X_i}) without X_j.

In equations (137) to (149) the relation between the *Conditional Mutual Information* and the arrows is deduced for the simple example from Figure 16.

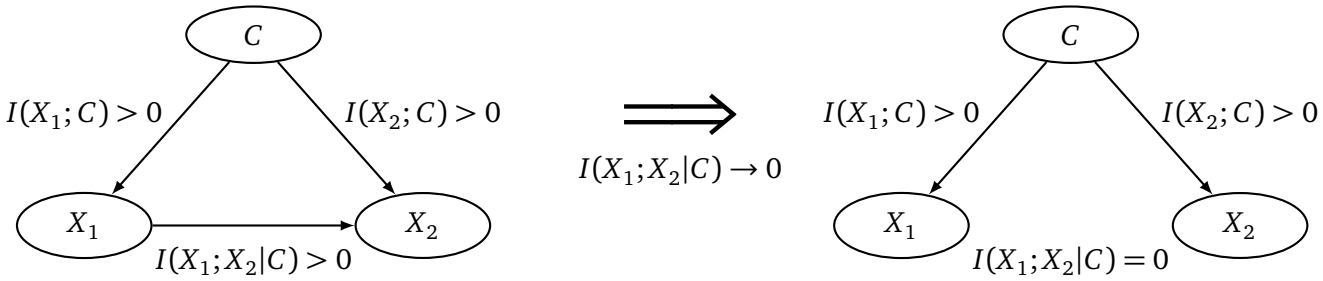

Figure 16: *Full Bayesian Network* with three variables and three information measures that represent the requirement of the arrows.

$$P(X_1,X_2,C) = P(C)\cdot P(X_1|C)\cdot P(X_2|C,X_1) \tag{137}$$
$$P(X_2|C,X_1) = \frac{P(X_1,X_2,C)}{P(C,X_1)} \tag{138}$$
$$= \frac{P(X_1,X_2|C)\cdot P(C)}{P(X_1|C)\cdot P(C)} \tag{139}$$
$$= \frac{P(X_1,X_2|C)}{P(X_1|C)} \tag{140}$$
$$P(X_1,X_2,C) = P(C)\cdot P(X_1|C)\cdot \frac{P(X_1,X_2|C)}{P(X_1|C)} \tag{141}$$

$$I(X_1;C) = 0 \Leftrightarrow P(X_1|C) = P(X_1) \tag{142}$$
$$\Leftrightarrow P(X_1,X_2,C) = P(C)\cdot P(X_1|\cancel{C})\cdot P(X_2|C,X_1) \tag{143}$$

$$I(X_2;C) = 0 \Leftrightarrow P(X_2|C,X_1) = P(X_2|X_1) \tag{144}$$
$$\Leftrightarrow P(X_1,X_2,C) = P(C)\cdot P(X_1|C)\cdot P(X_2|\cancel{C},X_1) \tag{145}$$

$$I(X_1;X_2|C) = 0 \Leftrightarrow P(X_1,X_2|C) = P(X_1|C)\cdot P(X_2|C) \tag{146}$$
$$\Leftrightarrow \frac{P(X_1,X_2|C)}{P(X_1|C)} = \frac{P(X_1|C)\cdot P(X_2|C)}{P(X_1|C)} = P(X_2|C) \tag{147}$$
$$\Leftrightarrow P(X_2|C,X_1) = P(X_2|C) \tag{148}$$
$$\Leftrightarrow P(X_1,X_2,C) = P(C)\cdot P(X_1|C)\cdot P(X_2|C,\cancel{X_1}) \tag{149}$$

Using a more general approach the relation between the *Conditional Mutual Information* and an arrow in a *Bayesian Network* can be determined as follows:

Usually an arrow removal takes effect on the conditional probability distribution of the variable on that the arrow is pointing. If the arrow removal do not take effect on the conditional probability distribution, the removal will not cause a loss of accuracy and a particular *Conditional Mutual Information* is zero.
For example in Figure 15 the removal of an arrow is shown. The corresponding conditional probability distribution $P(X_3|X_1,X_2,C)$ is assumed to be equal to $P(X_3|X_1,C)$. The following equations ((150) to (158)) clarify how the particular *Conditional Mutual Information*, that has to be zero, can be determined.

$$P(X_3|X_1,X_2,C) = P(X_3|X_1,C) \Leftrightarrow \frac{P(X_1,X_2,X_3,C)}{P(X_1,X_2,C)} = \frac{P(X_1,X_3,C)}{P(X_1,C)} \tag{150}$$
$$\Leftrightarrow \frac{P(X_2,X_3|X_1,C)\cdot P(X_1,C)}{P(X_2|X_1,C)\cdot P(X_1,C)} = \frac{P(X_3|X_1,C)\cdot P(X_1,C)}{P(X_1,C)} \tag{151}$$
$$\Leftrightarrow P(X_2,X_3|X_1,C) = P(X_2|X_1,C)\cdot P(X_3|X_1,C) \tag{152}$$
$$\Leftrightarrow I(X_2;X_3|X_1,C) = 0 \tag{153}$$
$$\Leftrightarrow P(C,X_1,X_2,X_3) = P(C)\cdot P(X_1|C)\cdot P(X_2|X_1,C)\cdot P(X_3|X_1,\cancel{X_2},C) \tag{154}$$

This approach can be generalized to an arbitrary arrow that points from X_j to X_i as follows:

$$P(X_i|\Pi_{X_i}) = P(X_i|\Pi_{X_i}\backslash X_j) \Leftrightarrow \frac{P(X_i,\Pi_{X_i})}{P(\Pi_{X_i})} = \frac{P(X_i,\Pi_{X_i}\backslash X_j)}{P(\Pi_{X_i}\backslash X_j)} \tag{155}$$
$$\Leftrightarrow \frac{P(X_i,X_j|\Pi_{X_i}\backslash X_j)\cdot P(\Pi_{X_i}\backslash X_j)}{P(X_j|\Pi_{X_i}\backslash X_j)\cdot P(\Pi_{X_i}\backslash X_j)} = \frac{P(X_i|\Pi_{X_i}\backslash X_j)\cdot P(\Pi_{X_i}\backslash X_j)}{P(\Pi_{X_i}\backslash X_j)} \tag{156}$$
$$\Leftrightarrow P(X_i,X_j|\Pi_{X_i}\backslash X_j) = P(X_i|\Pi_{X_i}\backslash X_j)\cdot P(X_j|\Pi_{X_i}\backslash X_j) \tag{157}$$
$$\Leftrightarrow I(X_i;X_j|\Pi_{X_i}\backslash X_j) = 0 \tag{158}$$

where Π_{X_i} is the set of variables from which arrows point on X_i (before the arrow removal) and $\Pi_{X_i}\backslash X_j$ is the set of variables from which arrows point on X_i after the arrow removal.

Consequently, if $I(X_i;X_j|\Pi_{X_i}\backslash X_j) = 0$ holds true, the arrow from X_j to X_i can be removed without losing accuracy at the calculation of the joint probability distribution $P(X_1,...,X_N,C)$.

3.3.4 Matrix based Bayesian Network Representation

A *Bayesian Network* can be represented by a matrix. Every row of the matrix describes the conditions for one variable. The variables from that arrows are pointing to a certain variable are called parents of the certain variable. The parents of a certain variable are the variables that are already known while

considering the probability of a certain variable. The set of parents of a feature which is later referred to as $\Pi_{X_i}, i = 1, ..., N$ can be determined from the row of X_i. Matrix examples are shown in the Tables 2 to 4.

The first row of the matrix represent the arrows that point from a feature to the class variable C. The *Bayesian Networks* that are used in our classifiers do not contain arrows that point from features to the class variable, hence the first row consists of zeros and there is no Π_C considered. However, to keep the matrix representation more general the first row is kept.
An example for a Bayesian Network where arrows are pointing on C is an alternative description of a *Full Bayesian Network*, where N arrows point from $X_1, ..., X_N$ to C. In that case $a_{0,1}$ to $a_{0,N}$ in the first row would be set to one and the rest of the matrix would consist of zeros. The conditional probability of the class that is described by that *Bayesian Network* is $P(C|X_1, ..., X_N)$ [20].
A further example of a *Full Bayesian Network* where arrows point to C was shown in Figure 14. Therefore this *Full Bayesian Network* can not be described by $\Pi_{X_i}, i = 1, ..., N$ without an additional Π_C.

Arrow	From				
To	C	X_1	X_2	...	X_N
C	0	$a_{0,1}$	$a_{0,2}$	...	$a_{0,N}$
X_1	$a_{1,0}$	0	$a_{1,2}$	...	$a_{1,N}$
X_2	$a_{2,0}$	$a_{2,1}$	0	...	$a_{2,N}$
...	...	...	...	0	...
X_N	$a_{N,0}$	$a_{N,1}$	$a_{N,2}$	...	0

$$A = \begin{bmatrix} 0 & a_{0,1} & a_{0,2} & ... & a_{0,N} \\ a_{1,0} & 0 & a_{1,2} & ... & a_{1,N} \\ a_{2,0} & a_{2,1} & 0 & ... & a_{2,N} \\ ... & ... & ... & 0 & ... \\ a_{N,0} & a_{N,1} & a_{N,2} & ... & 0 \end{bmatrix}$$

Table 2: Matrix of a general *Bayesian Network*

Arrows	From			
To	C	X_1	X_2	X_3
C	0	0	0	0
X_1	1	0	0	0
X_2	1	1	0	0
X_3	1	1	0	0

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \qquad \begin{aligned} \Pi_{X_1} &= \{C\} && \text{(from 2. row of } A\text{)} \\ \Pi_{X_2} &= \{C, X_1\} && \text{(from 3. row of } A\text{)} \\ \Pi_{X_3} &= \{C, X_1\} && \text{(from 4. row of } A\text{)} \end{aligned}$$

$$P(C, X_1, X_2, X_3) = P(C) \cdot \prod_{i=1}^{3} P(X_i|\Pi_{X_i})$$

Table 3: Matrix of the *Bayesian Network* in Figure 15 on the right side and calculation of $P(C, X_1, X_2, X_3)$

Arrows	From			
To	C	X_1	X_2	X_3
C	0	0	0	0
X_1	1	0	0	0
X_2	1	1	0	0
X_3	1	1	1	0

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad \begin{aligned} \Pi_{X_1} &= \{C\} && \text{(from 2. row of } A\text{)} \\ \Pi_{X_2} &= \{C, X_1\} && \text{(from 3. row of } A\text{)} \\ \Pi_{X_3} &= \{C, X_1, X_2\} && \text{(from 4. row of } A\text{)} \end{aligned}$$

$$P(C, X_1, X_2, X_3) = P(C) \cdot \prod_{i=1}^{3} P(X_i | \Pi_{X_i})$$

Table 4: Matrix of the *Full Bayesian Network* in Figure 13 and calculation of $P(C, X_1, X_2, X_3)$

3.3.5 Naive Bayesian Classifier

The special properties of the *Naive Bayesian Classifier* are induced by the following assumption:

Assumption
The input variables (features) $X_i, i = 1, ..., n$ are independent given the value $c_r, r = 1, ..., m$ of the class variable C (conditional independence) (see also Figure 17).

Conditional independence should satisfy the following condition:

$$P(\alpha|\beta, \gamma) = P(\alpha|\beta) \tag{159}$$

where the random variables α and γ become independent given the random variable β.

Transferred to our case we obtain the assumption:

$$P(X_i|X_j, C) = P(X_i|C) \; \forall i, j \in 1, ..., N, i \neq j \tag{160}$$

$$P(X_i|C, X_1, ..., X_{i-1}, X_{i+1}, ..., X_N) = P(X_i|C) \tag{161}$$

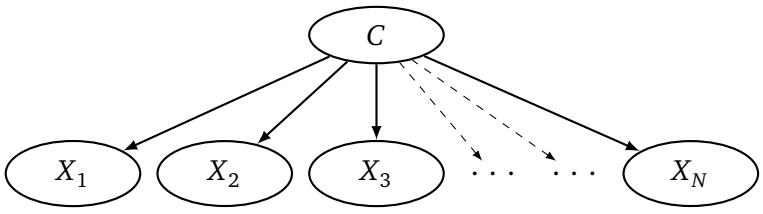

Figure 17: *Bayesian Network* of *Naive Bayesian Classifier* [14] [20] [17]

Training and Classification of the Naive Bayesian Classifier

Training
The parameters of the *Naive Bayesian Classifier* are the distributions $P(C)$ and $P(X_i|C), i = 1, ..., n$.

The distribution $P(C)$ can be calculated by counting the relative frequency of the values $c_r, r = 1, ..., m$ in the dataset. The distributions $P(X_i|C)$ can be calculated by counting the relative frequency of the values that X_i takes in the data samples, where $C = c_r, r = 1, ..., m$, hence, there are $n \cdot m$ distributions contained in $p_{i,r} = P(X_i|C = c_r), r = 1, ..., m, i = 1, ..., n$. The number of discrete values that $X_i|C = c_r$ can take determines the number of parameters (probabilities) that are required to represent a single distribution $p_{i,r}$.
An example of the *Naive Bayesian Classifier* is calculated in Section 3.3.7 on page 51.

Classification
Calculating the probability distribution p_r of the output variable C given the values of the N features $X_1, ..., X_N$ is the goal of this method. The distribution p_r and the predicted output value c_{out} can be defined as follows:

$$p_r = P(C = c_r|(X_1, X_2, ..., X_N) = (x_1, x_2, ..., x_N)),\ r = 1, ..., m \tag{162}$$
$$r_{out} = \arg\max_r(p_r) \tag{163}$$
$$c_{out} = c_{r_{out}} \tag{164}$$

where m is the number of classes.

The distribution can be calculated as follows [14]:

$$P(\beta|\alpha) = \frac{P(\alpha|\beta) \cdot P(\beta)}{P(\alpha)} = \frac{P(\alpha, \beta)}{P(\alpha)} \text{ (Bayes' theorem)} \tag{165}$$
$$P(C|X_1, X_2, ..., X_N) = \frac{P(C, X_1, X_2, ..., X_N)}{P(X_1, X_2, ..., X_N)} \tag{166}$$
$$P(C, X_1, X_2, ..., X_N) = P(C) \cdot P(X_1, X_2, ..., X_N|C) \tag{167}$$
$$P(X_1, X_2, ..., X_N|C) = \prod_{i=1}^{N} P(X_i|C) \quad \text{(Assumption of conditional independence)} \tag{168}$$

Combining the equations (162), (166), (167) and (168) leads to:

$$p_r = P(C = c_r|(X_1, ..., X_N) = (x_1, ..., x_N)) = P(C = c_r)\frac{\prod_{i=1}^{N} P(X_i = x_i|C = c_r)}{P((X_1, ..., X_N) = (x_1, ..., x_N))},\ r = 1, ..., m \tag{169}$$

For a given input value combination $(x_1, ..., x_N)$ the value of $P(X_1, ..., X_N)$ becomes irrelevant for the classification (finding the class with the maximum probability) because it is the same value for each class c_r. Id est $P((X_1, ..., X_N) = (x_1, ..., x_N))$ is constant for a certain given value combination $(x_1, ..., x_N)$.

$$p_r = P(C = c_r|(X_1, ..., X_N) = (x_1, ..., x_N)) \propto P(C = c_r) \cdot \prod_{i=1}^{N} P(X_i = x_i|C = c_r),\ r = 1, ..., m \tag{170}$$

Advantages of the Naive Bayesian Classifier

Due to the restrictive assumption, Naive Bayesian Classifier has a reduced number of independent parameters and a reduced computation time. Because of the small number of parameters, smaller datasets allow a good performance. If the parameters (considered distributions) also contained joint probabilities of features, a much larger dataset would be required to obtain a good approximation of the probability distributions.

Disadvantages of the Naive Bayesian Classifier

If the assumption do not hold true for an application, the classification accuracy becomes poor. If dependent features are considered as conditional independent, their influence on the classification is too high compared to the influence of features that are really conditional independent from the other features.

3.3.6 Limited Dependence Classifier

The *Bayesian Network* of the *Limited Dependence Classifier* [20] [14] considers at most k conditional dependencies between the features.
The *KDB (k-dependence Bayesian) Algorithm* generates the *Bayesian Network* of the *Limited Dependence Classifier*.
The maximum number of conditional dependencies between the N features is limited by the first parameter k that is given to the algorithm, where $k \in \{0, 1, 2, ..., N-1\}$. Then the algorithm removes the $N-1-k$ conditional dependencies between the N features that have the smallest conditional mutual information. The second parameter is θ, which defines a minimum conditional mutual information between features that is required to be considered (in terms of arrows) in the *Bayesian Network*. If the value k is set to the maximum value $k = k_{max} = N-1$ and $\theta = 0$, the algorithm generates the *Bayesian Network* of a *Full Bayesian Classifier*. If the value k is set to the minimum value $k = k_{min} = 0$ and $\theta = 0$, the algorithm generates the *Bayesian Network* of a *Naive Bayesian Classifier*.
If θ is set to some positive value $\theta = \theta_1 > 0$ and $k = k_{max} = N-1$, it depends on the data and θ_1 how complex the generated *Bayesian Network* becomes.
In spite of setting k to $k = k_{max} = N-1$, at the beginning, the *Bayesian Network* of a *Naive Bayesian Classifier* is generated if θ is set too high. If θ is set reasonable, a *Bayesian Network* of a *Naive Bayesian Classifier* is only generated if the conditional dependencies between the features are so small that they do not have to be considered.

In summary, if the complexity of the generated *Bayesian Network* should be independent from the data, only the parameter k should be varied and θ should be set to zero ($\theta = 0$). However, if the complexity should be dependent from the data, θ should be set to a reasonable positive value $\theta = \theta_1 > 0$, while k can be used to set an upper limit of the complexity.

Algorithm 1 KDB Algorithm from [20] [14]

procedure KDB ALGORITHM
For each feature X_i, compute mutual information, $I(X_i;C)$, where C is the class.
Compute class conditional mutual information $I(X_i;X_j|C)$, for each pair of features X_i and X_j, where $i \neq j$.
Let the used variable list, S, be empty.
Let the Bayesian network being constructed, BN, begin with single class node, C.
    while S includes not all domain features **do**
        Select feature X_{max} which is not in S and has the largest values $I(X_{max};C)$.
        Add a node to BN representing X_{max}.
        Add an arc from C to X_{max} in BN.
        Consider $m = min(|S|,k)$ distinct features X_j in S with the highest value for $I(X_{max};X_j|C)$, and only add arcs from X_j to X_{max} if $I(X_{max};X_j|C) > \theta$, where θ is a mutual information threshold and k is the maximum number of dependencies that are considered.
        Add X_{max} to S.
    end while
Compute the conditional probability tables infered by the structure of BN by using counts from DB, and output BN.
end procedure

Number of arrows (or arcs) in a *Bayesian Network*
For $\theta = 0$ the *KDB Algorithm* generates N arrows that point from the class C to the N features. For the first $k+1$ features with the highest value of $I(X_i;C)$ the features are fully connected by $\frac{(k+1)((k+1)-1)}{2}$ arrows and the rest of the features ($N-(k+1)$ features) is connected to k previous features that have the highest $I(X_i;X_j|C)$ values (as explained in the *KDB Algorithm*).
Therefore, the total number of arrows is

$$\begin{aligned}
n_A|_{(\theta=0)} =& N + \frac{(k+1)\cdot((k+1)-1)}{2} + k\cdot(N-(k+1)) \\
=& N + \frac{k\cdot(k+1)}{2} + k\cdot(N-k-1)) \\
=& N + k\frac{(k+1)}{2} + k\frac{2N-2k-2}{2}) \\
=& N + \frac{k}{2}(2N-k-1) \quad \text{(Limited Dependence Bayesian Classifier)} \qquad (171) \\
n_A|_{(\theta=0,\, k=N-1)} =& N + \frac{N-1}{2}(2N-N+1-1) = N + \frac{N\cdot(N-1)}{2} \\
=& \frac{N\cdot(N+1)}{2} \quad \text{(Full Bayesian Classifier)} \qquad (172) \\
n_A|_{(\theta=0,\, k=0)} =& N + \frac{0}{2}(2N-0-1) = N \quad \text{(Naive Bayesian Classifier)} \qquad (173)
\end{aligned}$$

In Figures 18 to 21 is showed how the complexity of the *Limited Dependence Bayesian Classifier* can be scaled by changing the parameter k while the parameter θ is set to zero.

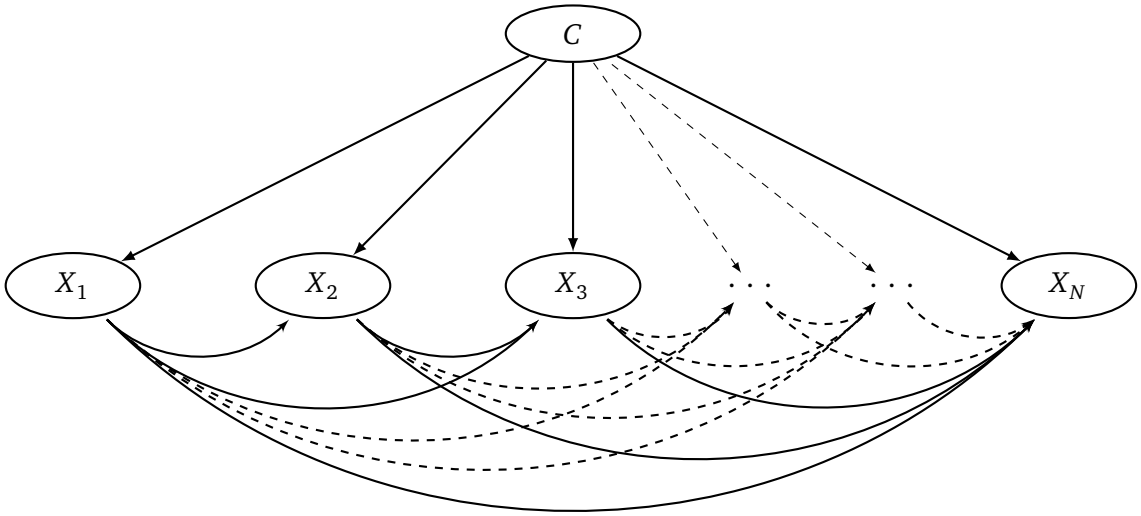

Figure 18: *Bayesian Network* of a *Limited Dependence Classifier* with N features
Parameters: $k = k_{max} = N - 1,\ \theta = 0$
The *Bayesian Network* has $\frac{N \cdot (N+1)}{2}$ arrows (see equation (172)) and is identical to the *Bayesian Network* of the *Full Bayesian Classifier*.

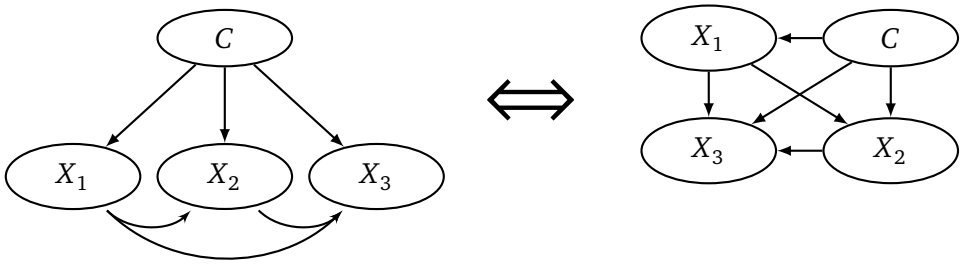

Figure 19: *Bayesian Network* of a *Limited Dependence Classifier* with $N = 3$ features
Parameters: $k = k_{max} = 3 - 1 = 2,\ \theta = 0$
The *Bayesian Network* has $\frac{N \cdot (N+1)}{2} = \frac{3 \cdot (3+1)}{2} = 6$ arrows (see equation (172)) and is identical to the *Bayesian Network* of the *Full Bayesian Classifier*.

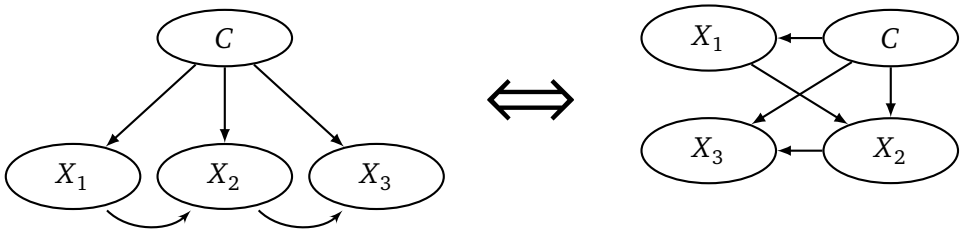

Figure 20: *Bayesian Network* of a *Limited Dependence Classifier* with $N = 3$ features
Parameters: $k = 1,\ \theta = 0$
Due to $I(X_2; X_3|C) > I(X_1; X_3|C)$ the arrow to X_3 comes from X_2 and not from X_1.
The *Bayesian Network* has $N + \frac{k}{2}(2N - k - 1) = 3 + \frac{1}{2}(2 \cdot 3 - 1 - 1) = 5$ arrows (see equation (171)).

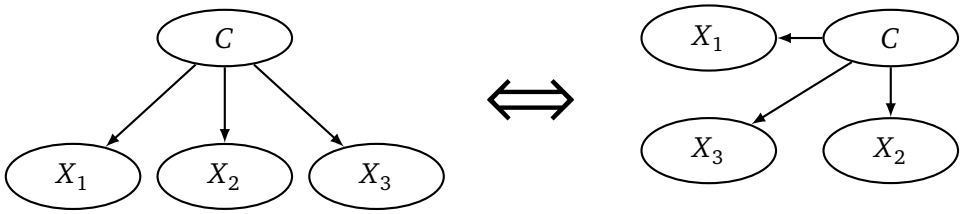

Figure 21: *Bayesian Network* of a *Limited Dependence Classifier* with $N = 3$ features
Parameters: $k = k_{min} = 0,\ \theta = 0$
The *Bayesian Network* has $N = 3$ arrows (see equation (173)) and is identical to the *Bayesian Network* of the *Naive Bayesian Classifier*.

Training and Classification of the Limited Dependence Classifier

Training
The training consists of the following two steps:
1) Computing the *Bayesian Network* using the *KDB Algorithm* (Creating a suitable model structure for the data).
2) Calculating the conditional probability tables for the classifier (Actual training: Calculation of the model parameters).

The first step was explained in the *KDB Algorithm* the second step can be realized as follows:

2.1) Calculation of the required probability distributions from the data:

$$P(C) \tag{174}$$
$$P(\Pi_{X_i}) \ \forall i = 1,...,N \tag{175}$$
$$P(X_i, \Pi_{X_i}) \ \forall i = 1,...,N \tag{176}$$

where $P(C)$ can be calculated by counting the relative frequency of the classes $c_r, r = 1,...,m$ in the data samples. While $P(\Pi_{X_i})$ and $P(X_i, \Pi_{X_i})$ can be calculated by counting the relative frequency of the value combinations that Π_{X_i} and (X_i, Π_{X_i}) can take respectively in the data samples.

2.2) The following calculation of the conditional probability distributions can either be calculated while training or while the classification.

$$P(X_i|\Pi_{X_i}) = \frac{P(X_i, \Pi_{X_i})}{P(\Pi_{X_i})} \ \forall i = 1,...,N \tag{177}$$

The conditional probabilities are used for the equation

$$P(C|X_1,X_2,...,X_N) \propto P(C,X_1,X_2,...,X_N) = P(C) \cdot \prod_{i=1}^{N} P(X_i|\Pi_{X_i}) \tag{178}$$

of the classification. If the conditional probabilities are calculated while training, more memory space is required. If the conditional probabilities are calculated while the classification, only one value of the conditional probability distributions Π_{X_i} and (X_i, Π_{X_i}) has to be calculated because the values $(x_1, x_2, ..., x_N)$ that $(X_1, X_2, ..., X_N)$ take are known.

Classification
If the *Limited Dependence Classifier* was trained, a classification of an input value combination $(x_1, x_2, ..., x_N)$ can be performed.
The classification using the *Limited Dependence Classifier* is similar to the classification that is used in the *Naive Bayesian Classifier*, which is a particular case of the *Limited Dependence Classifier*. The only difference is that the conditional probabilities of every feature can have different conditions. The condition of a feature $X_i,\ i = 1, ..., N$ is Π_{X_i}, where Π_{X_i} represents the set of parents of X_i in the *Bayesian Network* [17]. A *Bayesian Network* is completely represented by $\Pi_{X_i}, i = 1, ..., N$.

Then the classification is performed with the following steps:

$$p_r = P(C = c_r | (X_1, X_2, ..., X_N) = (x_1, x_2, ..., x_N)) \propto P(C = c_r) \cdot \prod_{i=1}^{N} P(X_i | \Pi_{X_i} = \pi_{X_i,r}) \tag{179}$$

$$r = 1, ..., m$$

$$r_{out} = \arg\max_r(p_r) \tag{180}$$

$$c_{out} = c_{r_{out}} \tag{181}$$

where Π_{X_i}, that always contains the variable C, takes the values $\pi_{X_i,r}$, which contain c_r.

Computational Complexity of the Limited Dependence Classifier

Computing the *Bayesian Network* using the *KDB Algorithm* requires $\mathcal{O}(N^2 \cdot s \cdot c \cdot v^2)$.
Calculating the conditional probability tables for the classifier takes $\mathcal{O}(N \cdot (s + v^k))$.
Classification of one test sample using the trained classifier requires $\mathcal{O}(N \cdot c \cdot k)$.

s: Number of data samples (observations)
c: Number of classes or discrete output values
v: Maximum number of discrete values that a feature may take [20]

Advantages of the Limited Dependence Classifier

The *Limited Dependence Classifier* can use *Bayesian Networks* of different complexities for the classification. The number of conditional dependencies of one feature can be limited using the parameter k. The *KDB Algorithm* makes reasonable decisions regarding which conditional dependencies are kept under a limitation. In addition the *KDB Algorithm* has the optional capability to remove insignificant conditional dependencies in order to get an even smaller number of considered conditional dependencies than the limit allows. That is the case if the parameter θ is set to a reasonable positive value, while this capability is switched off for $\theta = 0$.
The number of model parameters that has to be computed decreases for every conditional dependence that is removed from the *Bayesian Network* (see Section 3.3.2 on page 39 for an example). Hence, the complexity of the model can be controlled.

Disadvantages of the Limited Dependence Classifier

The capabilites of the *Limited Dependence Classifier* are consuming more calculation time compared to the *Naive Bayesian Classifier*. Especially if the given system has not many useful conditional dependencies between the features, the investment of the higher computation time will not lead to a significant higher classification accuracy than the *Naive Bayes Classifier* can achieve.

3.3.7 Laplacian Correction

The *Laplacian Correction* or *Laplace Estimator* [10] is a method that allows to avoid computing probability values of zero. The underlying assumption is that the accuracy of the estimated probability distributions computed from data will hardly be affected if for every value combination one additional occurrence is assumed and the entire number of observations (training data samples) is great enough. Therefore, one additional observation for every possible value combination is added to the data and all value combinations that did not occur in the training data, before, obtain a probability that is very small but not zero.

The advantage of avoiding the zero probabilities is explained in the following example of a *Naive Bayesian Classifier*:

Original Training Data
Consider training data that consist of 1000 samples with the following probability distributions:

$$P(C = c_1, X_1 = x_{1,1}, X_2 = x_{2,1}) = 25\% \text{ (250 samples)}$$
$$P(C = c_1, X_1 = x_{1,1}, X_2 = x_{2,2}) = 25\% \text{ (250 samples)}$$
$$P(C = c_2, X_1 = x_{1,2}, X_2 = x_{2,1}) = 50\% \text{ (500 samples)}$$

$$P(X_1 = x_{1,1}) = 50\%$$
$$P(X_1 = x_{1,2}) = 50\%$$
$$P(X_2 = x_{2,1}) = 75\%$$
$$P(X_2 = x_{2,2}) = 25\%$$
$$P(C = c_1) = 50\%$$
$$P(C = c_2) = 50\%$$

Probability distributions of classifier:

$$P(X_1 = x_{1,1} | C = c_1) = 100\% \tag{182}$$
$$P(X_1 = x_{1,2} | C = c_1) = 0\% \tag{183}$$
$$P(X_1 = x_{1,1} | C = c_2) = 0\% \tag{184}$$
$$P(X_1 = x_{1,2} | C = c_2) = 100\% \tag{185}$$

$$P(X_2 = x_{2,1} | C = c_1) = 50\%$$
$$P(X_2 = x_{2,2} | C = c_1) = 50\%$$
$$P(X_2 = x_{2,1} | C = c_2) = 100\%$$
$$P(X_2 = x_{2,2} | C = c_2) = 0\%$$

Classification of an input value combination:

$$P(C|X_1, X_2) \propto P(C) \cdot P(X_1|C) \cdot P(X_2|C) \tag{186}$$

Interpretation of the Training Data
In the training data $x_{1,1}$ always occurs in combination with c_1, while $x_{1,2}$ occurs in combination with

c_2, therefore, the occurrence of $x_{1,1}$ or $x_{1,2}$ is an important indication of the output value C (see also equations (182) to (185)).

Classification without *Laplacian Correction*
If an unknown test data sample $(x_{1,2}, x_{2,2})$ which was not included in the training data has to be classified, the classification using the equations (182) to (186) leads to the following result:

$$\begin{aligned} P(X_1 = x_{1,2}, X_2 = x_{2,2}) &= 0\% \\ P(C = c_1 | X_1 = x_{1,2}, X_2 = x_{2,2}) &\propto P(C = c_1) \cdot P(X_1 = x_{1,2} | C = c_1) \cdot P(X_2 = x_{2,2} | C = c_1) \\ &= 50\% \cdot 0\% \cdot 50\% = 0\% \\ P(C = c_2 | X_1 = x_{1,2}, X_2 = x_{2,2}) &\propto P(C = c_2) \cdot P(X_1 = x_{1,2} | C = c_2) \cdot P(X_2 = x_{2,2} | C = c_2) \\ &= 50\% \cdot 100\% \cdot 0\% = 0\% \end{aligned} \tag{187}$$

Due to the identical probabilities

$$P(C = c_1 | X_1 = x_{1,2}, X_2 = x_{2,2}) = P(C = c_2 | X_1 = x_{1,2}, X_2 = x_{2,2}) = 0\% \tag{188}$$

there is no explicit classification possible.
As mentioned before there is a dependence between X_1 and C. This dependence is ignored in the classification because $P(X_2 = x_{2,2} | C = c_2) = 0\%$ cancels $P(X_1 = x_{1,2} | C = c_2) = 100\%$ in equation (187). The reason is the absence of training samples which contain combinations of the values c_2 and $x_{2,2}$.
The *Laplacian Correction* is aimed at avoiding this kind of problem.

Training Data with *Laplacian Correction*

$$\begin{aligned} P(C = c_1, X_1 = x_{1,1}, X_2 = x_{2,1}) &= \frac{251}{1008} = 24.9008\% \text{ (251 samples)} \\ P(C = c_1, X_1 = x_{1,1}, X_2 = x_{2,2}) &= \frac{251}{1008} = 24.9008\% \text{ (251 samples)} \\ P(C = c_2, X_1 = x_{1,2}, X_2 = x_{2,1}) &= \frac{501}{1008} = 49.7024\% \text{ (501 samples)} \\ P(C = c_2, X_1 = x_{1,2}, X_2 = x_{2,2}) &= \frac{1}{1008} = 0.0992\% \text{ (1 sample)} \end{aligned} \tag{189}$$

$$\begin{aligned} P(C = c_1, X_1 = x_{1,2}, X_2 = x_{2,1}) &= \frac{1}{1008} = 0.0992\% \text{ (1 sample)} \\ P(C = c_1, X_1 = x_{1,2}, X_2 = x_{2,2}) &= \frac{1}{1008} = 0.0992\% \text{ (1 sample)} \\ P(C = c_2, X_1 = x_{1,1}, X_2 = x_{2,1}) &= \frac{1}{1008} = 0.0992\% \text{ (1 sample)} \\ P(C = c_2, X_1 = x_{1,1}, X_2 = x_{2,2}) &= \frac{1}{1008} = 0.0992\% \text{ (1 sample)} \end{aligned} \tag{190}$$

Classification with *Laplacian Correction*
After the *Laplacian Correction*, the test sample $(x_{1,2}, x_{2,2})$ is contained in two samples of the training data, one sample with c_1 and one sample with c_2 (see equations (189) and (190)). Hence, both samples together can not explain, which class is correct for this test sample. However, because of the other samples, the classification leads to the following (generalization) result:

Probability distributions of classifier:

$$P(C = c_1) = 50\% \tag{191}$$

$$P(C = c_2) = 50\% \tag{192}$$

$$
\begin{aligned}
P(X_1 = x_{1,1}|C = c_1) &= \frac{1}{50\%} \cdot \frac{251+251}{1008} = 99.6032\% \\
P(X_1 = x_{1,2}|C = c_1) &= \frac{1}{50\%} \cdot \frac{1+1}{1008} = 0.3968\% \qquad (193) \\
P(X_1 = x_{1,1}|C = c_2) &= \frac{1}{50\%} \cdot \frac{1+1}{1008} = 0.3968\% \\
P(X_1 = x_{1,2}|C = c_2) &= \frac{1}{50\%} \cdot \frac{1+501}{1008} = 99.6032\% \qquad (194)
\end{aligned}
$$

$$
\begin{aligned}
P(X_2 = x_{2,1}|C = c_1) &= \frac{1}{50\%} \cdot \frac{1+251}{1008} = 50\% \\
P(X_2 = x_{2,2}|C = c_1) &= \frac{1}{50\%} \cdot \frac{1+251}{1008} = 50\% \qquad (195) \\
P(X_2 = x_{2,1}|C = c_2) &= \frac{1}{50\%} \cdot \frac{1+501}{1008} = 99.6032\% \\
P(X_2 = x_{2,2}|C = c_2) &= \frac{1}{50\%} \cdot \frac{1+1}{1008} = 0.3968\% \qquad (196)
\end{aligned}
$$

Using the values from the equations (191) to (196) we obtain the following result:

$$
\begin{aligned}
P(C = c_1|X_1 = x_{1,2}, X_2 = x_{2,2}) &\propto P(C = c_1) \cdot P(X_1 = x_{1,2}|C = c_1) \cdot P(X_2 = x_{2,2}|C = c_1) \\
&= 50\% \cdot 0.3968\% \cdot 50\% = 0.0992\% \\
P(C = c_2|X_1 = x_{1,2}, X_2 = x_{2,2}) &\propto P(C = c_2) \cdot P(X_1 = x_{1,2}|C = c_2) \cdot P(X_2 = x_{2,2}|C = c_2) \qquad (197) \\
&= 50\% \cdot 99.6032\% \cdot 0.3968\% = 0.1976\%
\end{aligned}
$$

As we can see in equations (193) and (196) the probabilities $P(X_1 = x_{1,2}|C = c_1)$ and $P(X_2 = x_{2,2}|C = c_2)$ have increased from 0% to 0.3968%, while we can see in equation (194) that the probability $P(X_1 = x_{1,2}|C = c_2)$ has decreased from 100% to 99.6032% due to the *Laplacian Correction*.

Consequently, the accuracy of the estimation of the probability distributions with respect to the original training data reduces slightly, but the generalization capability of the classifier increases due to the assumption that every value combination is possible (with a very small probability), even if it is not contained in the original training data.
With respect to the original training data the results of both classes are still near to zero, but the class which is more probable can be identified now ($0.0992 < 0.1976$). Therefore, with a *Laplacian Correction* the test sample $(x_{1,2}, x_{2,2})$ is assigned to class c_2 and the dependence of X_1 and C in the training data is not longer ignored or canceled while the classification.

3.3.8 Linear Single-Layer Perceptron

The *Linear Single-Layer Perceptron* is a simple artificial neural network, which is able to learn affine (linear with offset) relations between input and output variables. In order to predict the output value z given the input values x and y, the linear single-layer perceptron from [1] was implemented for two inputs and one output as depicted in Figure 22.

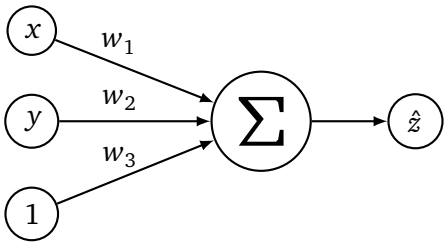

Figure 22: Schematic diagram of a *Linear Single-Layer Perceptron* with two input variables x and y and a prediction $\hat{z}$ of the output variable z. The learned content consists of the weights w_1, w_2, w_3. The input 1 is used in order to describe not only linear relations but also affine relations, which means that after the linear transformation of (x, y) an additional offset (w_3) can be added.

The behavior of the *Linear Single-Layer Perceptron* in Figure 22 can be described by the equation

$$\hat{z} = (w_1, w_2, w_3) \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = w_1 \cdot x + w_2 \cdot y + w_3, \tag{198}$$

where x and y are the input values and $\hat{z}$ is the predicted value of z. The weights $\mathbf{w}^{\mathbf{T}} = (w_1, w_2, w_3)$ are the model parameters. These model parameters are adjusted while the training of the model in order to fit the model behavior into the training data. After the training the determined parameters can be used in the model to predict output values for the given input values, even if the input values were not contained in the training data.

Training of the *Linear Single-Layer Perceptron*:
The training data consists of m examples of input and output values from a system. The training data $\mathbf{D}_{\mathbf{train}}$ is split in a matrix of input values $\mathbf{X}_{\mathbf{train}}$ and a vector with the corresponding output values $\mathbf{z}_{\mathbf{train}}$. An additional column with ones is added to the $\mathbf{X}_{\mathbf{train}}$ matrix to allow an offset value in the output that is adjusted by w_3 (see equation (198)):

$$\mathbf{D}_{\mathbf{train}} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_i & y_i & z_i \\ \vdots & \vdots & \vdots \\ x_m & y_m & z_m \end{pmatrix} \quad \Rightarrow \quad \mathbf{X}_{\mathbf{train}} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \\ x_i & y_i & 1 \\ \vdots & \vdots & \\ x_m & y_m & 1 \end{pmatrix}, \quad \mathbf{z}_{\mathbf{train}} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_i \\ \vdots \\ z_m \end{pmatrix} \tag{199}$$

The goal is to find the parameters $\mathbf{w^T} = (w_1, w_2, w_3)$ that minimize the *mean squared error* ϵ_{train} of the predicted output values $\hat{z}_i,\ i = 1, ..., m$, while testing the model on all m training data samples.

$$\epsilon_{train} = \frac{1}{m}\sum_{i=1}^{m}(\hat{z}_i - z_i)^2 \tag{200}$$

$$\epsilon_{train} = \frac{1}{m}\sum_{i=1}^{m}(w_1 \cdot x_i + w_2 \cdot y_i + w_3 - z_i)^2$$

$$\epsilon_{train} = \frac{1}{m}\sum_{i=1}^{m}(w_1 \cdot x_i + w_2 \cdot y_i + w_3 - z_i) \cdot (w_1 \cdot x_i + w_2 \cdot y_i + w_3 - z_i)$$

$$\epsilon_{train} = \frac{1}{m}(\mathbf{X_{train}} \cdot \mathbf{w} - \mathbf{z_{train}})^{\mathbf{T}} \cdot (\mathbf{X_{train}} \cdot \mathbf{w} - \mathbf{z_{train}}) \tag{201}$$

Because ϵ_{train} is squared, it has only one global extremum, which is a minimum. Therefore, the minimum of ϵ_{train} can be found by setting the derivation $\frac{\partial \epsilon_{train}}{\partial \mathbf{w}}$ to zero.

$$\begin{aligned}
\frac{\partial \epsilon_{train}}{\partial \mathbf{w}} &= \frac{2}{m} \cdot (\mathbf{X_{train}} \cdot \mathbf{w} - \mathbf{z_{train}})^{\mathbf{T}} \cdot \mathbf{X_{train}} = 0 && (202)\\
&\Leftrightarrow (\mathbf{X_{train}} \cdot \mathbf{w} - \mathbf{z_{train}})^{\mathbf{T}} \cdot \mathbf{X_{train}} = 0\\
&\Leftrightarrow \mathbf{X_{train}^T} \cdot (\mathbf{X_{train}} \cdot \mathbf{w} - \mathbf{z_{train}}) = 0\\
&\Leftrightarrow \mathbf{X_{train}^T} \cdot \mathbf{X_{train}} \cdot \mathbf{w} - \mathbf{X_{train}^T} \cdot \mathbf{z_{train}} = 0\\
&\Leftrightarrow \mathbf{X_{train}^T} \cdot \mathbf{X_{train}} \cdot \mathbf{w} = \mathbf{X_{train}^T} \cdot \mathbf{z_{train}}\\
&\Leftrightarrow \mathbf{w} = (\mathbf{X_{train}^T} \cdot \mathbf{X_{train}})^{-1} \cdot \mathbf{X_{train}^T} \cdot \mathbf{z_{train}} && (203)\\
&\Leftrightarrow \mathbf{w} = \mathbf{X_{train}^+} \cdot \mathbf{z_{train}}, && (204)
\end{aligned}$$

where $\mathbf{X^+} = (\mathbf{X^T} \cdot \mathbf{X})^{-1} \cdot \mathbf{X^T}$ is called the *Moore-Penrose pseudo-inverse* of $\mathbf{X}$.

Thus, the parameters $\mathbf{w^T} = (w_1, w_2, w_3)$ that minimize the *mean squared error* ϵ can be exactly determined.

Using the trained *Linear Single-Layer Perceptron* model for the prediction on test data:
At first the test data samples are splitted in the same manner as the training data samples.

$$\mathbf{D_{test}} = \begin{pmatrix} x_{m+1} & y_{m+1} & z_{m+1} \\ x_{m+2} & y_{m+2} & z_{m+2} \\ \vdots & \vdots & \vdots \\ x_i & y_i & z_i \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{pmatrix} \quad \Rightarrow \quad \mathbf{X_{test}} = \begin{pmatrix} x_{m+1} & y_{m+1} & 1 \\ x_{m+2} & y_{m+2} & 1 \\ \vdots & \vdots & \\ x_i & y_i & 1 \\ \vdots & \vdots & \\ x_n & y_n & 1 \end{pmatrix}, \quad \mathbf{z_{test}} = \begin{pmatrix} z_{m+1} \\ z_{m+2} \\ \vdots \\ z_i \\ \vdots \\ z_n \end{pmatrix} \tag{205}$$

The prediction on all test data can be performed in one matrix multiplication:

$$\mathbf{\hat{z}_{test}} = \mathbf{X_{test}} \cdot \mathbf{w} \tag{206}$$

where $\mathbf{w} = (\mathbf{X_{train}^T} \cdot \mathbf{X_{train}})^{-1} \cdot \mathbf{X_{train}^T} \cdot \mathbf{z_{train}}$ are the parameters of the trained model.

The *mean squared error* ϵ on test data can be used to evaluate the generalization capability of the model, it can be determined as follows:

$$\epsilon_{test} = \frac{1}{n-m} \sum_{i=m+1}^{n} (\hat{z}_i - z_i)^2 \tag{207}$$

$$\epsilon_{test} = \frac{1}{n-m} (\hat{\mathbf{z}}_{\mathbf{test}} - \mathbf{z}_{\mathbf{test}})^{\mathbf{T}} \cdot (\hat{\mathbf{z}}_{\mathbf{test}} - \mathbf{z}_{\mathbf{test}}) \tag{208}$$

Note: A single prediction $\hat{z}$ for a single input (x, y) can be performed using the equation (198) with trained parameters $\mathbf{w}^{\mathbf{T}} = (w_1, w_2, w_3)$.

3.3.9 Multi-Layer Perceptron

The *Multi-Layer Perceptron (MLP)* is an artificial neural network model that consists of multiple layers with multiple neurons, where the neurons of a layer are fully connected with all neurons of the next layer. The most commonly used MLPs have two or three neuron layers [1]. The neurons are similar to the *Linear Single-Layer Perceptron* from the previous section. However, they can have a non-affine behavior using an activation function that transforms the sum of the neuron inputs before it is passed over to the next layer.
In Figure 23 a schematic diagram of a MLP is depicted. The depicted MLP has two input variables x and y, an input layer (layer 0), a sigmoid hidden layer (layer 1) with two neurons, a linear output layer (layer 2) with one neuron and the output $\hat{z}$, which is a prediction of the output variable z. The input layer consists of inputs and has no neurons, therefore, it is not counted as neuron layer. The layers between the input layer and the output layer are called hidden layers. While the training of the MLP the parameters or weights $w_{r,j,l}$ are adjusted in order to fit the MLP model into the training data. A weight $w_{r,j,l}$ is the value which is multiplied with the output of neuron l from layer $(r-1)$ in order to be passed over to neuron j from layer r. If layer $(r-1)$ is the input layer, the inputs are treated as outputs from neurons. In Figure 23 nine weights (parameters) $w_{r,j,l}$ are used. The neurons of the hidden layer are labeled with $\Sigma|S(z_{r,j})$ which means that the input values of neuron j in layer r are at first added together to a value $z_{r,j}$, which is the output of a neuron in a linear layer. Secondly, before the value is passed over to the next layer, the value $z_{r,j}$ is non-affine transformed by the sigmoid activation function $S(z) = (1 + e^{-z})^{-1}$, which is known as logistic function. A sigmoid layer enables a non-affine behavior of the MLP and is required in order to take advantage of the multiple layers. If the MLP consisted of linear layers the model could only describe the same behavior as a simple *Linear Single-Layer Perceptron*. Depending on the application the output layer can be used as linear layer or in combination with an activation function.

Figure 24 depicts a simplified version of the MLP model from Figure 23. The input y was removed. Consequently, the model has only 7 remaining weights. The simplified model is used in an experiment (Section 4.3.2), where in a local feature subspace only the input x is relevant.

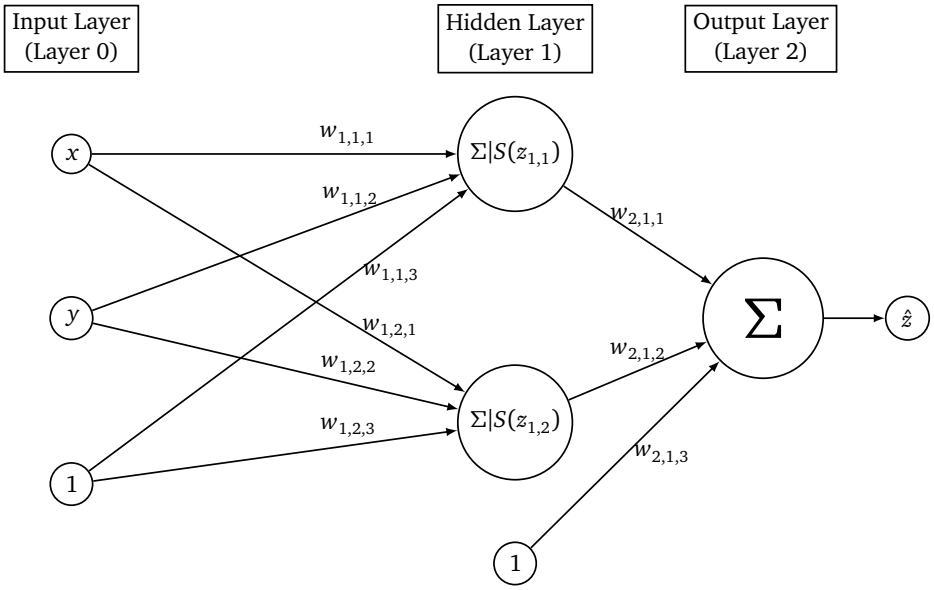

Figure 23: Schematic diagram of a *Multilayer Perceptron* with two input variables x and y, an input layer (layer 0), a sigmoid hidden layer (layer 1) with two neurons, a linear output layer (layer 2) with one neuron and the output $\hat{z}$, which is a prediction of the output variable z. The learned content consists of the 9 weights $w_{r,j,l}$, where a weight $w_{r,j,l}$ is the value which is multiplied with the output of neuron l from layer $(r-1)$ in order to be passed over to neuron j from layer r. If layer $(r-1)$ is the input layer, the inputs are treated as outputs from neurons. The neurons of the hidden layer are labeled with $\Sigma|S(z_{r,j})$ which means that the input values of a neuron are at first added together to a value $z_{r,j}$ and the output of the neuron is $S(z_{r,j})$. $S(z)$ is the sigmoid activation function $S(z) = (1+e^{-z})^{-1}$, which is known as logistic function.

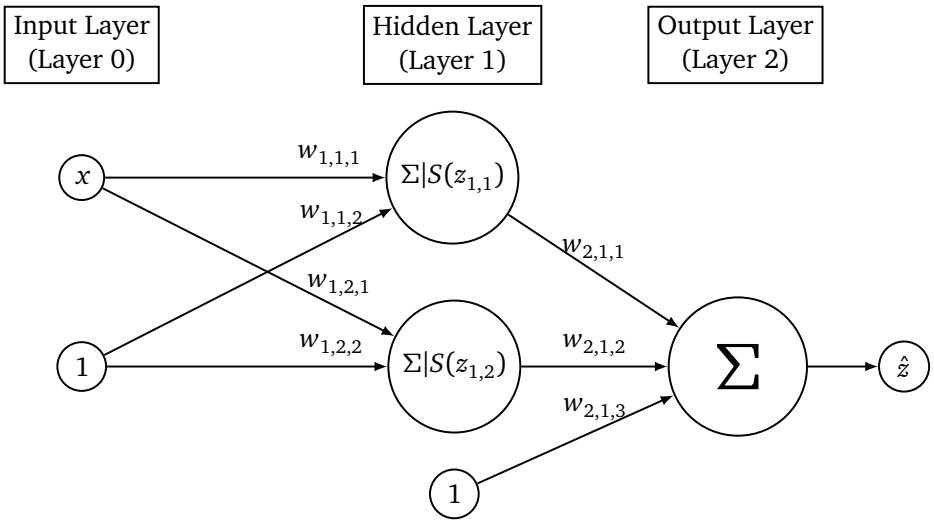

Figure 24: Schematic diagram of a *Multilayer Perceptron* with one input variable x, an input layer (layer 0), a sigmoid hidden layer (layer 1) with two neurons, a linear output layer (layer 2) with one neuron and the output $\hat{z}$, which is a prediction of the output variable z. The learned content consists of the 7 weights $w_{r,j,l}$, where a weight $w_{r,j,l}$ is the value which is multiplied with the output of neuron l from layer $(r-1)$ in order to be passed over to neuron j from layer r. If layer $(r-1)$ is the input layer, the inputs are treated as outputs from neurons. The neurons of the hidden layer are labeled with $\Sigma|S(z_{r,j})$ which means that the input values of a neuron are at first added together to a value $z_{r,j}$ and the output of the neuron is $S(z_{r,j})$. $S(z)$ is the sigmoid activation function $S(z) = (1+e^{-z})^{-1}$, which is known as logistic function.

Backpropagation

The *Backpropagation Algorithm* is a method that allows to adjust the weights of the MLP in order to fit it into the training data. It is the common training method for the MLP [1].

The goal of the training is to find the parameters or weights that minimize the *mean squared error (MSE)*

$$\epsilon_{train} = \frac{1}{m}\sum_{i=1}^{m}(\hat{z}_i - z_i)^2 \tag{209}$$

of the predicted output values $\hat{z}_i,\ i = 1,...,m$, while testing the model on all m training data samples. Due to the non-affine behavior of the MLP, a global minimum of the MSE ϵ_{train} can not be determined analytically [1].

Therefore, an iterative learning method (e.g. the *Backpropagation Algorithm*) is required. The *Backpropagation Algorithm* uses the gradient descent algorithm in order to minimize ϵ_{train}. The following learning steps describe the principle of the algorithm for online learning, more details can be found in [1].

In the online learning method not the MSE ϵ_{train} of all training data samples is considered but the prediction error $\epsilon = \hat{z} - z$ on a single training data sample is minimized in every learning step. Usually, the online learning method converges faster [1].

The weights are initialized with random values. The adjustment of the weights is performed by the learning step

$$w_{r,j,l}(t) = w_{r,j,l}(t-1) - h \cdot \frac{\partial \epsilon}{\partial w_{r,j,l}}, \tag{210}$$

which is performed for all weights $w_{r,j,l}$, where the weight $w_{r,j,l}$ connects neuron l from layer $(r-1)$ with neuron j from layer r and h is the learning rate.

The name *Backpropagation* comes from the circumstance that the derivation of the error ($\frac{\partial \epsilon}{\partial w_{r,j,l}}$) is determined recursively from the output layer of the MLP back towards the input layer.

At first the $z_{r,i}$ and $y_{r,i}$ values of all neurons are calculated for the given input values of the currently used training data sample, where $z_{r,i} = \mathbf{w}_{r,i}^T \cdot \mathbf{y}_{r-1}$ is the sum of weighted outputs from layer $r-1$ that is determined in neuron i in layer r and $y_{r,i} = f(z_{r,i})$ is the output of neuron i in layer r, which is determined using the activation function $f(z)$ (e.g. $f(z) = S(z)$). The number of layers is k, while the number of neurons per layer $q(r)$ can be different in every layer.

Then the backpropagation can be started by determining the values

$$\delta_{r,j} = \begin{cases} \frac{\partial \epsilon}{\partial y_j} \cdot f_k'(z_{k,j}),\ r = k \text{ (output layer)} \\ \left(\sum_{i=1}^{q(r+1)} \delta_{r+1,i} \cdot w_{r+1,i,j}\right) \cdot f_r'(z_{r,j}),\ r < k \text{ (hidden layers)} \end{cases} \qquad \forall r = k,...,1 \tag{211}$$

for every neuron j in every layer r recursively, starting at the last layer $r = k$, where

$$f_r'(z_{r,j}) = \frac{\partial f_r(z_{r,j})}{\partial z_{r,j}}, \tag{212}$$

$$\frac{\partial \epsilon}{\partial y_j} = \epsilon_j, \tag{213}$$

t is the current iteration and ϵ_j is the error of output j in the output layer. If only one output is used, $\frac{\partial \epsilon}{\partial y_j}$ can replaced by the prediction error ϵ. If $\delta_{r,j}$ was determined for every neuron i in every layer r, finally

$$\frac{\partial \epsilon}{\partial w_{r,j,l}} = \delta_{r,j} \cdot y_{r-1,l} \tag{214}$$

can be determined for every weight $w_{r,j,l}$ in order to perform the learning step.

The learning process can, for example, be stopped if a certain number of learning steps were performed or if the error in a certain number of iterations was below a defined threshold.

3.3.10 Decision Tree Classifier

A *Decision Tree* is a structure of hierarchical organized decision rules. It can provide a predictive answer considering given information, while applying the rules. The tree structure starts at the root node, which is connected with further nodes via branches. Every node at the end of the branches represents another case. Based on the decision rule at the root node, the first decision can be made and the next node can be entered via the branch. At the current node a further decision rule determines which node is entered next. This process is recursively performed until a node is entered that has no branches to further nodes. Those nodes are called leaves and provide the final answers of the decision tree. The maximum number of decisions that can occur while determining one answer is called the *depth* of the tree.
A *Decision Tree* can be used in many domains and can be created by experts in order to give newcomers an idea how decisions should be made.

In data mining and machine learning domains the approach to use *Decision Trees* as predictive models is called *Decision Tree Learning*, where the *Decision Trees* can be used for classification tasks using *Classification Trees* but also for regression tasks using *Regression Trees*.
Applying a *Classification Tree* is simple because the tree is partitioning the feature space recursively in many partitions using simple rules and every partition is labeled with a certain predicted class. Hence, for a given input value combination only the corresponding partition has to be determined by testing simple case discriminations (decisions) at the tree nodes. An example for a decision rule of a node is: "if $(x < 4)$, take the left branch else take the right branch".
The more challenging part of the *Decision Tree Learning* is to determine suitable rules for the given training data. Many algorithms were developed for that purpose, e.g. the ID3 (Iterative Dichotomiser 3), C4.5 (successor of ID3) and CART (Classification And Regression Tree) are common algorithms [19].

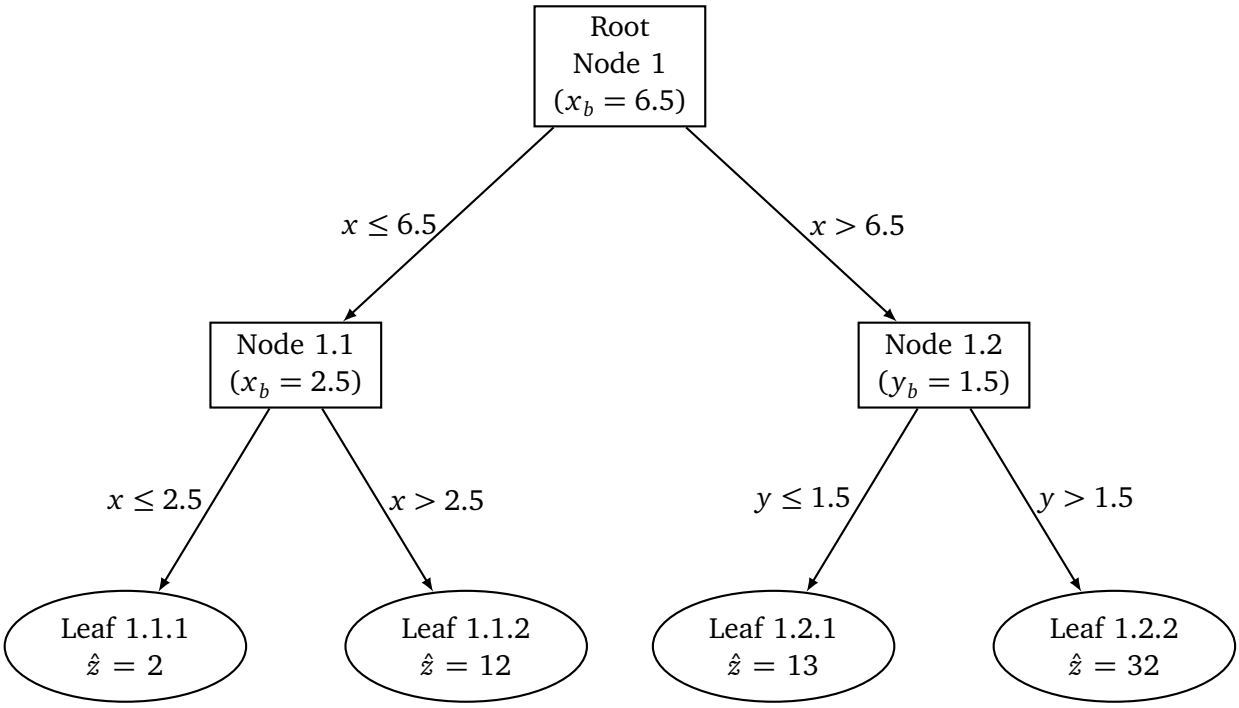

Figure 25: Schematic diagram of a *Decision Tree Classifier* with a maximum depth of 2 that was trained with Dataset 1 from Figure 27 on page 62. The feature space is partitioned stepwise. At the *Root Node 1* the feature space is split at the boundary $x = x_b = 6.5$ into two subspaces, which are split again at *Node 1.1* and *Node 1.2*. For the final four feature subspaces, the predicted output value (or class) $\hat{z}$ is defined at the *Leaves 1.1.1, 1.1.2, 1.2.1 and 1.2.2*.

In Figure 25 a schematic diagram of a *Classification Tree* with a maximum depth of 2 is depicted. It was trained with Dataset 1 from Figure 27 on page 62 using the *Decision Tree Classifier* python implementation of the *sklearn* package from http://scikit-learn.org/ [15] with a maximum depth of 2. The used split criterion was the *Information Gain*, which led to the same result as the default split criterion *Gini Impurity*. The feature space is partitioned stepwise. At the *Root Node 1* the feature space is split at the boundary $x = x_b = 6.5$ into two subspaces, which are each split again at *Node 1.1* and *Node 1.2*. For the final four feature subspaces, the predicted output value (or class) $\hat{z}$ is defined at the *Leaves 1.1.1, 1.1.2, 1.2.1 and 1.2.2*.
The resulting classification for all (x, y) combinations in the feature space is depicted in Figure 26. The areas where the same values are predicted are the partitions. In order to create a more detailed prediction, the maximum depth of the tree has to be increased. However, a smaller tree structure is easier to understand and therefore suitable for an illustration.

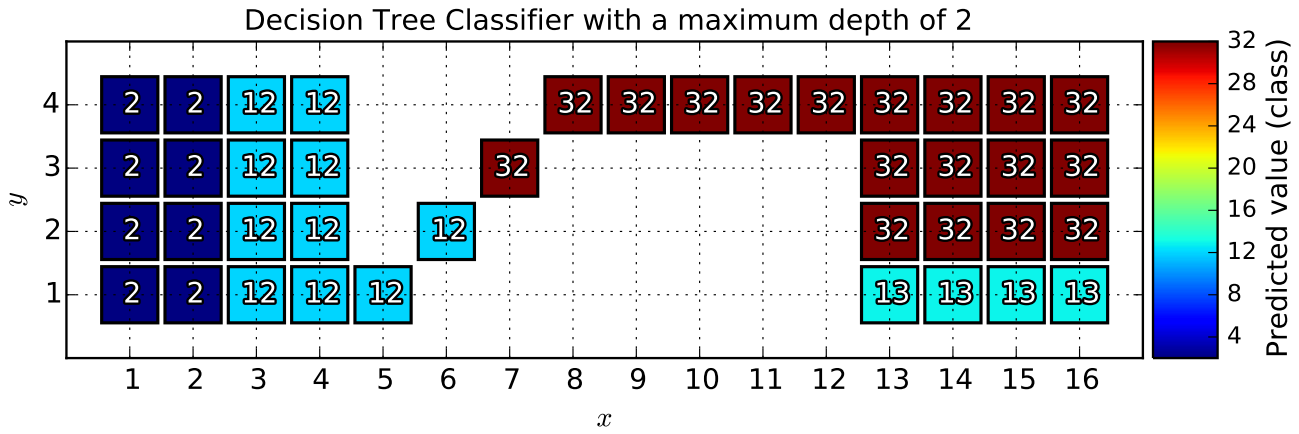

Figure 26: Result and feature space partitioning of the *Decision Tree Classifier* from Figure 25 on Dataset 1 (Figure 27 on page 62) with a maximum depth of 2. The feature subspaces are labeled with different numbers. The numbers are the predicted outputs $\hat{z}$ of the Classifier for every input value combination (x, y). The colors are also corresponding to the predicted output values $\hat{z}$, which can be read from the color bar on the right.

Information Gain as a criterion to determine suitable splits
The ID3 and C4.5 algorithms using the *Information Gain* as a splitting criterion [19]. The *Information Gain* of feature (attribute) a_i on dataset S in [19] is defined as

$$InformationGain(a_i, S) = Entropy(y, S) - \sum_{v_{i,j} \in dom(a_i)} \frac{|\sigma_{a_i = v_{i,j}} S|}{|S|} \cdot Entropy(y, \sigma_{a_i = v_{i,j}} S), \tag{215}$$

$$Entropy(y, S) = -\sum_{c_j \in dom(y)} \frac{|\sigma_{y = c_j} S|}{|S|} \log_2 \frac{|\sigma_{y = c_j} S|}{|S|}, \tag{216}$$

where S is the set of remaining data samples at the node where the $InformationGain(a_i, S)$ is determined, $\frac{|\sigma_{y=c_j} S|}{|S|}$ is the probability that the output variable y takes the value c_j in S, $\frac{|\sigma_{a_i=v_{i,j}} S|}{|S|}$ is the probability that feature (attribute) a_i takes the value $v_{i,j}$ in S, $\sigma_{a_i=v_{i,j}} S$ is the subset of S where $a_i = v_{i,j}$, $dom(y)$ is the set of values that y can take and $dom(a_i)$ is the set of attributes.
This definition is used in order to compare the attributes in their ability to reduce the entropy of the output variable y for the following nodes. The attribute a_i with the highest value of the *Information Gain* is the winner attribute of the current node. This criterion is only sufficient for building a decision

tree if for every value $v_{i,j}$ that the winner attribute can take, a following node is created. Otherwise, the decision boundary (where to split on the winner attribute) is unclear.
In order to split the remaining data samples at every node in two subsets, the method can be modified. If the attribute a_i can take for example four sorted values $v_{i,j}, j = 1, .., 4$, three possible decision boundaries are possible (between all consecutive values). The $InformationGain(a_i, S)$ is determined for the three following cases:
1.) The two value groups $\{v_{i,1}\}$ and $\{v_{i,2}, v_{i,3}, v_{i,4}\}$ are replaced by the two values $\tilde{v}_{i,1}$ and $\tilde{v}_{i,2}$.
2.) The two value groups $\{v_{i,1}, v_{i,2}\}$ and $\{v_{i,3}, v_{i,4}\}$ are replaced by the two values $\tilde{v}_{i,1}$ and $\tilde{v}_{i,2}$.
3.) The two value groups $\{v_{i,1}, v_{i,2}, v_{i,3}\}$ and $\{v_{i,4}\}$ are replaced by the two values $\tilde{v}_{i,1}$ and $\tilde{v}_{i,2}$.
Hence, for every possible split on every attribute a_i an $InformationGain(a_i, S)$ can be calculated. The overall highest $InformationGain(a_i, S)$ determines the best attribute with the corresponding split.

Due to the hierarchical determination of the decision boundaries, the decision boundaries are only locally but not globally optimal.

4 Results

4.1 Experimental Data

In this section several datasets are introduced. The datasets are designed to evaluate and compare the quality of the partitioning methods. As a baseline for the partitioning evaluations reasonable partitionings were manually applied.

4.1.1 Dataset 1: Multiplications

In *Dataset 1* (see Figure 27) the output values z are generated by a multiplication of the input values x and y. Due to missing (improbable) value combinations, the dependencies between the variables are locally changing. Therefore, the *Dataset 1* contains feature subspaces with *Synergy*, *Redundancy*, *Non-Interaction* and *Compensation*.

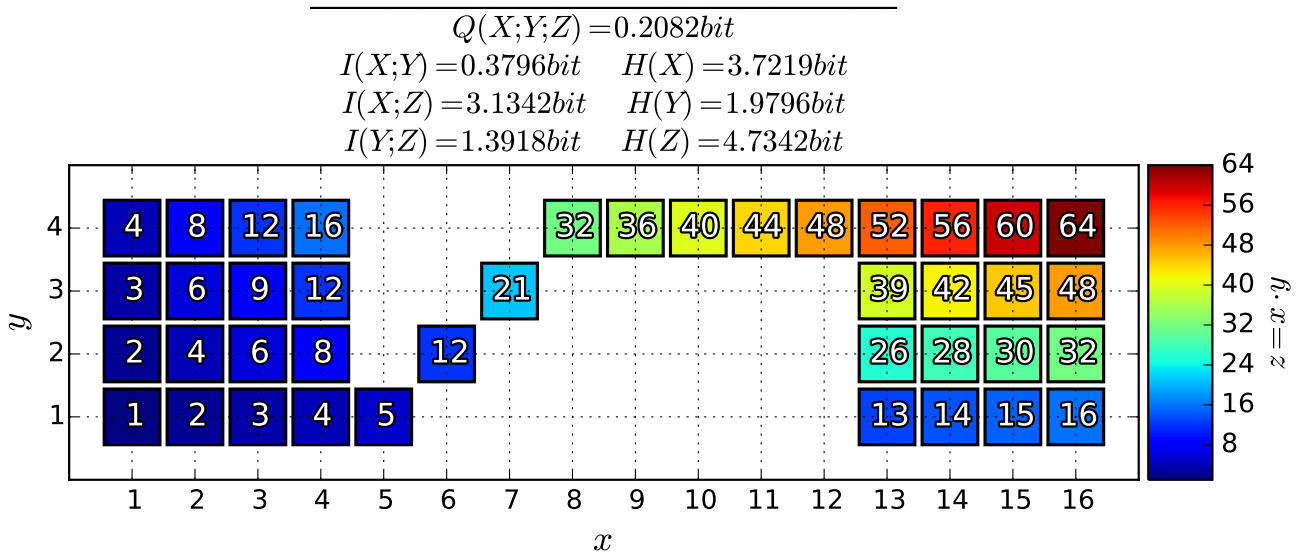

Figure 27: Samples of Dataset 1 with interval variables

The feature subspaces have the following properties:

Dataset 1-1: Local Synergy:
In Dataset 1-1 which describes the local subspace ($1 \leq x \leq 4$), X and Y are independent ($I(X;Y) = 0bit$). The independence of X and Y can also be observed in Figure 28, where the samples are distributed in the whole subspace because the knowledge of X does not constrain the values of Y and vice versa. Furthermore, the entropy $H(Z) = 3.0778bit$ is not covered by the information that is provided by $I(X;Z) = 1.0778bit$ and $I(Y;Z) = 1.0778bit$ (X and Y separately), but the *Interaction Information* $Q(X;Y;Z) = 0.9222bit$ provides the rest. The reason for the *Synergy* is that $I(X;Z) + I(Y;Z)$ is smaller than $I(X,Y;Z)$ which is caused by the circumstance that for example the output value $z = 4$ can occur for any value $x \in \{1,2,4\}$ or $y \in \{1,2,4\}$. Hence, X and Y separately do not provide an information about whether $z = 4$ occurs or not, if $x \neq 3$ and $y \neq 3$. However, if $x = 3$ or $y = 3$, $z = 4$ can not occur. While $z = 16$ can not occur if $x \neq 4$ or $y \neq 4$, thus a certain amount of dependency ($I(X;Z) = I(Y;Z) > 0$) exists. Consequently, there is no pure but a significant amount of *Synergy* in the feature subspace.

Dataset 1-2: Local Redundancy:
In Dataset 1-2 which describes the local subspace ($5 \leq x \leq 8$), X and Y are strongly dependent ($I(X;Y) = 2bit$). The strong dependency of X and Y can also be observed in Figure 29, where the samples are distributed on a diagonal line of the subspace because the knowledge of X constrains the values of Y and vice versa.
Furthermore, the entropy $H(Z) = 2bit$ is covered by the information that is provided by $I(X;Z) = 2bit$ or $I(Y;Z) = 2bit$ because X and Y are redundant. The *Interaction Information* $Q(X;Y;Z) = -2bit$ confirms the *Redundancy*.
The reason for the *Redundancy* is that $I(X;Y) = H(X) = H(Y) = H(Z) = I(X;Z) = I(Y;Z) = 2bit$, hence the information that X and Y share is the same amount of information that X and Z share, while it covers the whole entropy $H(Z)$. Therefrom, X and Y provide the same, entire information about Z.
The strong *Redundancy* in this subspace allows a lossless local feature selection because the entire information about Z is provided if only X or Y is available.

Dataset 1-3: Local Non-Interaction:
In Dataset 1-3 which describes the local subspace ($9 \leq x \leq 12$), Y is independent from X, Z and (X,Z) due to $I(X;Y) = I(Z;Y) = I(X,Z;Y) = 0bit$, thus there is a *Non-Interaction* $Q(X;Y;Z) = 0bit$. The independence of Y from the other variables can also be observed in in Figure 30, where the samples are distributed on a horizontal line of the subspace because Y is constant ($y = 4 \; \forall x \in \{9,10,11,12\} \iff H(Y) = 0$). Y can not contribute information about Z, therefrom the feature Y is useless an can be removed by a lossless local feature selection in this local subspace.
Note: If Q(X;Y;Z)=0 and at least two values in the *Mutual Information* set $\{I(X;Y), I(X;Z), I(Y;Z)\}$ are zero, a *Non-Interaction* case was found. For example, if $I(X;Y) = I(Y;Z) = Q(X;Y;Z) = 0$ then $I(Y;X,Z) = Q(X;Y;Z) + I(X;Y) + I(Y;Z) = 0$, hence, does not help to find the value that Z takes.

Dataset 1-4: Local Compensation:
In Dataset 1-4 which describes the local subspace ($13 \leq x \leq 16$), X and Y are independent ($I(X;Y) = 0bit$). The independency of X and Y can also be observed in Figure 31, where the samples are distributed in the whole subspace because the knowledge of X does not constrain the values of Y and vice versa.
Furthermore, the entropy $H(Z) = 4bit$ is exactly covered by the information that is provided by $I(X;Z) = 2bit$ and $I(Y;Z) = 2bit$ (X and Y separately), consequently the *Interaction Information* $Q(X;Y;Z) = 0bit$ vanishes.
The *Compensation* can be identified because $Q(X;Y;Z) = 0bit$, while there is no variable that is independent from the other variables.
In contrast to the subspace ($1 \leq x \leq 4$) the output variable Z has 16 different output values with the same probability ($H(Z) = 4bit$), where the knowledge of X or Y reduces the number of possible output values to four different values ($I(X;Z) = I(Y;Z) = 2bit$). Therefore, X and Y have the same importance to predict the value of Z.
As well as in the *Synergy* subspace ($1 \leq x \leq 4$), both features (X and Y) are required to cover the entropy $H(Z)$, which is a *Synergy* case. However, for the calculation of $Q(X;Y;Z)$ the variables X, Y and Z are interchangeable, hence if Z was the input variable, no other variable would be required to predict the values of X or Y. If Z is known, X and Y are known, too, which is a *Redundancy* case.
Thus, *Synergy* and *Redundancy* are contained in Dataset 1-4 and compensate each other.

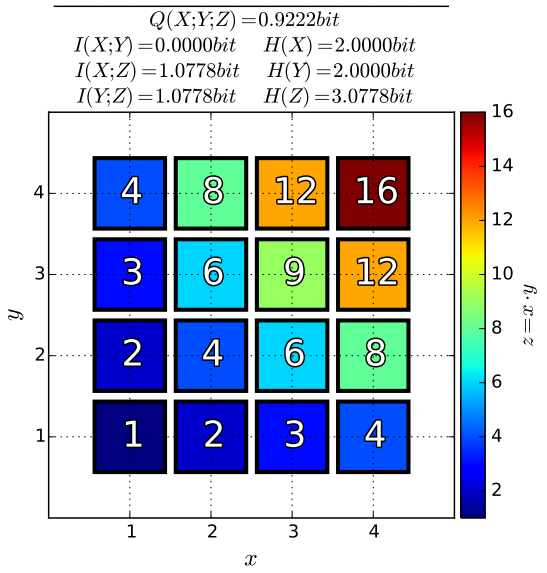

Figure 28: Samples of the feature subspace Dataset 1-1, which consists of all the samples from Dataset 1, which fulfill $1 \leq x \leq 4$.

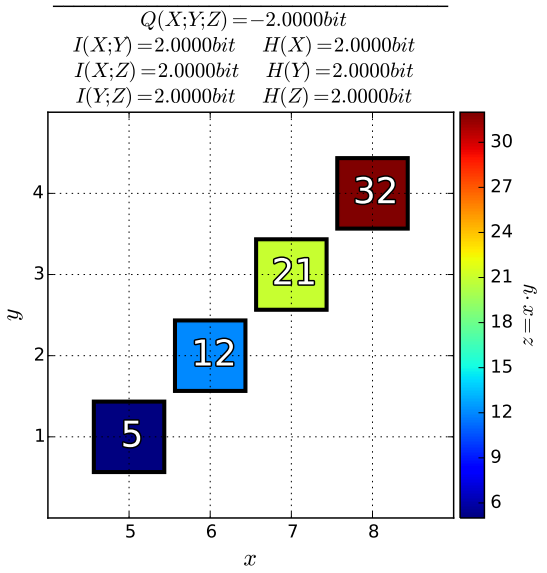

Figure 29: Samples of the feature subspace Dataset 1-2, which consists of all the samples from Dataset 1, which fulfill $5 \leq x \leq 8$.

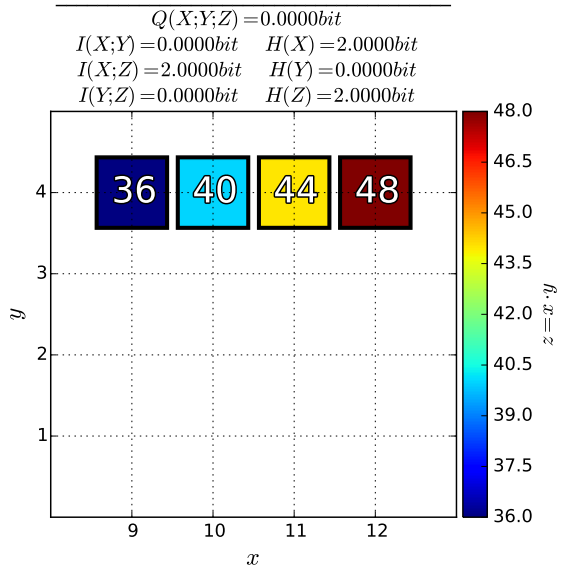

Figure 30: Samples of the feature subspace Dataset 1-3, which consists of all the samples from Dataset 1, which fulfill $9 \leq x \leq 12$.

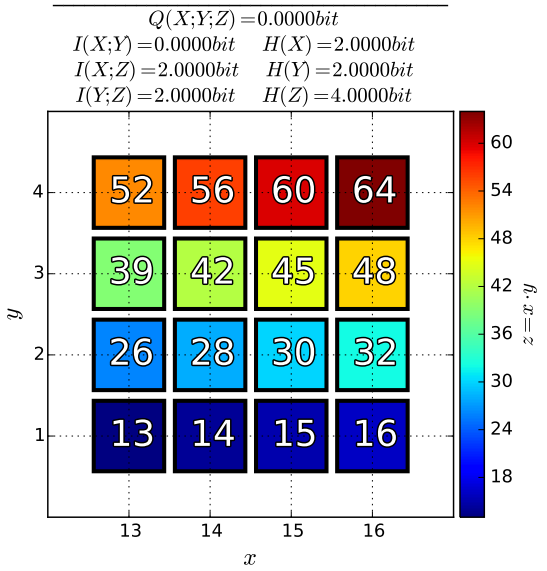

Figure 31: Samples of the feature subspace Dataset 1-4, which consists of all the samples from Dataset 1, which fulfill $13 \leq x \leq 16$.

Many classifiers are not able to learn a relation that is based on a multiplication. Thus, a simpler relation was used to generate the next dataset.

4.1.2 Dataset 2: Summation with Saturation

Simple models are not able to perform multiplications. In order to have a dataset which has local relations that can be described by a linear model, this dataset was created. If the feature space is partitioned

in the right way, the relations in the feature subspaces can be fully described by a affine relation (linear relation with offset).

In *Dataset 2* (see Figure 32) the output values z are generated by a non-linear function $z = x + min(y, 10) + 10$ which is a summation of x and y, while the influence of y goes in saturation at $y = 10$. Therefore, the *Dataset 2* contains a feature subspace with *Synergy*, before the influence of y goes in saturation. It also contains a feature subspace with *Non-Interaction*, where the change of y has no influence on z. The value relations in both feature subspaces are affine (linear with offset).

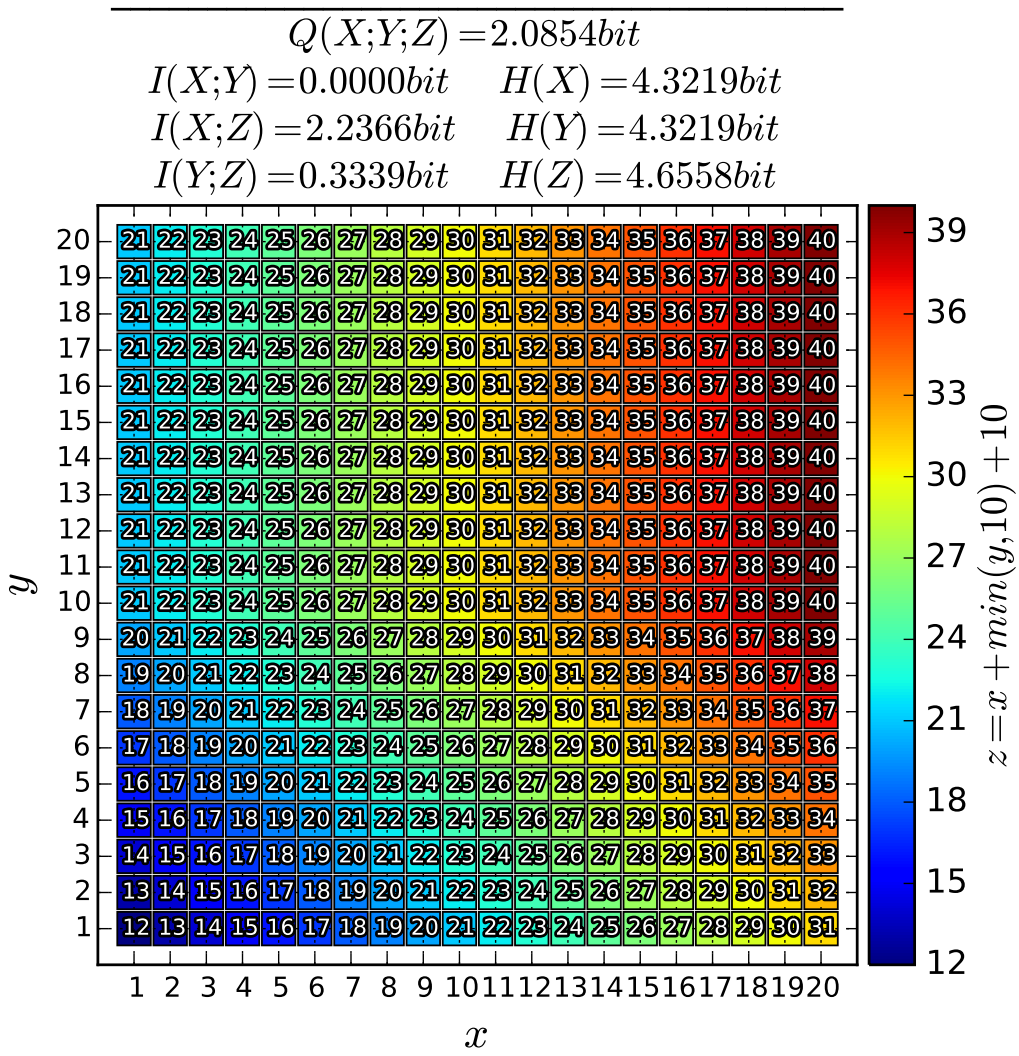

Figure 32: Samples of Dataset 2 with interval variables

Saturations

Saturations are common in technical applications, especially in the automatic control engineering, where the actuating variable is usually bounded.

A simple example is a bottle filled with gas, where one input variable describes the pressure in the bottle and another input variable describes how far the valve of the bottle was turned. The output variable is the volumetric flow of the gas that leaves the bottle. A higher pressure increases the value of the output and a wider opened valve increases the output value, too. However, if the valve is completely opened, the turning of the valve will not increase the output value. In that case the interaction of the input

variables vanishes.

The feature subspaces have the following properties:

Dataset 2-1: Local Synergy:
In Dataset 2-1 which describes the local subspace ($1 \leq y \leq 10$), X and Y are independent ($I(X;Y) = 0bit$). The independence of X and Y can also be observed in Figure 33, where the samples are distributed in the whole subspace because the knowledge of X does not constrain the values of Y and vice versa.
The relation of the variable values is $z = x + y + 10$ in that feature subspace, therefore it is an affine (linear with offset) relation.
The reason for the *Synergy* is that $I(X;Z) + I(Y;Z)$ is smaller than $I(X,Y;Z)$ which is caused by the circumstance that both input values x and y are required to predict the output value z. However, knowing x or y alone also reduces $H(Z)$. For example the output value $z = 21$ can occur for any value $x \in \{1,..,10\}$ or $y \in \{1,..,10\}$. Hence, X and Y separately do not provide an information about whether $z = 21$ occurs or not, if $x, y \leq 10$. However, if $x, y > 10$, $z = 21$ can not occur, thus a certain amount of dependency ($I(X;Z) = I(Y;Z) > 0$) exists. Hence, there is no pure *Synergy* but a significant amount of *Synergy* ($Q(X;Y;Z) = 2.9676bit$) in the feature subspace.

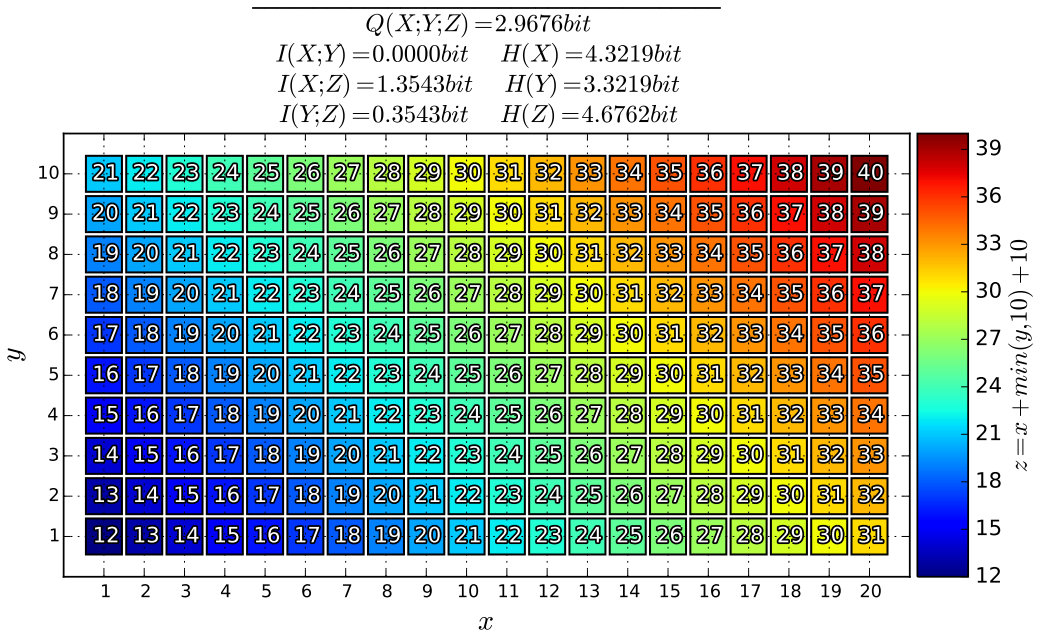

Figure 33: Samples of the feature subspace Dataset 2-1, which consists of all the samples from Dataset 2, which fulfill $1 \leq y \leq 10$.

Dataset 2-2: Local Non-Interaction:
In Dataset 2-2 which describes the local subspace ($11 \leq y \leq 20$), X and Y are independent ($I(X;Y) = 0bit$). The independence of X and Y can also be observed in Figure 34, where the samples are distributed in the whole subspace because the knowledge of X does not constrain the values of Y and vice versa.
The relation of the variable values is $z = x + 20$ in that feature subspace, therefore it is an affine (linear with offset) relation.
The reason for the *Non-Interaction* is that the value y does not influence the output value z. That can be identified by recognizing that two values in the *Mutual Information* set $\{I(X;Y), I(X;Z), I(Y;Z)\}$ are

zero, while $Q(X;Y;Z)$ is zero, too.
$I(X;Y) = I(Y;Z) = Q(X;Y;Z) = 0 \Rightarrow Y$ does not help to find the value that Z takes.

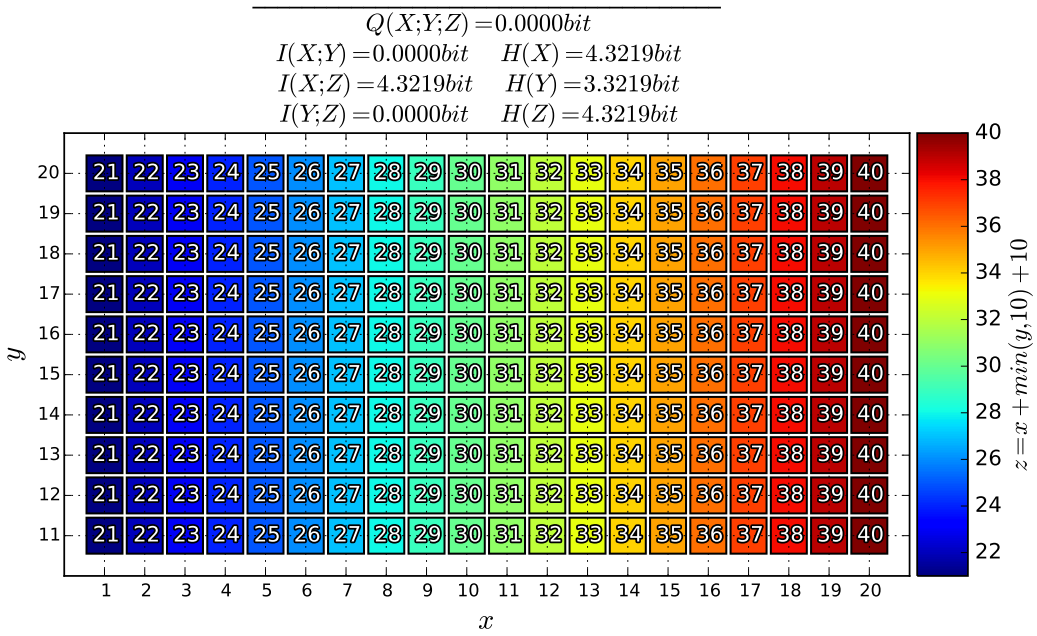

Figure 34: Samples of the feature subspace Dataset 2-2, which consists of all the samples from Dataset 2, which fulfill $11 \leq y \leq 20$.

4.1.3 Dataset 3: Weighted Synergy and Redundancy

In contrast to the previous dataset the numbers in Dataset 3 can be understood as classes because the results are not generated by calculations.
This flexible dataset allows to create a desired ratio between *Synergy* and *Redundancy*. The *Interaction Information* can be adjusted. It was mainly used to analyze the behavior of the Naive Bayesian Classifier for different *Interaction Information* values.

In *Dataset 3* (see Figure 35) every single value was set manually.
The idea is to provide a pure *Synergy* in one feature subspace and a pure *Redundancy* in the other feature subspace. The samples of each subspace can be duplicated by a given number.
Let f_{Syn} be the frequency of every sample in the pure *Synergy* feature subspace ($1 \leq x \leq 4$) and f_{Red} be frequency of every sample in the pure *Redundancy* feature subspace ($5 \leq x \leq 8$). In Figure 36 the case $f_{Syn} = 1, f_{Red} = 4$ is shown. These are the default values of the Dataset 3 and were used to calculate the information measures in Figure 35. In the case $f_{Syn} = 1, f_{Red} = 4$ the *Synergy* and *Redundancy* compensate each other. The greater f_{Syn} is compared to f_{Red} the greater is the *Interaction Information* and vise versa. Hence, the *Interaction Information* of the Dataset 3 can be set between $-2bit$ and $2bit$ using f_{Syn} and f_{Red} as follows:

$$-2bit \leq Q(X;Y;Z) \leq 2bit \tag{217}$$
$$Q(X;Y;Z) \approx +2bit \text{ (if } 4 \cdot f_{Syn} >> f_{Red}) \tag{218}$$
$$Q(X;Y;Z) = 0bit \text{ (if } 4 \cdot f_{Syn} = f_{Red}) \tag{219}$$
$$Q(X;Y;Z) \approx -2bit \text{ (if } 4 \cdot f_{Syn} << f_{Red}) \tag{220}$$

Note: The information measures in the local subspaces are not affected by the values of f_{Syn} and f_{Red} as long as $f_{Syn}, f_{Red} \geq 1$ is fulfilled. The probabilities do not change if all frequencies are increased by the same factor.

Interesting insight:
In order to produce an *Interaction Information* $Q(X;Y;Z) = 2bit$ (pure *Synergy*), at least 16 data samples are required, while only 4 data samples are required to produce an *Interaction Information* $Q(X;Y;Z) = -2bit$ (pure *Redundancy*).
Hence, a *Redundancy* should rather occur in random data than *Synergy*.
For the pure *Synergy* 16 data samples are required because all values in the set *Mutual Information* set $\{I(X;Y), I(X;Z), I(Y;Z)\}$ have to be zero, while $\min(H(X), H(Y), H(Z)) = 2bit$ (every variable has at least 4 different values that it can take). $I(X;Y) = 0$ is only possible if all value combinations $(x_i, y_j), i, j \in \{1, ..., 4\}$ have the same frequency.

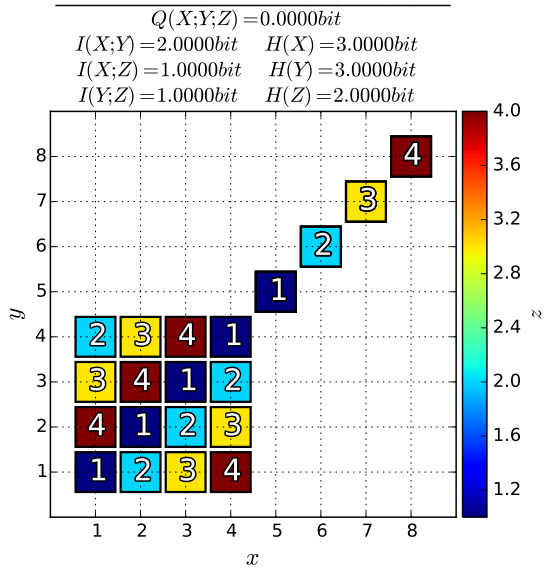

Figure 35: Samples of Dataset 3 with interval variables, where $f_{Syn} = 1$ and $f_{Red} = 4$

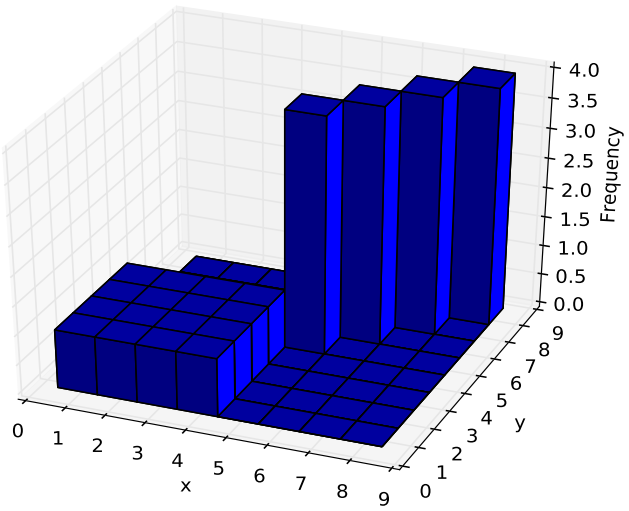

Figure 36: Histogramm of the input values of Dataset 3, where $f_{Syn} = 1$ and $f_{Red} = 4$

Dataset 3-1: Local pure Synergy:
In Dataset 3-1 which describes the local subspace $(1 \leq y \leq 4)$, X and Y are independent $(I(X;Y) = 0bit)$. The independence of X and Y can also be observed in Figure 37, where the samples are distributed in the whole subspace because the knowledge of X does not constrain the values of Y and vice versa.
The was manually designed to describe pure *Synergy*, which is a rare property in generated or real data. A pure *Synergy* describes a case, where the entropy of the output variable $H(Z)$ is only reduced if both input variables X and Y are known $(I(Z;X,Y) = Q(X;Y;Z) > 0)$. The *Mutual Information* of Z with one input variable alone is zero $(I(Z;X) = I(Z;Y) = 0)$. Dataset 3-1 is similar to a bitwise XOR operation, which is also an example for a pure *Synergy*.
The most important property of a pure *Synergy* is that every pairwise *Mutual Information* is zero $(I(X;Y) = I(X;Z) = I(Y;Z) = 0)$. This property requires that in every row and in every column the frequency of all values that Z can take has to be equal.

Dataset 3-2: Local pure Redundancy:

In Dataset 3-2 which describes the local subspace ($5 \leq y \leq 8$), X, Y and Z are strongly dependent (see Figure 38, $I(X;Y) = I(X;Z) = I(Y;Z) = 2bit$), which describes a pure *Redundancy* (see also Dataset 1-2).

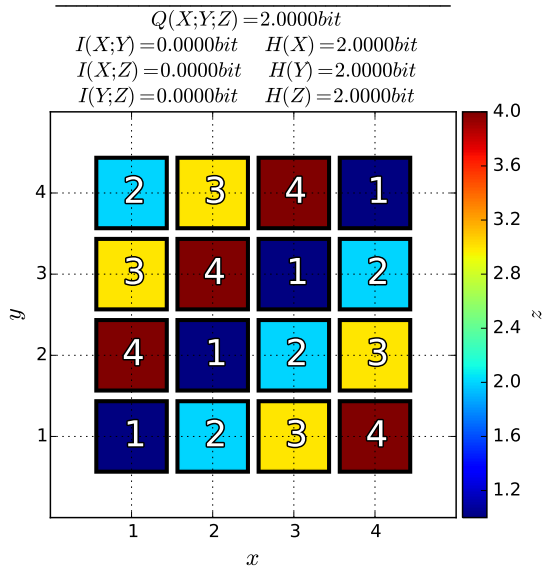

Figure 37: Samples of the feature subspace Dataset 3-1, which consists of all the samples from Dataset 3, which fulfill $1 \leq x \leq 4$.

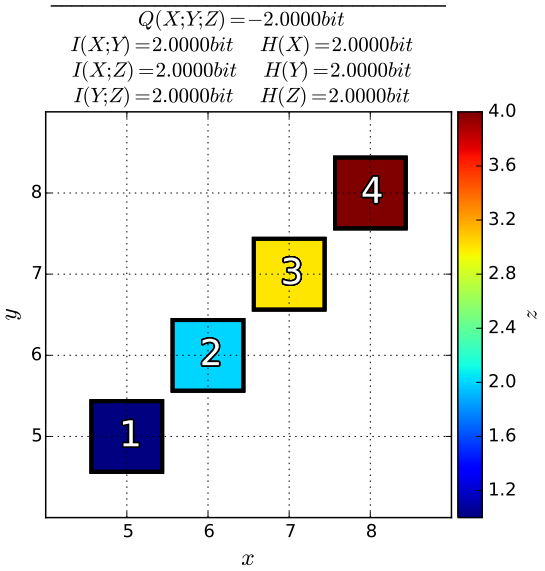

Figure 38: Samples of the feature subspace Dataset 3-1, which consists of all the samples from Dataset 3, which fulfill $5 \leq x \leq 8$.

4.2 Case Studies

4.2.1 Information Maps

In this section the *Information Maps* that were introduced in Section 3.1.2 are created for the three datasets from Section 4.1. The *Information Maps* depict the information share that is caused by all probabilities that correspond to the local values x or y or both. The sum of all information shares from all locations on the map is the entire information.
The maps were created for the *Mutual Information*, the *Conditional Mutual Information* and for the *Interaction Information*. Actually, we are interested in the *Interaction Information*, however the *Interaction Information Map* is sometimes hard to analyze compared to the *Mutual Information Map* and the *Conditional Mutual Information Map*. If the *Mutual Information Map* is subtracted from the *Conditional Mutual Information Map* we obtain the *Interaction Information Map*, hence, it can be helpful to analyze the *Mutual Information Map* and the *Conditional Mutual Information Map* if the shares of the *Interaction Information Map* do not lead to sensible conclusions.

Mutual Information Maps
The *Mutual Information Maps* are easy to analyze because they do not consider a third variable Z. A share $i(x,y)$ on the map at position (x,y) is higher if the value x occurs more probable in combination with the value y than in combination with other values and vice versa. If the share $i(x,y)$ is zero, the combination of x and y is either as probable as the multiplication of their separate probabilities (independence of the values) or does not happen (probability of the combination is zero). If a share $i(x,y)$ is negative, a combination of x and y is less probable than the multiplication of their separate probabilities. Every negative $i(x,y)$ can only exist if even higher positive shares $i(x,y)$ at other locations exist. The sum of all shares $i(x,y)$ of the map is the *Mutual Information*. The *Mutual Information* can only become zero if all shares $i(x,y)$ on the map are zero, otherwise there exists an interdependency between the variables X and Y.
In Figure 39-41 the *Mutual Information Maps* of Dataset 1-3 are shown. They visualize the *Mutual Information Maps* shares $i(x,y)$ for all possible combination of x and y.
Value combinations (x,y) that are labeled with "+" have a positive share and value combinations (x,y) that are labeled with "-" have a negative share, while the value combinations (x,y) that have the maximum positive share are labeled with "++" and the value combinations (x,y) that have the minimum negative share are labeled with labeled with "- -". The value combinations (x,y) that have a share of zero are labeled with "0". The color of a (x,y) combination corresponds to the value of the share, which can be read from the color bar on the right.

In Figure 39 the highest shares $i(x,y)$ occur at the positions $(5,1)$, $(6,2)$ and $(7,4)$ that means that the diagonal chain of samples increases the *Mutual Information* at most. That is caused by the circumstance that the value y can be reconstructed from x and vice versa on the diagonal. In a similar manner y can be reconstructed from x in the area $(x,y), x \in \{8,9,10,11,12\}, y = 4$. However, $y = 4$ does only provide the information that x is not equal to 5, 6 and 7. Thus, the shares are smaller. In the areas $1 \leq x \leq 4$ and $13 \leq x \leq 16$ the variables X and Y are almost independent because the 16 samples in each of the areas occur with the same frequency. But the shares are not zero because they are influenced from the area $5 \leq x \leq 12$. In that area all shares at $y \in \{1,2,3\}$ have the same small positive value and all shares at $y = 4$ have a small negative value and x and y can not be reconstructed from each other. But the probability that x lies in the areas $1 \leq x \leq 4$ or $13 \leq x \leq 16$ is smaller in the case $y = 4$ than in each of the cases $y \in \{2,3,4\}$ because row $y = 4$ has the most samples. For example, knowing $x = 4$ means that the probability of $y = 4$ is smaller than the probability of $y = 3$, which is a slight information.
The value combinations that are labeled with zero do not occur in Dataset 1.

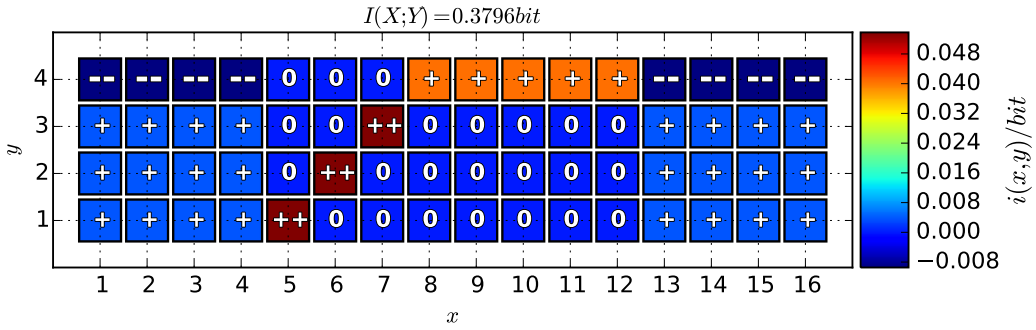

Figure 39: Mutual Information Map of Dataset 1

In Figure 40 all shares $i(x, y)$ are zero because X and Y are independent.

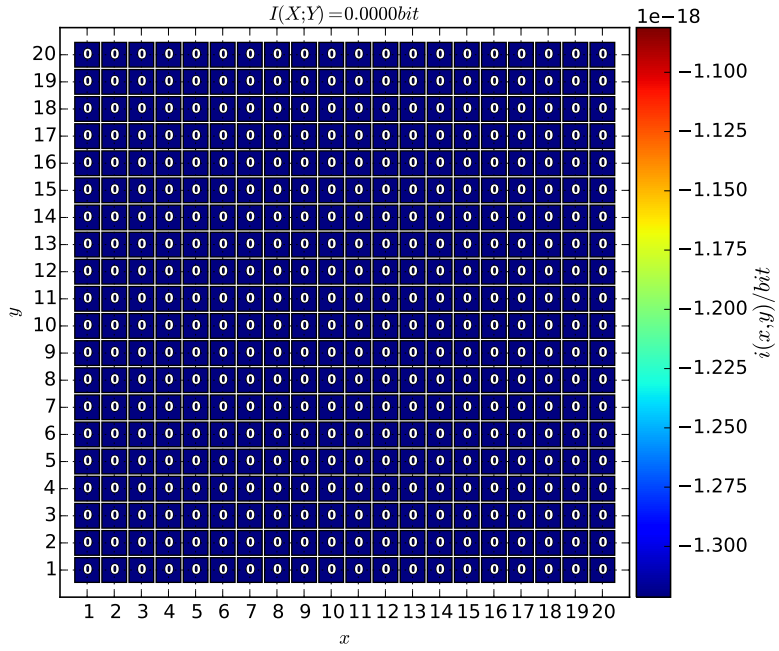

Figure 40: Mutual Information Map of Dataset 2

In Figure 41 the highest shares $i(x, y)$ occur at the positions $(5,5)$, $(6,6)$, $(7,7)$ and $(8,8)$ that means that the diagonal chain of samples increases the *Mutual Information* at most. That is caused by the circumstance that the value y can be reconstructed from x and vice versa on the diagonal. Positive shares can also be found in the area $x, y \in \{1,2,3,4\}$. The knowledge of $x \in \{1,2,3,4\}$ allows to infer that $y \in \{1,2,3,4\}$ and vice versa, which describes a small shared information. The value combinations that are labeled with zero do not occur in Dataset 3.

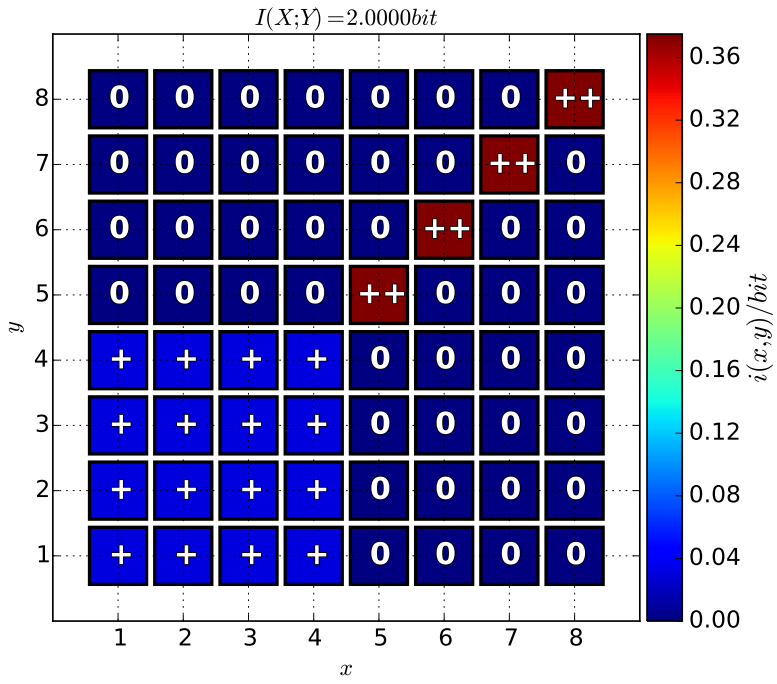

Figure 41: Mutual Information Map of Dataset 3

Conditional Mutual Information Maps

The *Conditional Mutual Information Maps* are not as easy to analyze as the *Mutual Information Maps* because they consider the third variable Z while there is no third dimension on the map visible. If Z can take q values $z_k, k = 1, ..., q$, we can take out a subset k of data samples from the dataset, where the variable Z takes the value z_k. For subset k we can calculate the *Mutual Information Map* k. We can calculate q separate *Mutual Information Maps*. If every *Mutual Information Map* k is weighted with the probability of z_k then we can sum up the q *Mutual Information Maps* to obtain the *Conditional Mutual Information Map*. The sum of all shares $i_z(x, y)$ of the map is the *Conditional Mutual Information*. *Conditional Mutual Information* has at least the value zero, because it is the sum of positive weighted *Mutual Information Maps*. In Figure 42-44 the *Conditional Mutual Information Maps* of Dataset 1-3 are shown.
Value combinations (x, y) that are labeled with "+" have a positive share and value combinations (x, y) that are labeled with "-" have a negative share, while the value combinations (x, y) that have the maximum positive share are labeled with "++" and the value combinations (x, y) that have the minimum negative share are labeled with labeled with "- -". The value combinations (x, y) that have a share of zero are labeled with "0". The color of a (x, y) combination corresponds to the value of the share, which can be read from the color bar on the right.

In Figure 42 the highest shares are at the (x, y)-positions $(1, 4)$, $(2, 2)$, $(4, 1)$, $(3, 4)$, $(4, 3)$ and $(6, 2)$. Because the z values are not shown in the map, we have to look them up in Figure 27 on page 62. At the (x, y)-positions $(1, 4)$, $(2, 2)$ and $(4, 1)$ the value of z is 4, while the value of z is 12 at the positions $(3, 4)$, $(4, 3)$ and $(6, 2)$. The values $z = 4$ and $z = 12$ each occur three times in the data, while the z values at the (x, y)-positions, which are labeled with "+" occur twice in the data (the z-values are 2, 3, 6, 8, 16, 32, 48). The (x, y)-positions that are labeled with "0" correspond either to samples that do not occur in the data or to samples with z-values that occur only once in the data. The *Conditional Mutual Information Map* is a sum of q *Mutual Information Maps* weighted with $p(z_k)$, $k = 1, ..., q$, where every map k is created for the data samples with $z = z_k$ and q is the number of discrete values that Z can take. Therefore, every *Mutual Information Map* k that corresponds to a value z_k that occurs only once in the data, is a *Mutual Information Map* of one sample, which has only shares that are equal to zero

because the entropies $H(X|Z=z_k)$ and $H(Y|Z=z_k)$ are zero. The knowledge of z_k allows to reconstruct the values of x and y in that case. An example would be the (x,y)-position $(10,4)$ with $z=40$, which occurs only once.
Every *Mutual Information Map* k that corresponds to a value z_k that occurs twice in the data is a *Mutual Information Map* of two samples (e.g. the (x,y)-positions $(2,3)$ and $(3,2)$ with $z=6$). Because the z-pairs have never the same x and y in Dataset 1, the entropies $H(X|Z=z_k)$ and $H(Y|Z=z_k)$ are $1bit$ and can be reconstructed from each other $I(X;Y|Z=z_k)=1bit$ $(x=2 <=> y=3, x=3 <=> y=3$, given $z=6)$. Therefore, the *Mutual Information Map* k has two shares equal to $0.5bit$ at the (x,y)-positions of the two samples. In the *Conditional Mutual Information Map* the shares occur weighted with $p(z_k)$.
Every *Mutual Information Map* k that corresponds to a value z_k that occurs twice in the data is a *Mutual Information Map* of two samples (e.g. the (x,y)-positions $(2,3)$ and $(3,2)$ with $z=6$). Because the z-pairs have never the same x and y in Dataset 1, the entropies $H(X|Z=z_k)$ and $H(Y|Z=z_k)$ are $1bit$ and can be reconstructed from each other $I(X;Y|Z=z_k)=1bit$ $(x=2 <=> y=3, x=3 <=> y=3$, given $z=6)$. Therefore, the *Mutual Information Map* k has two shares equal to $0.5bit$ at the (x,y)-positions of the two samples. In the *Conditional Mutual Information Map* the shares occur weighted with $p(z_k)$.
Every *Mutual Information Map* k that corresponds to a value z_k that occurs three times in the data is a *Mutual Information Map* of three samples (e.g. the (x,y)-positions $(3,4)$, $(4,3)$ and $(6,2)$ with $z=12$). Because the z-triples have never the same x and y in Dataset 1, the entropies $H(X|Z=z_k)$ and $H(Y|Z=z_k)$ are $log_2(3)bit$ and can be reconstructed from each other $I(X;Y|Z=z_k)=log_2(3)bit$ $(x=3 <=> y=4, x=4 <=> y=3, x=6 <=> y=2$, given $z=12)$. Therefore, the *Mutual Information Map* k has three shares equal to $\frac{1}{3}log_2(3)bit$ at the (x,y)-positions of the three samples. In the *Conditional Mutual Information Map* the shares occur weighted with $p(z_k)$.
Because only a particular z_k occurs at a certain (x,y)-position in Dataset 1, every share of the *Conditional Mutual Information Map* is only influenced by one *Mutual Information Map* k.
The positive shares of the *Conditional Mutual Information Map* correspond to the (x,y)-position, where on the one hand the knowledge of the value z is not sufficient to reconstruct the values x and y and on the other hand the knowledge of x helps to reconstruct y and vice versa. Higher shares are caused by a higher number of possible x and y values for a given value of z and a good capability to reconstruct x from y and vice versa.

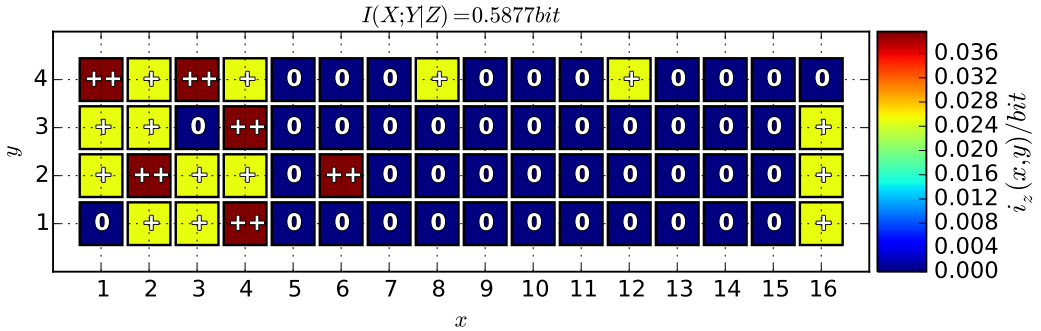

Figure 42: Conditional Mutual Information Map of Dataset 1

In Figure 43 the *Conditional Mutual Information Map* of Dataset 2 has the highest shares at the (x,y)-positions, where $11 \leq x+y \leq 21$ and $1 \leq y \leq 9$ (diagonal chains of "++"), correspond to the z values that occur at most in Dataset 2 e.g. $z=30$ (see also Figure 32 on page 65). The value $z=30$ occurs at the diagonal chain of (x,y)-positions $y=20-x, 1 \leq y \leq 9$ and on the vertical chain $x=10, 10 \leq y \leq 20$. Because $z=30$ is one of the z values that occurs at most, the potential for higher shares is given because more samples are available for the given z, which means that the entropy of X and Y given z can be

higher. In the area $1 \leq y \leq 9$, if $z = 30$ is known, 9 possible cases of x and y are left. Because x and y can be reconstructed from each other, the shares, which are lying on the diagonal chain ("++"), are high. But on the vertical chain ($x = 10, 10 \leq y \leq 20$) y can not be reconstructed from $x = 10$. The information of $x = 10$ knowing $z = 30$ does only provide the information that $y \geq 10$, therefore, the shares are slightly higher than zero.
The rest of the shares on the *Conditional Mutual Information Map* are created in the same manner, but the frequency of the corresponding z values is smaller. Therefore, the shares are smaller due to smaller entropies $H(X|Z = z)$ and $H(Y|Z = z)$ which limit the information that is shared by X and Y.

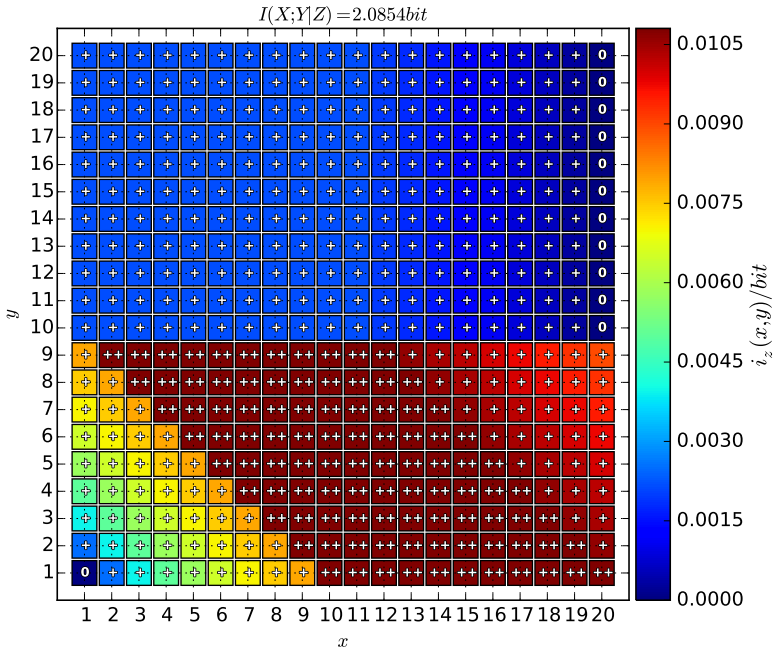

Figure 43: Conditional Mutual Information Information Map of Dataset 2

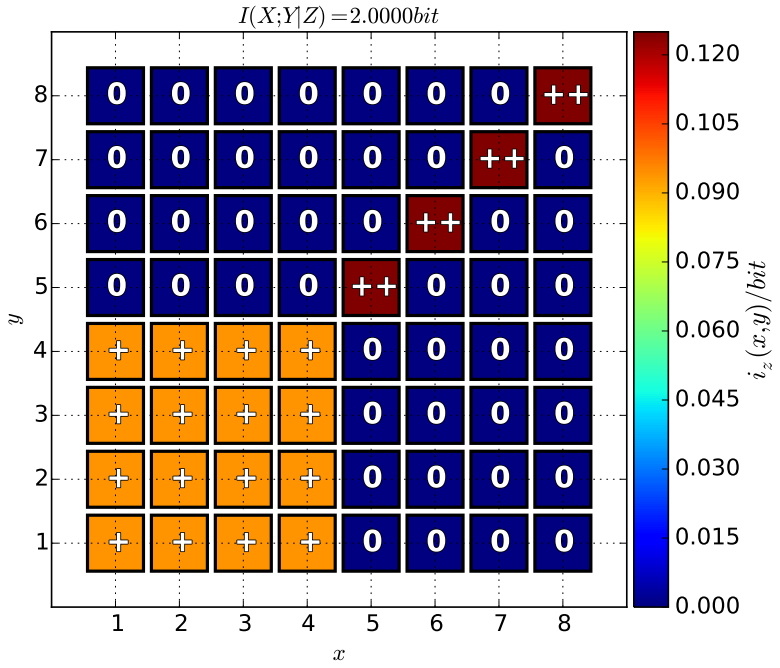

Figure 44: Conditional Mutual Information Map of Dataset 3

In Figure 44 the *Conditional Mutual Information Map* shows only positive shares, where the shares in the area $x = y \in \{5,6,7,8\}$ are higher because the frequency of the samples in that area is four, while the remaining samples occur only once in Dataset 3. Every possible value $z \in \{1,2,3,4\}$ occurs with the same frequency (see Figure 35). For a given z, five (x,y)-positions are left, where the five x and y values can be reconstructed from each other. Therefore, all shares at (x,y)-positions, where at least one sample exists are positive.

Interaction Information Maps

The *Interaction Information Maps* consider the third variable Z, while there is no third dimension on the map visible. A *Interaction Information Map* can be obtained by subtracting the *Mutual Information Map* from the *Conditional Mutual Information Map*. The sum of all shares $q_z(x,y)$ of the *Interaction Information Map* is the *Interaction Information*, which can be positive, negative or zero. In Figure 45-47 the *Conditional Mutual Information Maps* of Dataset 1-3 are shown.

Value combinations (x,y) that are labeled with "+" have a positive share and value combinations (x,y) that are labeled with "-" have a negative share, while the value combinations (x,y) that have the maximum positive share are labeled with "++" and the value combinations (x,y) that have the minimum negative share are labeled with labeled with "- -". The value combinations (x,y) that have a share of zero are labeled with "0". The color of a (x,y) combination corresponds to the value of the share, which can be read from the color bar on the right.

In Figure 45 the highest shares of the *Interaction Information Map* correspond to the (x,y)-positions $(1,4)$ and $(3,4)$ the *Conditional Mutual Information Map* has the maximum share and the *Mutual Information Map* the minimum share at the same (x,y)-position. All values can be explained by the difference of the *Conditional Mutual Information Map* and the *Mutual Information Map*. Reading the *Interaction Information Map* allows us to observe at which (x,y)-positions the values x and y can better reconstructed from each other knowing the value of z compared to the case that z is unknown. For example, at (x,y)-position $(1,4)$ (see also Figure 27 on page 62 for the z values) the value $x = 1$ does not allow us to reconstruct y because for $x = 1$ y can take the maximum of possible values $y = \{1,2,3,4\}$, we have only the insignificant information that each of the cases $y = \{2,3,4\}$ is slightly more probable than $y = 1$. However, if $z = 4$ at the (x,y)-position $(1,4)$ is known, the values of x and y can be reconstructed from each other because Dataset 1 has only one case where $x = 1, y = 4, z = 4$. Hence, the positive shares indicate an increased capability to reconstruct x and y from each other caused by the additional knowledge of the z value.

In contrast at the (x,y)-position $(5,1)$ the share has the smallest value. That means that the knowledge of z decreases the capability to reconstruct x and y from each other. In our case knowing $z = 5$ allows to infer that $x = 5$ and $y = 1$ because the value z occurs only at exactly one (x,y)-position. If x and y are already known due to the knowledge of z, they can not be reconstructed from each other. But without knowing z the values of x and y can be reconstructed from each other. Because the cases $z = 5$ and $x = 5$ occur only once in Dataset 1, the same sample can be identified by only knowing one of both values, therefore, they provide redundant information.

The *Interaction Information Map* allows to see the shares of the *Interaction Information* for every (x,y), which is a more detailed information than only the sum of the shares. For example, in Figure 28 on page 64 was shown that the Interaction Information is high in the feature subspace $1 \leq x \leq 4$ of Dataset 1. But the *Interaction Information Map* uncovers that in the overall synergetic area also negative shares are included, which correspond to the (x,y)-positions $(1,1)$ and $(3,3)$.

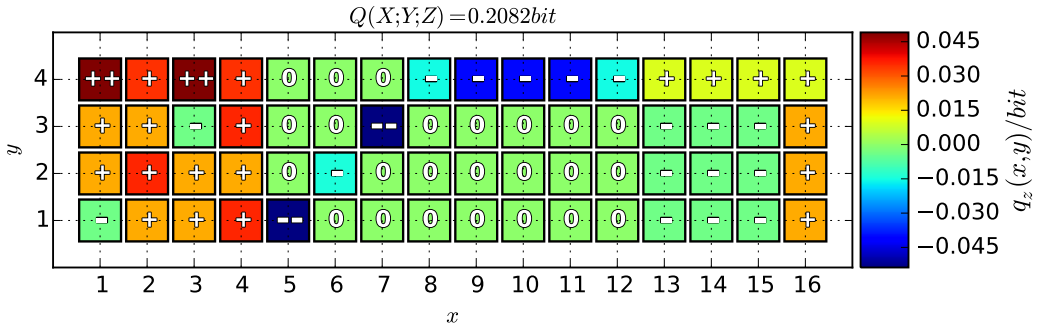

Figure 45: Interaction Information Map of Dataset 1

In Figure 46 the *Interaction Information Map* is equal to the *Conditional Mutual Information Map* because the *Mutual Information Map* has only shares equal to zero. In all areas where the shares are positive the information that the values x and y provide about each other could be increased by additional knowledge of z.

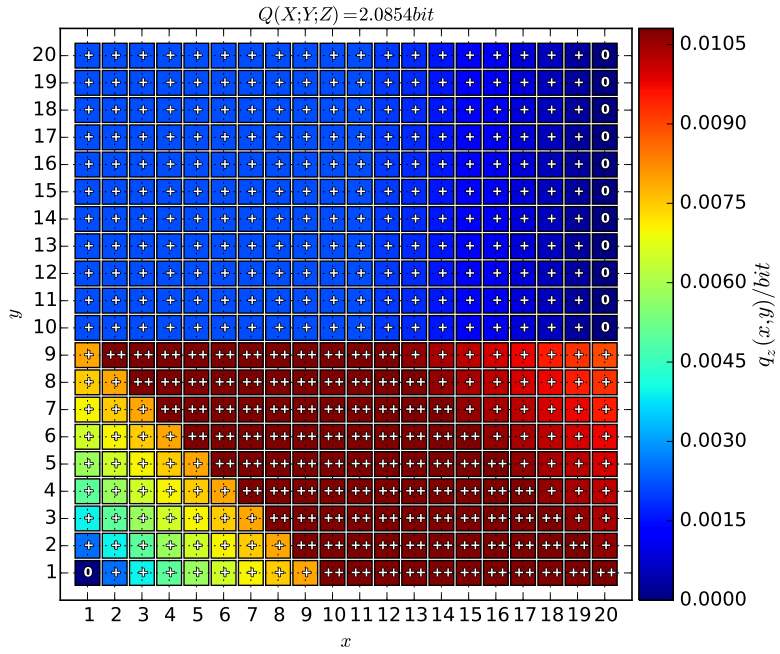

Figure 46: Interaction Information Map of Dataset 2

In Dataset 3 (Figure 47) the *Interaction Information Map* has a synergetic area ($1 \leq x, y \leq 4$) with shares of equal size due to its pure local *Synergy* and due to the pure local *Redundancy* the shares are also equal in the area ($1 \leq x = y \leq 4$). In the area ($1 \leq x, y \leq 4$) the knowledge of z is required in order to use the knowledge that the values x and y provide about each other (*Synergy*), consequently the shares are positive. But in the area ($1 \leq x = y \leq 4$) x and y can be reconstructed from each other without knowing z. If z is known the x and y can still be reconstructed from each other, but the remaining uncertainty of both is reduced, therefore the remaining information they can share is reduced due to the knowledge of z. For example, if $z = 3$, $x, y \in \{5, 6, 8\}$ is not longer possible and a large amount of information that x and y could share is redundant to the information that z provides (see Figure 35 on page 68).

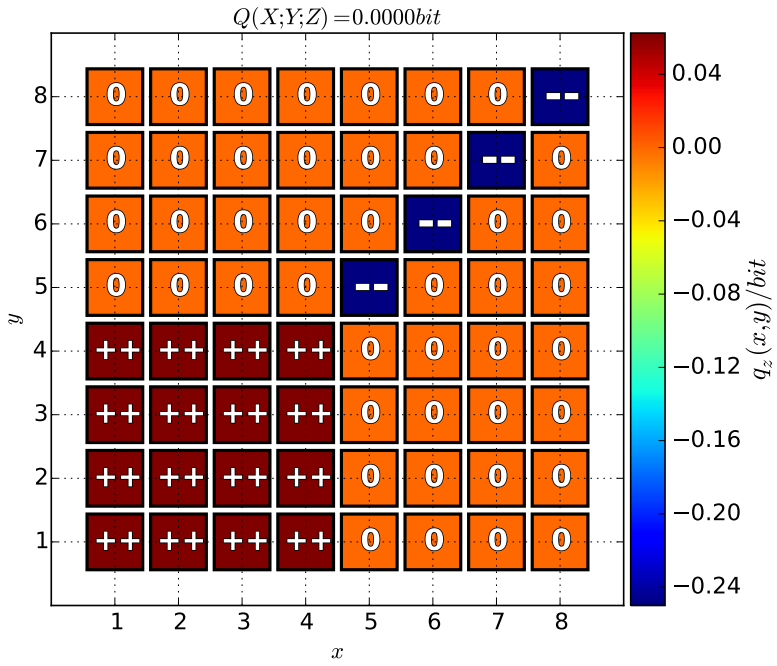

Figure 47: Interaction Information Map of Dataset 3

Interaction Information Maps for Feature Space Partitioning

The *Interaction Information* $Q(X;Y;Z)$ does not change if the variables X, Y and Z are interchanged. Assuming that the variables X and Y are the inputs and Z the output of a system, the goal is to predict the value z from the values x and y. Consequently, the feature space has the dimensions x and y. A possible goal of the feature space partitioning could be to find a feature subspace, where the *Synergy* of the contained data samples is maximized.

The *Interaction Information Map* provides exact local shares that allow to identify the locations which increase the overall *Interaction Information* at most. However, the local shares are dependent from all data samples. Separating a local feature subspace from the data means that data samples are removed. Consequently, the *Interaction Information Map* of the local feature subspace is not a local part of the *Interaction Information Map* that was created from the original dataset. Instead, the *Interaction Information Map* of the local feature subspace is different than the original map due to the missing data samples. It is possible that the *Synergy* is increased by removing the data samples, but its also possible that removed data samples would increase the *Synergy* in the current feature subspace.

In order to maximize the *Synergy* in one feature subspace an iterative method for a feature space partitioning based on the *Interaction Information Map* was developed (see Section 3.2.3). The method uses relaxed assignments (introduced in Section 3.2.1) of the data samples to the feature subspace. Relaxed means that the calculation of the *Discrete Probability Density Function* is modified in that way that it is able to handle data samples as if they only partly exist. For example, instead of removing a data sample completely, the influence of the data sample can be reduced continuously. The partitioning method increases the influence of the data samples that correspond to shares of the *Interaction Information Map* that are positive and decreases the influence of the data samples that correspond negative shares. After the adaptation of the influences, the *Interaction Information Map* is created again to provide the required information for the next adaptation of the influences. Using the iterative method avoids that data samples are removed if they are required for the *Synergy*. The method can also be used to find a feature subspace where the *Redundancy* is maximized. It would even be possible to use the method with the *Conditional Mutual Information Map* or the *Mutual Information Map* if a feature subspace with a maximized *Conditional Mutual Information* or *Mutual Information* was the goal.

4.3 Global vs. Local Models

4.3.1 Bayesian Classifier

The *Limited Dependence Classifier* (see Section 3.3.6) is a *Bayesian Classifier* that builds its own *Bayesian Network*. While building the Network, the *Limited Dependence Classifier* decides whether an arrow between two input variables X_1 and X_2 is reasonable or not by evaluating the *Conditional Mutual Information* $I(X_1;X_2|C)$. If $I(X_1;X_2|C)$ is below a certain threshold, a connection between X_1 and X_2 is assumed to be unnecessary. In Section 3.3.3 was shown that in a system with two inputs X_1 and X_2 an arrow between X_1 and X_2 is unnecessary if X_1 and X_2 are conditional independent ($I(X_1;X_2|C) = 0$). For a system with only two inputs, the structural difference between a *Naive Bayesian Classifier* and a *Full Bayesian Classifier* is only the arrow between X_1 and X_2.
In this experiment we want to test whether the *Interaction Information* is an alternative measure for the requirement of an arrow between X_1 and X_2 in the *Bayesian Network* structure.

Dataset 3 from Section 4.1 has been used for the following experiments. The parameters f_{Syn} and f_{Red}, which are describing the the frequency of the samples in the *Synergy* and *Redundancy* part of Dataset 3, have been replaced by the parameters $s = f_{Syn}$ and $r = 4 \cdot f_{Red}$. Consequently, for the case $r = s$, *Synergy* and *Redundancy* compensate each other. For $r > s$ the *Interaction Information* gets negative and for $s > r$ it gets positive values.
Note that for $f_{Syn} = f_{Red} = 1$ the dataset would have 16 samples that represent *Synergy* and only 4 samples that represent *Redundancy*. Hence, for a compensation $f_{Red} = 4 \cdot f_{Syn}$ is required.

In order to produce different *Interaction Information* values, the ratio between r and s has been varied. The values for different ratios have been generated in several ways to get a high resolution of values in different areas of the *Interaction Information* between -2bit and 2bit.
Figure 48 shows the values that are generated for $r = 1$ and variations of s between 0 and 100. This allows to collect many values about the area where the *Interaction Information* is greater zero, while the area where it is negative is very small. In order to collect more values, where the *Interaction Information* is negative, more values have been generated for $r = 100$ and a varied s between 0 and 100 as shown in Figure 49. The same procedure was made with constant values of $s = 1$ and $s = 100$ and a varied r in order to see whether the values can be produced in that way, too (see Figure 50 and 51).
By merging all values (accuracy, *Interaction Information* and *Conditional Information*) from the curves of Figure 48 to 51 in one list, the values can be drawn for every given *Interaction Information* as shown in Figure 52. Smooth and accurate curves in every area of the plot are the result of the merge of the support points of the previous curves.

Discussion of the result

The difference between the *Full Bayesian Classifier* and *Naive Bayesian Classifier* of two input variables is that a *Full Bayesian Classifier* models the interdependencies between X_1 and X_2, while the *Naive Bayesian Classifier* models X_1 and X_2 as conditional independent ($I(X_1;X_2|C) = 0$). Hence, if there is an information in X_1 about C that is only useful while knowing X_2, the *Naive Bayesian Classifier* is not able to take this synergetic information into account.

As shown in Figure 52 the *Conditional Mutual Information* $I(X_1;X_2|C)$ measures indeed the requirement of an arrow connection between X_1 and X_2 because the accuracy of the *Naive Bayesian Classifier* decreases with the increasing *Conditional Mutual Information* $I(X_1;X_2|C)$ at first. However, at the end of the curve, where $Q(X_1;X_2;C) > 1.25$, the *Conditional Mutual Information* $I(X_1;X_2|C)$ is decreasing again in spite of a still decreasing accuracy of the *Naive Bayesian Classifier*.
The *Interaction Information* $Q(X_1;X_2;C)$ seems to be a better measure for the the requirement of an

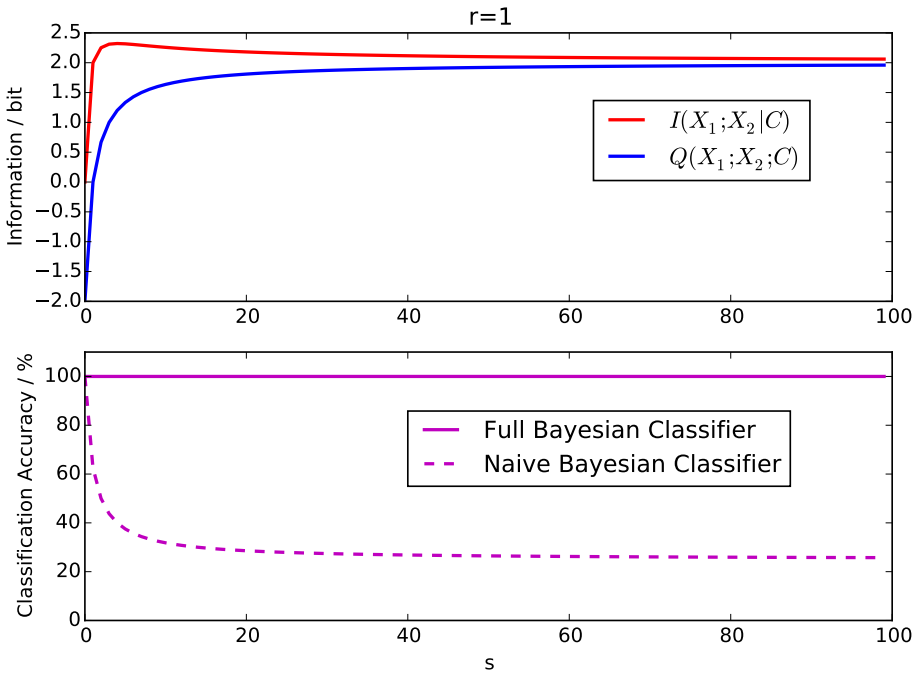

Figure 48: Classification accuracy of the *Full* and *Naive Bayesian Classifier* tested on training data. The training data and the corresponding *Interaction Information* and *Conditional Information* vary with the parameter s. The training data is Dataset 3 with the parameters $s = f_{Syn}$ and $r = 4 \cdot f_{Red}$, where $r = 1$ and s is varied.

arrow connection between X_1 and X_2 because the accuracy of the *Naive Bayesian Classifier* decreases linearly with the increasing *Interaction Information* $Q(X_1;X_2;C)$ across the complete range.
Hence, we can expect an slight improvement of the *Limited Dependence Classifier* if the *Conditional Mutual Information* criterion is replaced by the *Interaction Information* criterion.
Apart from that, we can conclude that partitioning of the feature space into a subspace with a dominant *Synergy* and another subspace with a dominant *Redundancy* is reasonable in order to use the simpler *Naive Bayesian Classifier* in the *Redundancy* subspace and the more complex *Full Bayesian Classifier* in the *Synergy* subspace.

However, the *Full Bayesian Classifier* has no generalization capability because it learns the probability of every value combination of the training data, while it assumes that every value combination that is not known from the training data is zero. The generalization capability of a *Bayesian Classifier* is activated if at least one arrow is removed from the *Full Bayesian Network*. The more arrows are removed the better the generalization capability. Usually, the maximum removement of arrows leads to a *Naive Bayesian Classifier*. In a feature space with only two dimensions the difference of the simplest and the most complex network is only one arrow.
Because we are still working with a feature space of only two dimensions, we can not compare two local *Bayesian Classifiers* of different complexities without being forced to use a *Full Bayesian Classifier* as one of both local models. Consequently, no reasonable generalization experiment with test data that contain different input value combinations than the training data can be performed.

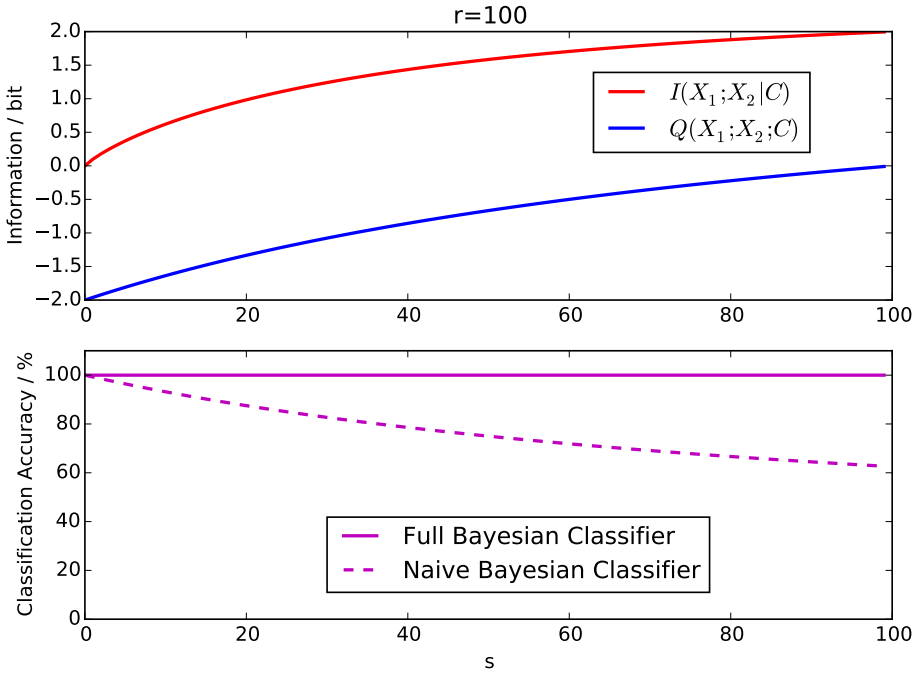

Figure 49: Classification accuracy of the *Full* and *Naive Bayesian Classifier* tested on training data. The training data and the corresponding *Interaction Information* and *Conditional Information* vary with the parameter s. The training data is Dataset 3 with the parameters $s = f_{Syn}$ and $r = 4 \cdot f_{Red}$, where $r = 100$ and s is varied.

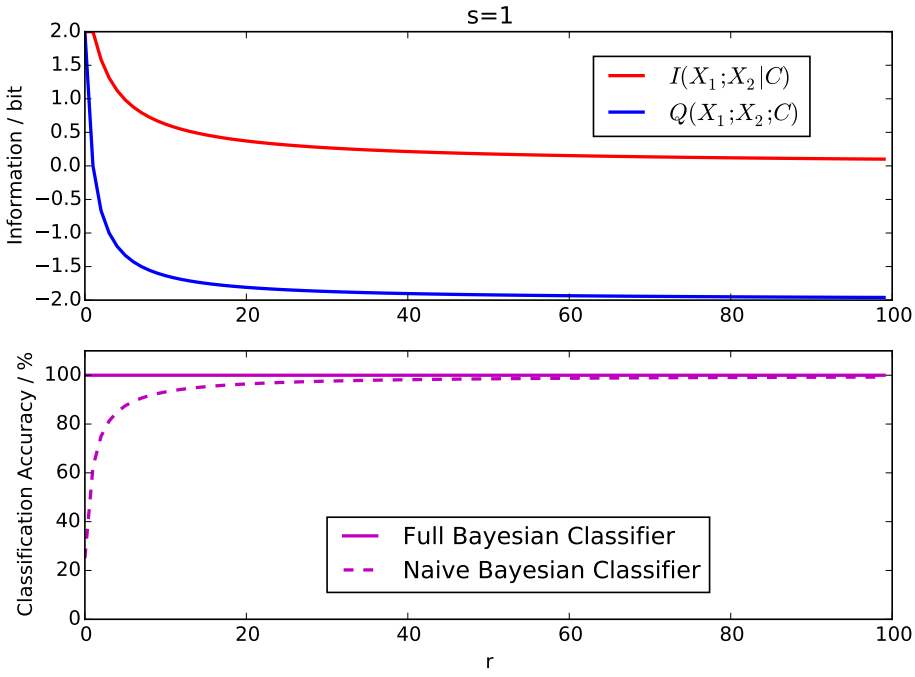

Figure 50: Classification accuracy of the *Full* and *Naive Bayesian Classifier* tested on training data. The training data and the corresponding *Interaction Information* and *Conditional Information* vary with the parameter r. The training data is Dataset 3 with the parameters $s = f_{Syn}$ and $r = 4 \cdot f_{Red}$, where $s = 1$ and r is varied.

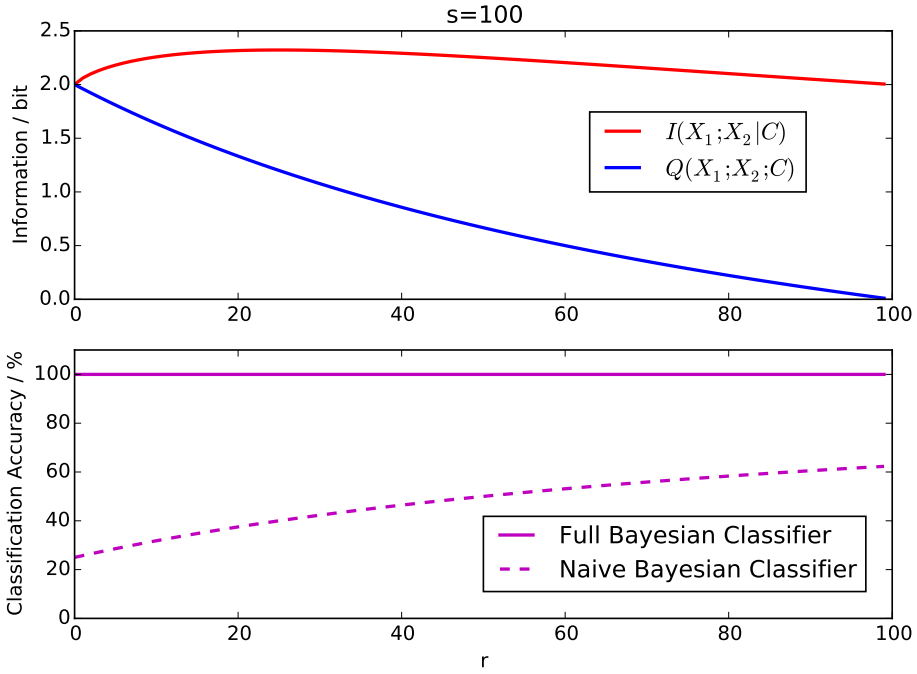

Figure 51: Classification accuracy of the *Full* and *Naive Bayesian Classifier* tested on training data. The training data and the corresponding *Interaction Information* and *Conditional Information* vary with the parameter r. The training data is Dataset 3 with the parameters $s = f_{Syn}$ and $r = 4 \cdot f_{Red}$, where $s = 100$ and r is varied.

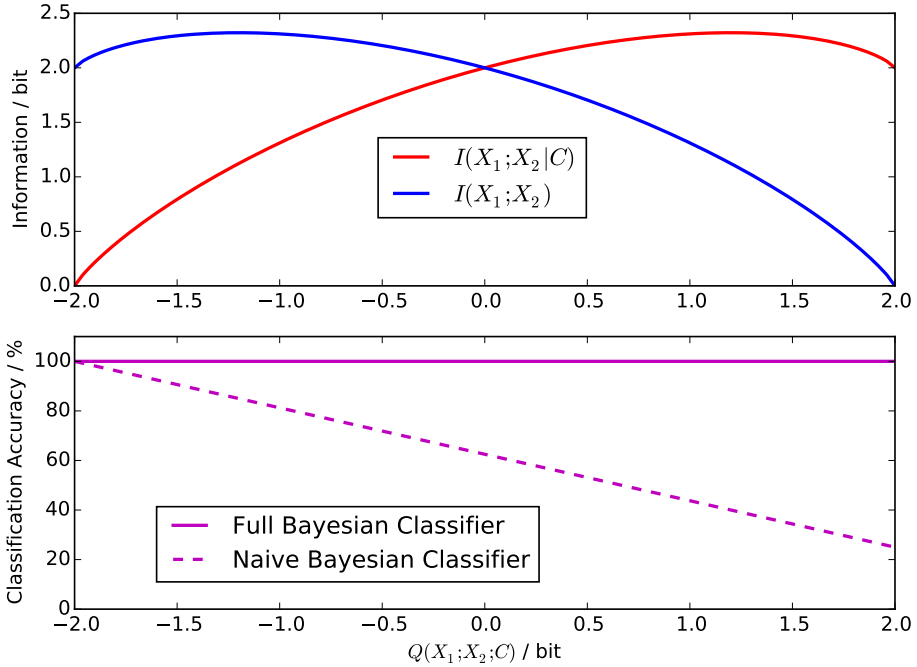

Figure 52: Classification accuracy of the *Full* and *Naive Bayesian Classifier* tested on training data. The training data and the corresponding *Conditional Information* and *Mutual Information* vary with the *Interaction Information*. The corresponding values for a given *Interaction Information* were taken from the curves of Figure 48 to 51. Note: The *Mutual Information* is simply the difference between the *Conditional Information* and the *Interaction Information*.

4.3.2 Multilayer Perceptron, Linear Perceptron, Classification Tree

The purpose of the feature space partitioning is to find a feature subspace where a local model that considers interactions of two inputs is required and a second feature subspace where a simpler local model with only one input is required. A simpler model is excepted to provide a improved generalization capability (see Section 3.3 for more details).
In this experiment a *Linear Single-Layer Perceptron* or *Linear Perceptron (LP)* and a *Multilayer Perceptron (MLP)* are used as models (see Section 3.3.8 and 3.3.9 for details about the models).
The generalization capability of a global model is compared with the generalization capability of two local models of different complexity on Dataset 2 (see Figure 32).

Global Model:
A global model means the common case, where one model is responsible for the whole feature space. The models (LP and MLP) are used as they are usually used. Because the feature space of Dataset 2 has two dimensions, the global models have two inputs.

Local Models:
The local models are responsible for different parts of the feature space. The feature space is partitioned in two local feature subspaces and for every feature subspace one local model is responsible. The feature space of Dataset 2 was manually partitioned into **Feature Subspace 1 (FSS1)** a synergetic feature subspace, where the inputs interact (see Figure 33), and **Feature Subspace 2 (FSS2)**, where the inputs do not interact and only one input is important (see Figure 34).
For the LP both local models are equal to the global model because the LP removes one input autonomous while the training, hence, the model simplifies its structure while the training in FSS2. Because the training of the LP is not iterative, the training process is not harmed by the second input. In FSS1 both inputs are required and the structure of the LP considers two inputs after the training.
In the MLP the second input (i.e. y) of the simpler local model, that corresponds to FSS2 was manually removed before the training. The decision of a local feature selection is discussed in Section 3.2. The local MLP model that corresponds to FSS1 has two inputs like the global model.

Comparison of the generalization capability
In order to compare the the generalization capabilities, learning curves of the models were determined using the following steps:
1) The samples of Dataset 2 are shuffled.
2) The shuffled Dataset 2 are divided in $K = 10$ folds, where the number of data samples in every fold is equal (40 samples per fold).
3) For every fold $k = 1, ..., K$ steps 3.2) and 3.3) are performed.
3.2) Fold k is defined as test data and the remaining folds as training data.
3.3) The model is trained with 14 different fractions of the training data. For every fraction the prediction errors of the model on the test data are collected. The pairs of local models are treated as one model that is responsible for the entire feature space, therefore, their errors are collected together. The root-mean-square error (RMSE) is calculated for every fraction for the training data separately. The result is **one** learning curve, with the % of used training data (fractions) on the x-axis and the RMSE on the y-axis. Because the results of the MLP are influenced by a random initialization of its weights (parameters), 10 learning curves are determined per fold in order to get a more representative result.
4) The learning curves in the following Figures (e.g. Figure 57) are depicting the **mean** of the **100 learning curves** and the corresponding **standard deviation**.

The RMSE is defined as

$$RMSE(\mathbf{e}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(e_i)^2},\ \ \mathbf{e} = (e_1, ..., e_n)^T,\ \ e_i = (\hat{z}_i - z_i)\ \forall\ i = 1, ..., n \tag{221}$$

where n is the number of samples in the test data, $(\hat{z}_1, ..., \hat{z}_n)$ are the output values that the model has predicted for the input values from the test data and $(z_1, ..., z_n)$ are the correct output values from the test data and $(e_1, ..., e_n)$ are the prediction errors.

Training of the models
The LP was exactly trained and implemented as discussed in Section 3.3.8.
For the MLP the python implementation of the *pybrain* package from http://www.pybrain.org/ [21] was used. The training was performed using the backpropagation method. The structure and the learning parameters were adjusted by trial and error to achieve the best results with the global MLP. The finally used learning parameters are a learning rate of 0.001 and an iteration number of 200. The final model structure for two inputs is shown in Figure 23. The structure for one input, which is used for the model that is responsible for FSS2, is depicted in Figure 24.

Results for a feature space partitioning into a local Synergy and a local Non-Interaction

Results of the global and local LPs
In Figure 53 the learning curve of a global LP and the learning curve of a pair of local LPs are depicted. Because the variables in Dataset 2 have the non-affine relation $z = x + min(y, 10)$ a global LP is not able to perform correct predictions on the entire feature space. The global LP model is fitted as good as possible into the non-affine relation. However, if more than 10% of training data are available the parameters have approximately achieved their best values ($w_1 = 1, w_2 = 0.4624, w_3 = 12.8947$) and the value of z is approximated using $\hat{z} = w_1 \cdot x + w_2 \cdot y + w_3$.
The non-affine relations in Dataset 2 are caused by a saturation of the influence of y. Due to the saturation the interaction of X, Y and Z does not exist for $y > 10$. Hence, the partitioning of feature space in the synergetic FSS1 and the non-interacting FSS2 leads to two feature subspaces, where no saturation occurs and the relations of the variables are affine (linear with offset). In FSS1 the relation is $z = x + y + 10$ and in FFS2 the relation is $z = x + 20$. Therefore, the local LPs are fitting perfectly into the data using the model $\hat{z} = w_1 \cdot x + w_2 \cdot y + w_3$ with the parameters ($w_1 = 1, w_2 = 1, w_3 = 10$) in FSS1 and the parameters ($w_1 = 1, w_2 = 0, w_3 = 20$) in FSS2. As we can conclude from the parameter $w_2 = 0$ of the model in FSS2, the feature y is locally removed after the training of the local model. The perfectly suitable local models are the reason why the learning curve of the local LP is constantly zero. Only a few data samples of an affine feature subspace are required to determine the parameters of the LP model because Dataset 2 is not noisy. However, if the data was noisy only a few more samples would be required for learning very good parameters. In real datasets a saturation would not be that sharp but the RMSE would nevertheless be small even if the relations of the variables are not perfectly affine. Even the RMSE of the global model is good considering that the affine model is not suitable for the non-affine relation.

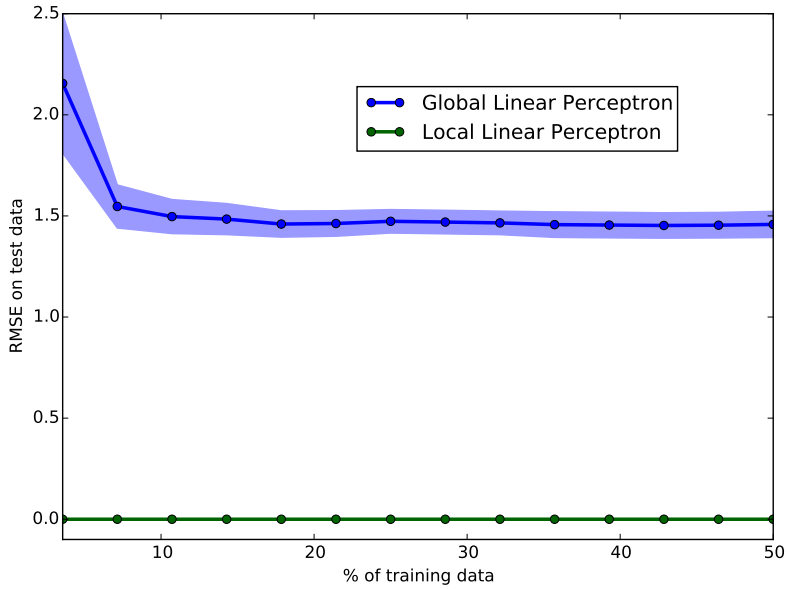

Figure 53: Learning curves of the Linear Perceptron on Dataset 2 using a global model and a pair of local models. Each local model corresponds to one of the both feature subspaces (see Figure 33 and 33). The training data is 90% of the entire data. The x-axis depicts the % of training data that was used to achieve the RMSE values on the y-axis. The curve that connects the dots is the mean value of the achieved RMSE values, while the brighter area in a similar color as the mean curve represents the standard derivation.

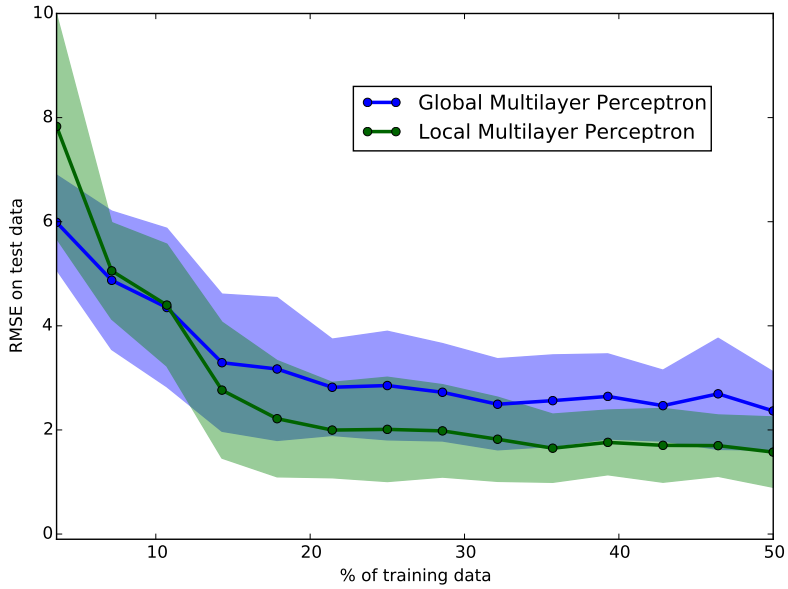

Figure 54: Learning curves of the Multilayer Perceptron on Dataset 2 using a global model and a pair of local models. Each local model corresponds to one of the both feature subspaces (see Figure 33 and 33). The training data is 90% of the entire data. The x-axis depicts the % of training data that was used to achieve the RMSE values on the y-axis. The curve that connects the dots is the mean value of the achieved RMSE values, while the brighter area in a similar color as the mean curve represents the standard derivation.

Results of the global and local MLPs

In Figure 54 the learning curve of a global MLP and the learning curve of a pair of local MLPs are depicted. Because only the half of the training data corresponds to every feature subspace, on average the local models obtain only the half number of available training samples. Therefore, the RMSE of the local

models is higher than the RMSE of the global model if less than 11% of the training data is available. Providing more than 11% of the training data the local MLPs achieve a smaller RMSE value due to a improved generalization capability of the simpler local MLP in FSS2.
If the combination of the two local models is treated as one model the whole number of parameters is higher than the number of parameters in the global model. Therefore, more training data samples are required to fit the model correctly into the data.
We can **conclude** that a partitioning of the feature space and the use of two local models of different complexity instead of a single global model can improve the generalization capability.

Results for a Decision Tree Classifier partitioning

In order to compare the previously used manual partitioning with a Decision Tree Classifier partitioning, the previous experiments are repeated with a Decision Tree Classifier partitioning.
The Decision Tree Classifier is partitioning the feature space in order to continue the further classification in a local feature subspace (see Section 3.3.10 for details). If the rest of the tree that corresponds to the first feature subspace is considered as local submodel 1 and the rest of the tree that corresponds to the second feature subspace is considered as local submodel 2, the approach is similar to our approach which was previously used. However, the partitioning criterion is quite different. In our approach the goal is to use two local models of different complexity by separating local *Synergy* from local *Redundancy* or from local *Non-Interaction*. The Decision Tree Classifier aims at a partitioning where the complexity of the relations in both feature subspaces is similar.
In Figure 55 the feature space partitioning of a Decision Tree Classifier for Dataset 2 is shown. For the Decision Tree Classifier the python implementation of the *sklearn* package from *http://scikit-learn.org/* [15] was used. The used split criterion was the *Information Gain*, which led to the same result as the default split criterion *Gini Impurity*. The maximum depth of the tree was set to 1. A higher maximum depth does not influence the partitioning at the first branch. In contrast to the manual partitioning the boundary is turned by 90 degrees. The blue area ($x \leq 10$) is the first feature subspace and the red area ($x > 10$) is the second feature subspace. In contrast, the manual partitioning used a y value as boundary with the feature subspaces ($y \leq 10$) and ($y > 10$).

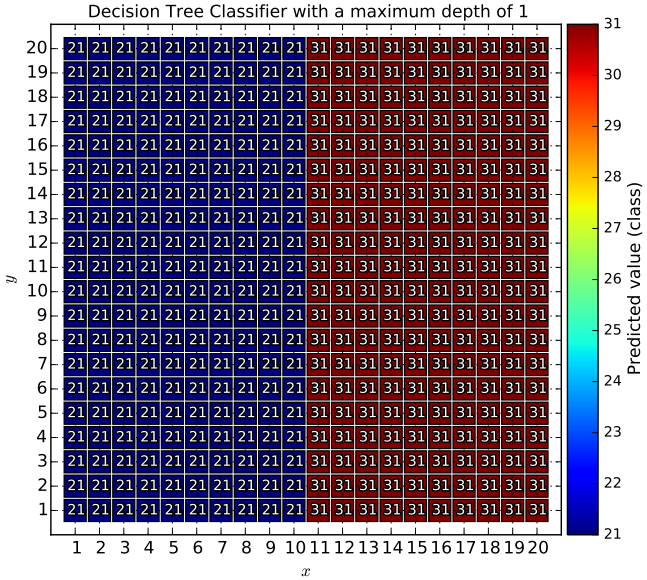

Figure 55: Partitioning of Dataset 2 using a Decision Tree Classifier with a maximum depth of 1. The Decision Tree Classifier with a depth of 1 can only predict two different classes. For $x \leq 10$ the prediction is $\hat{z} = 21$ (blue area) and for $x > 10$ the prediction is $\hat{z} = 31$ (red area).

In Figure 56 and 57 the learning curves of the global and local LPs and MLPs are depicted. They were created in the same manner as the previous learning curves, but the Decision Tree Classifier feature space partitioning from Figure 55 was used instead of the manual partitioning. Both local MLPs were used with two inputs because both inputs are relevant in both local feature subspaces of the Decision Tree Classifier.

Results of the global and local LPs

In Figure 56 the learning curve of a global LP and the learning curve of a pair of local LPs are depicted. Due to the circumstance that each local LP obtains only the half of the training data, the learning curve has a much higher RMSE values than the learning curve of the global LP in the area where less than 10% of the training data is available. For more than 20% of the training data, the learning curves are almost equal. However, the mean of the RMSE values that correspond to the local LPs is always above the mean of the RMSE values that correspond to the global LP.

The results are worse than the previous results (Figure 53) because the relation of the values in the local feature subspaces is $z = x + min(y, 10)$, which is the same relation that the values have in the global feature space. The LP models are not suitable for that non-affine relation. The global and local LPs have a similar complexity and similar parameters because they try to fit into the same non-affine relation $z = x + min(y, 10)$. But the global model has obtained more training data samples, therefore, the mean of the RMSE of the global model is always smaller.

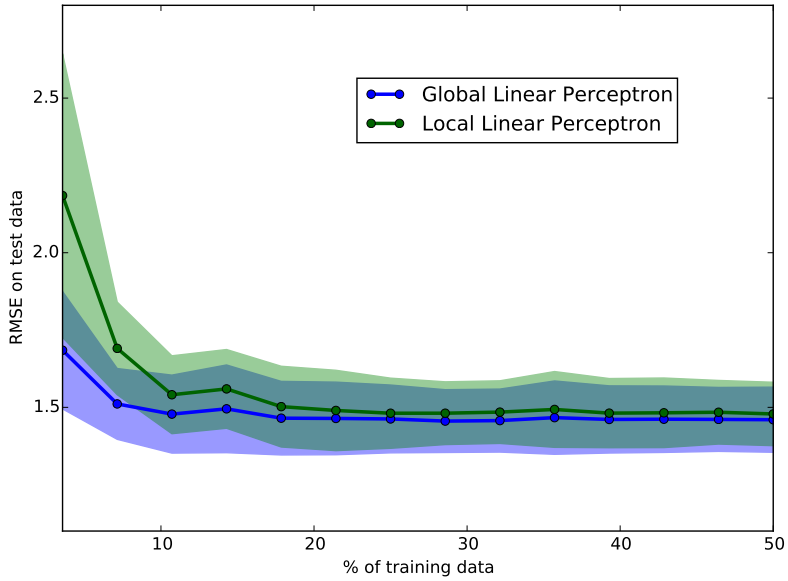

Figure 56: Learning curves of the Linear Perceptron on Dataset 2 using a global model and a pair of local models. Each local model corresponds to one of the both feature subspaces (see Figure 55). The training data is 90% of the entire data. The x-axis depicts the % of training data that was used to achieve the RMSE values on the y-axis. The curve that connects the dots is the mean value of the achieved RMSE values, while the brighter area in a similar color as the mean curve represents the standard derivation.

Results of the global and local MLPs

In Figure 57 the learning curve of a global MLP and the learning curve of a pair of local MLPs are depicted. Due to the circumstance that each local MLP obtains only the half of the training data, the learning curve is expected to have higher RMSE values than the learning curve of the global MLP. The expectation is approved where more than 4% of the training data is available. Only for 0% of the training data the expectation is not approved, however, the behavior of the models is biased by a random initialization in that area. The results are worse than the previous results (Figure 54) because the relation of the values in the local feature subspaces is $z = x + min(y, 10)$, which is the same relation that the values have in the global feature space. Therefore, a simpler local MLP model could not be used due to the

relevance of both inputs in all local feature subspaces.
The global and local MLPs have a similar complexity and similar parameters because they try to fit into the same non-affine relation $z = x + min(y, 10)$. But the global model has obtained more training data samples, therefore, the mean of the RMSE of the global model is always smaller (irrespective of 0% training data).

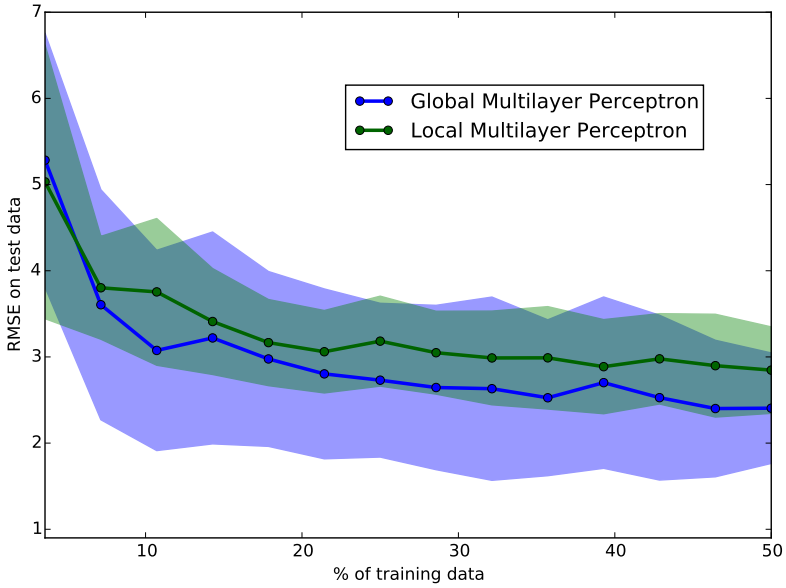

Figure 57: Learning curves of the Multilayer Perceptron on Dataset 2 using a global model and a pair of local models. Each local model corresponds to one of the both feature subspaces (see Figure 55). The training data is 90% of the entire data. The x-axis depicts the % of training data that was used to achieve the RMSE values on the y-axis. The curve that connects the dots is the mean value of the achieved RMSE values, while the brighter area in a similar color as the mean curve represents the standard derivation.

Conclusion
Comparing the results of both partitioning variants, the conclusion is that the generalization capability can be improved by replacing a global model by two local models of different complexity that are responsible for two feature subspaces where the relations are of different complexity, too. Using a Decision Tree Classifier partitioning for that purpose does not allow that kind of improvement. A local model which is reasonable for a synergetic feature subspace should have the same complexity as the global model, while the reasonable model for the remaining feature subspace should be simpler. If that is the case, the pair of local models has collectively more model parameters than the global model. The collective complexity of the local models can be greater than complexity of the global model. However, the results of the Decision Tree Classifier partitioning are demonstrating that not a higher complexity is the key for a improved generalization capability, but a reasonable partitioning of the feature space that allows that a simple local model is suitable for one of both feature subspaces.

4.4 Feature space partitioning

4.4.1 Data partitioning of a simple dataset with boolean variables

In this section a very small dataset is partitioned under different conditions. For every subset the information measures in particular the *Interaction Information* are determined. At first the dataset is partitioned without conditions and every possible data sample subset pair is built. Afterwards the partitioning is performed under the condition that all data samples, which have the same x-y-values have to be in the same partition (feature space partitioning).

Data partitioning of a simple compensation dataset with boolean variables

The assumption that *Synergy* and *Redundancy* are local properties of the feature space leads to the idea of partitioning the data into a *Synergy* partition and a *Redundancy* partition.
In the following experiments the data from the compensation example in Appendix A on page 110 is partitioned under varied conditions.

$z = \overline{x \cdot y}$		
x	y	z
0	0	1
0	0	1
0	1	1
1	0	1
1	1	0
1	1	0

The main motivation of these experiments is the following question:
Can we separate the data from the compensation example into redundant and synergetic data?

Data partitioning without conditions

In this experiment the samples (rows) of the data are partitioned systematically into two partitions. Every possible partitioning that can be created from the data is generated.

Considering N data samples, one bit for every sample which describes whether the sample is assigned to the first or to the second partition allows to define a certain partitioning by N bits. N bits can describe 2^N possible partitionings. If we also consider that every possible partitioning is described by two cases where the partitions are swapped, there are $2^{(N-1)}$ partitionings remaining. Finally, the partitioning where one partition is empty is removed and we have $2^{(N-1)} - 1$ left.

To filter the interesting combinations the *Interaction Information* of both partitions (Q_1 and Q_2) is observed.

Seaching for high *Synergy* and high *Redundancy* at the same time:
Partitioning where $|Q_1 - Q_2|$ reaches its maximum:

Table 5: Data, Entropies and information from six samples of a NAND operation

Data		
x	y	z
0	0	1
0	0	1
0	1	1
1	0	1
1	1	0
1	1	0

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
1.0000 bit	1.0000 bit	0.9183 bit	1.9183 bit	1.4591 bit	1.4591 bit	1.9183 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.0817 bit	0.4591 bit	0.4591 bit	0.9183 bit	0.5409 bit	0.5409 bit	0.0000 bit

Table 6: Data, Entropies and information from three samples of a NAND operation (partition 1)

Parti. 1		
x	y	z
0	0	1
0	0	1
1	1	0

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
0.9183 bit	0.9183 bit	0.9183 bit	0.9183 bit	0.9183 bit	0.9183 bit	0.9183 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.9183 bit	0.9183 bit	0.9183 bit	0.9183 bit	0.9183 bit	0.9183 bit	-0.9183 bit

Table 7: Data, Entropies and information from three samples of a NAND operation (partition 2)

Parti. 2		
x	y	z
1	0	1
1	1	0
0	1	1

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
0.9183 bit	0.9183 bit	0.9183 bit	1.5850 bit	1.5850 bit	1.5850 bit	1.5850 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.2516 bit	0.2516 bit	0.2516 bit	0.9183 bit	0.9183 bit	0.9183 bit	0.4150 bit

For this case a partitioning into *Redundancy* $Q_1 = -0.9183$ bit and *Synergy* $Q_2 = 0.4150$ bit could be achieved (see Tables 5 - 7).
Note: This is also the partitioning where $|Q_1 \cdot Q_2|$ reaches its maximum.

Searching for Synergy pairs or Redundancy pairs:
Partitioning where $Q_1 \cdot Q_2$ reaches its maximum:

Table 8: Data, Entropies and information from three samples of a NAND operation (partition 1)

Parti. 1		
x	y	z
0	0	1
0	1	1
1	1	0

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
0.9183 bit	0.9183 bit	0.9183 bit	1.5850 bit	0.9183 bit	1.5850 bit	1.5850 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.2516 bit	0.9183 bit	0.2516 bit	0.9183 bit	0.2516 bit	0.9183 bit	-0.2516 bit

Table 9: Data, Entropies and information from three samples of a NAND operation (partition 2)

Parti. 2		
x	y	z
1	0	1
0	0	1
1	1	0

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
0.9183 bit	0.9183 bit	0.9183 bit	1.5850 bit	1.5850 bit	0.9183 bit	1.5850 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.2516 bit	0.2516 bit	0.9183 bit	0.9183 bit	0.9183 bit	0.2516 bit	-0.2516 bit

For this case a partitioning into a *Redundancy* pair $Q_1 = -0.2516$ bit and $Q_2 = -0.2516$ bit could be achieved (see Tables 8, 9).

There are also other interesting partitionings which are leading to the following results:

Table 10: Results of further partitionings

Interaction Information		
Q_1	Q_2	sign combination
-0.9183 bit	0.4150 bit	[- , +]
-0.2516 bit	-0.2516 bit	[- , -]
0.0 bit	0.0 bit	[0 , 0]
0.0 bit	0.3774 bit	[0 , +]
0.0 bit	-1.0 bit	[0 , -]
$\emptyset$	$\emptyset$	[+ , +]

If we look at Table 10, we can see that almost every sign combination is possible. The only missing combination is a partitioning where we get a pair of *Synergy* partitions.

This experiment shows that we can manipulate the *Interaction Information* of data partitions by choosing the partition assignment of the data samples.

The freedom of the sample assignment has to be constrained if we are interested in partitioning the feature space into feature subspaces. All data samples which belong to the same input values or the same location in the feature space have to be assigned to the same data partition. Therefore, it is not allowed to have the sample (0,0,1) and (1,1,0) in both data partitions as it was the case in Table 8 and 9.
If x and y are discrete input values and z is a discrete output value of a system, the feature space consists of all possible value combinations that x and y are able to describe.

The following experiment performs the partitioning in the feature space.

Data partitioning with the condition that identical samples are in the same partition

In this experiment the samples (rows) of the data are partitioned systematically into two partitions. Every possible feature space partitioning was generated.

Considering M groups of samples, where every sample of a group has to be assigned to the same partition, $2^{(M-1)}-1$ partitionings are possible.
To filter the interesting combinations the *Interaction Information* of both partitions (Q_1 and Q_2) is observed.

Seaching for high Synergy and high *Redundancy* at the same time:
Partitioning where $|Q_1 - Q_2|$ reaches its maximum:

Table 11: Data, Entropies and information from two samples of a NAND operation (partition 1)

Parti. 1		
x	y	z
0	1	1
1	0	1

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
1.0000 bit	1.0000 bit	0.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
1.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	1.0000 bit	1.0000 bit	0.0000 bit

Table 12: Data, Entropies and information from four samples of a NAND operation (partition 2)

Parti. 2		
x	y	z
0	0	1
0	0	1
1	1	0
1	1	0

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
1.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
1.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit	1.0000 bit	-1.0000 bit

Seaching for the highest synergy:
Partitioning where $max\{Q_1, Q_2\}$ reaches its maximum:

Table 13: Data, Entropies and information from four samples of a NAND operation (partition 1)

Parti. 1		
x	y	z
0	1	1
1	0	1
1	1	0
1	1	0

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
0.8113 bit	0.8113 bit	1.0000 bit	1.5000 bit	1.5000 bit	1.5000 bit	1.5000 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.1226 bit	0.3113 bit	0.3113 bit	1.0000 bit	0.8113 bit	0.8113 bit	0.3774 bit

Table 14: Data, Entropies and information from two samples of a NAND operation (partition 2)

Parti. 2		
x	y	z
0	0	1
0	0	1

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit

Searching for *Synergy* pairs or *Redundancy* pairs:
Partitioning where $Q_1 \cdot Q_2$ reaches its maximum:
Note: In all possible partitionings at least one partition has an Interaction Information value of zero. Hence, there are no Synergy or Redundancy pairs.

Table 15: Data, Entropies and information from two samples of a NAND operation (partition 1)

Parti. 1		
x	y	z
1	1	0
1	1	0

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.0000 bit

Table 16: Data, Entropies and information from four samples of a NAND operation (partition 2)

Parti. 2		
x	y	z
0	0	1
0	0	1
0	1	1
1	0	1

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
0.8113 bit	0.8113 bit	0.0000 bit	1.5000 bit	0.8113 bit	0.8113 bit	1.5000 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.1226 bit	0.0000 bit	0.0000 bit	0.0000 bit	0.1226 bit	0.1226 bit	0.0000 bit

Table 17: Summary of sign combinations

Interaction Information		
Q_1	Q_2	sign combination
∅	∅	[- , +]
∅	∅	[- , -]
0.0 bit	0.0 bit	[0 , 0]
0.0 bit	0.3774 bit	[0 , +]
0.0 bit	-1.0 bit	[0 , -]
∅	∅	[+ , +]

If we look at Table 17, we can see that all possible partitionings contain at least one partition which has an *Interaction Information* value of zero.

This experiment as well as the previous experiment shows that we can manipulate the *Interaction Information* of data partitions by choosing the partition assignment of the sample groups in the data.

Because of the conditions, there are less sign combinations. The open-ended question is whether or not more sign combinations would be possible while keeping the conditions if the original data consisted of more samples.
The most interesting result of this experiment is that we can get a partitioning which results in two partitions with the *Interaction Information* $Q_1 = 0$bit and $Q_2 = -1$bit (see Table 11 and 12).

Both partitions describe extremely trivial variable relations:
The behavior that the first partition (Table 11) describes is:
"The output is always 1".
Because Z is independent from X, Y and (X, Y), we have a *Non-Interaction* case here.

And the behavior that the second partition (Table 12) describes is:
"The output is always $\overline{x}$." or likewise **"The output is always $\overline{y}$."**
Because the value z can be determined from the value x or the value y, we have a *Redundancy* case here.
Therefore, the *Synergy* was removed from the data by partitioning the feature space.
We can not conclude from this experiment that we can also remove the *Synergy* from more complex data using a feature space partitioning, but it demonstrates that there is a potential for a variable relation simplification by partitioning the feature space.

We also have to take into account that systems which have only two discrete output values $z_1 = 0$ and $z_2 = 1$ have a maximum entropy $H(Z) = 1bit$. By the assignment of every possible input value combination to partition 1 or 2 we also respectively provide 1bit of information per possible input value combination.
That is the reason why a partitioning itself can completely model the relation of the boolean variables in the data. If all samples with the output value 0 are assigned to partition 1 and all samples with the output value 1 are assigned to partition 2, we have two partitions with respectively a constant output value z, therefore, Z is independent from X, Y and (X, Y) and both partitions can be dedicated to the *Non-Interaction* case, where neither *Synergy* nor or *Redundancy* exists (see for example Table 15 and 16).

While this works perfectly for an output with only two possible output values (e.g. $z_1 = 0$, $z_2 = 1$), we would usually not aim to get this kind of *Non-Interaction* partition because we can not get additional information from the x and y values about the z value. Hence, if the output value can take more than two values, we will need a dependency between the input and output variables to gather more than 1bit of information about the output value z.
This can also be explained by the equation (41) on page 18. If the W is w_1 (partition 1) for samples where $z = 0$ and w_2 (partition 2) for samples where $z = 1$, the remaining entropy

$$H(Z|W) = H(W, Z) - H(W) = 0$$

is zero. Hence, the remaining uncertainty about Z is 0 after the assignment.
Due to the circumstance that the assignment of the samples to two different partitions can reduce the entropy of Z by 1bit, remaining interdependencies between X, Y and Z become more important if variables with higher entropies $(H(X), H(Y), H(Z) > H(W))$ are used, which are only slightly reduced by the assignment W. That is the reason why the more complicated datasets from Section 4.1 are used for further experiments.

4.4.2 Partitioning of the feature space using the developed methods

In this experiment the feature space partitioning methods are tested on the datasets that are described in Section 4.1. They are tested in their capability of finding a feature subspace with a maximum *Interaction Information* (maximum *Synergy*) and feature subspace with a minimum *Interaction Information* (maximum *Redundancy*). As a baseline for the partitioning of the dataset the manual partitioning from Section 4.1 is used.

The parameters of the methods were adjusted by trial and error in that way that they are best in maximizing the *Interaction Information* in subspace 1 of Dataset 1. That decision were made for the following reasons:
1) Dataset 1 contains four cases (Synergy, Redundancy, Non-Interaction and Compensation) and the methods should be able to work well even if all of these cases exist in the data.
2) Maximizing the *Synergy* is more challenging in Dataset 1 than maximizing the *Redundancy*. The maximum pure *Redundancy* requires only 4 samples that are in different rows and columns of the feature space (e.g. diagonal distributed), while the maximum *Synergy* requires a much specific selection of samples.
3) By using the same parameters for all datasets, we can test whether a method that was adjusted for one dataset will also work well on another dataset.

The meaning of the parameters that are noted in the upper part of the pictures is summarized in the following list:

Parameter overview of the feature space partitioning methods (see Section 3.2 for more details)

Method 1: Genetic Algorithm
μ: Number of parents, which is equal to the size of the initial population and the number of offspring $\lambda = \mu$.
$e = 1$: Number of elitists
$T = 300$: Number of iterations
$S = 1.4$: Selection pressure
$n_{point} = 8$: Number of random crossover points while the recombination
$p_m = 0.01$: Probability for every assignment bit in the offspring that it is inverted

Method 2: Gradient Descent
$\gamma = 0.1$: Step size in an optimization step
$sgn : yes$: If yes then the gradient is used in a signum function.
$\Delta w = 0.01$: The change of every relaxed sector assignment for the approximation of the gradient using the difference quotient
$c = 2$: Is a parameter of the penalty function that measures the constraint violations of the relaxed sector assignments. $c > 1$ allows a stronger weighted penalty for greater constraint violations.
$c_r = 8$: The higher c_r the later the penalty function f_p takes a significant effect in the optimization process.
$T = 11$: Number of iterations

Method 3: Map Reinforcement
$\gamma = 10$: Step size in an optimization step
$w_{norm,min} = 0.01$: Smallest value that occurs in $\hat{\boldsymbol{w}}$ after the normalization of $\hat{\boldsymbol{w}}$.
$n_{thr} = 10$: Number of w-thresholds that are tested, before the values of $\hat{\boldsymbol{w}}$ are quantized to 0 or 1.
$T = 10$: Number of iterations

All results of Q_1 are summarized in Table 18. A comparison of visualized sector assignments are shown in the following pictures, the complete collection of the separate pictures can be found in the Appendix F. For every partitioning method one exemplary Figure that depicts the optimization process using curve progressions is shown.

Results

In Figure 58 the partitioning results (sector assignments to feature subspace 1) of Dataset 1 are summarized. The Figure depicts six partitioning results, where every of the three rows corresponds to one partitioning method. Method 1 corresponds to row 1, Method 2 to row 2 and Method 3 to row 3. The first column (on the left side) shows the results of the *Interaction Information* maximization in feature subspace 1, while the second column (on the right side) shows the results of the *Interaction Information* minimization in feature subspace 1. The feature subspace 1 consists of the sectors that are colored in magenta. Hence, the areas in magenta on the left side are dominated by *Synergy* and the areas in magenta on the right side are dominated by *Redundancy*. In the Appendix F the Figures from the sector assignment comparison are collected separately and contain additional information.
Equivalent summaries of the sector assignments were created for Dataset 2 (see Figure 59) and Dataset 3 (see Figure 60).
Note: Dataset 2 has much more data sample positions in the feature space than Dataset 1 and 3. In order to reduce the computation time, the sector size 2x2 were chosen.

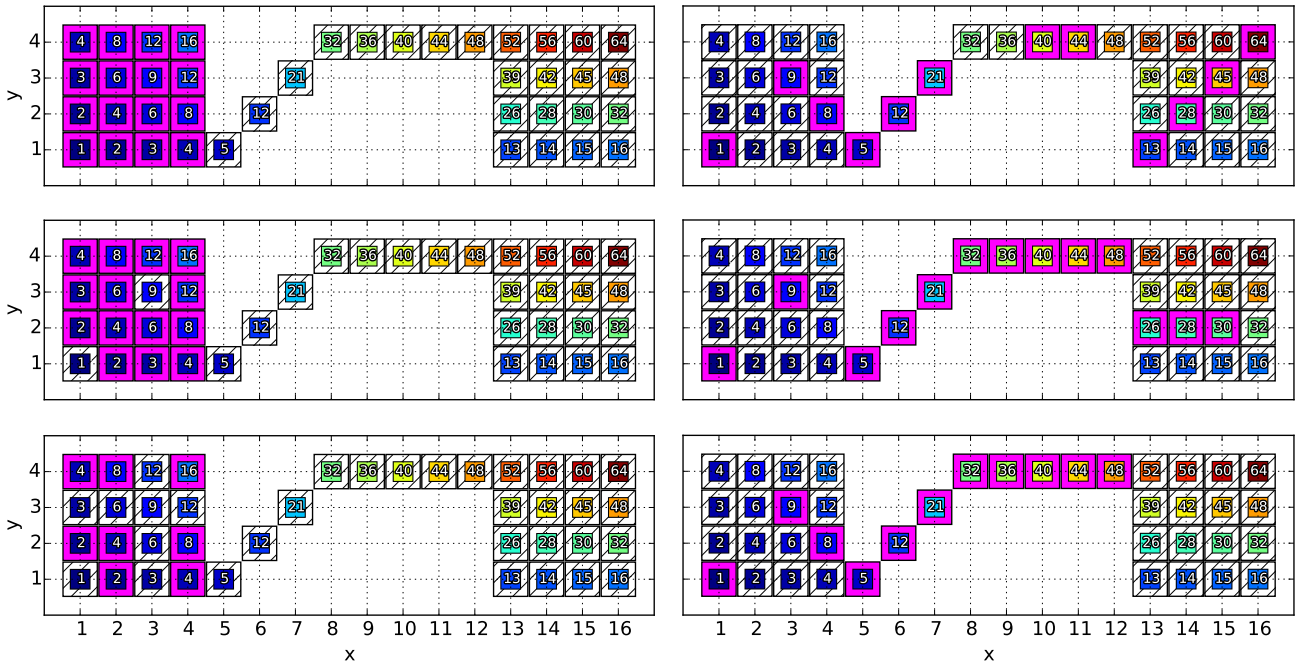

Figure 58: Comparison of the sector assignments for Dataset 1, where the feature spaces on the left side show the results of the *Interaction Information* maximization in feature subspace 1 and the feature spaces on the right side correspond to the *Interaction Information* minimization in feature subspace 1. The sectors that were assigned to feature subspace 1 are colored magenta. The results of Method 1 are shown in the first row and the results of Method 2 are placed in the second row, while the result in the third row corresponds to Method 3.

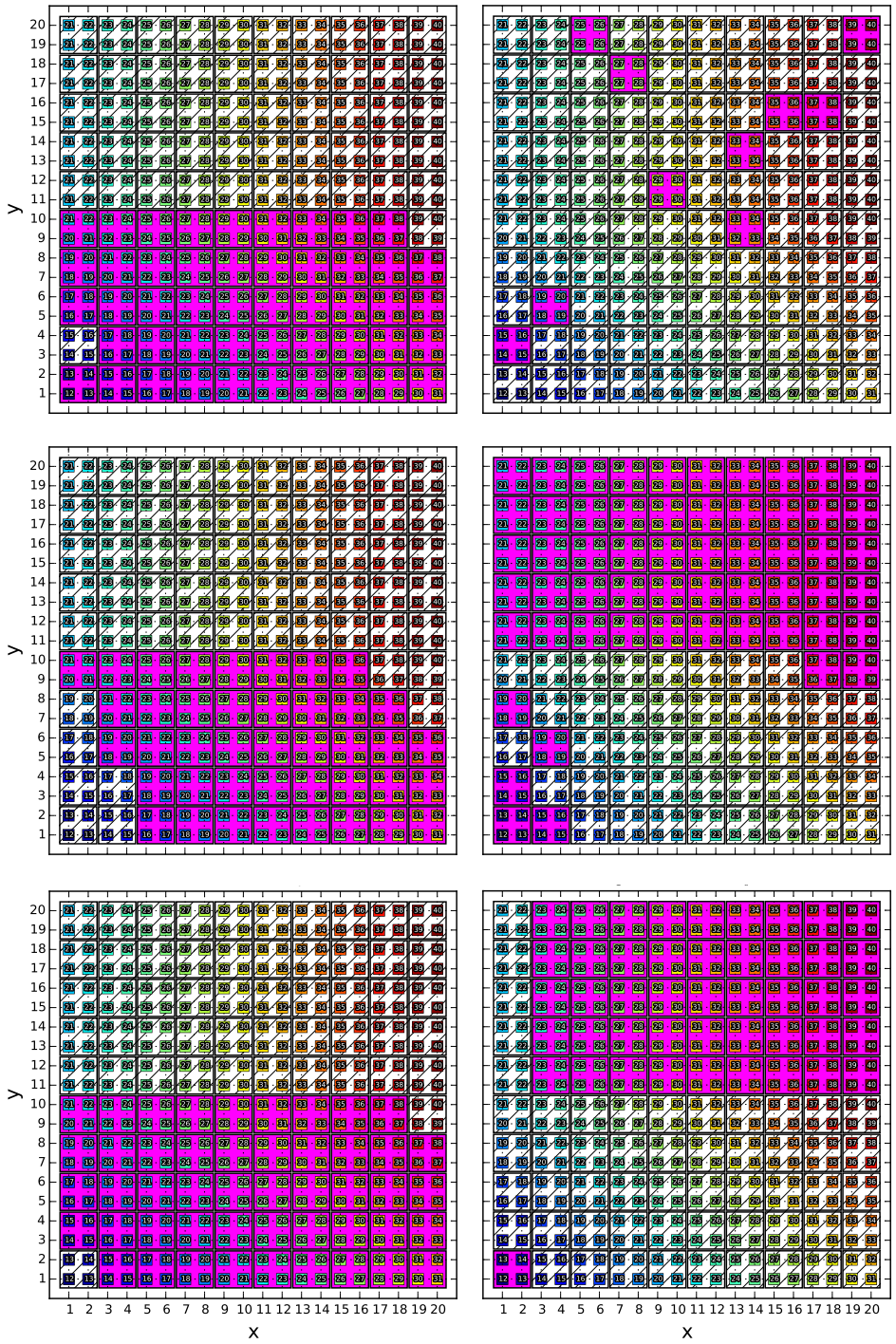

Figure 59: Comparison of the sector assignments for Dataset 2, where the feature spaces on the left side show the results of the *Interaction Information* maximization in feature subspace 1 and the feature spaces on the right side correspond to the *Interaction Information* minimization in feature subspace 1. The sectors that were assigned to feature subspace 1 are colored magenta. The results of Method 1 are shown in the first row and the results of Method 2 are placed in the second row, while the result in the third row corresponds to Method 3.

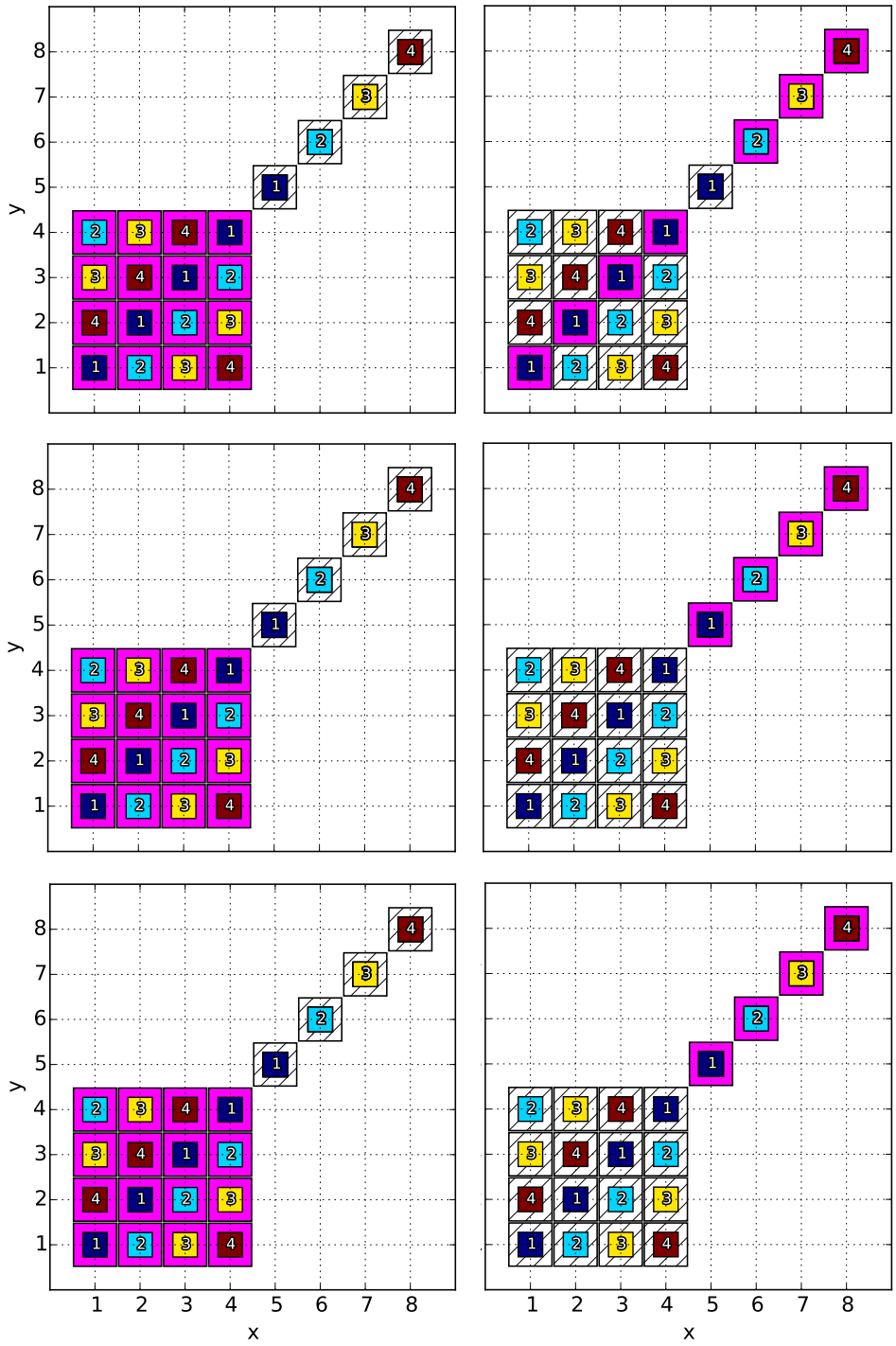

Figure 60: Comparison of the sector assignments for Dataset 3, where the feature spaces on the left side show the results of the *Interaction Information* maximization in feature subspace 1 and the feature spaces on the right side correspond to the *Interaction Information* minimization in feature subspace 1. The sectors that were assigned to feature subspace 1 are colored magenta. The results of Method 1 are shown in the first row and the results of Method 2 are placed in the second row, while the result in the third row corresponds to Method 3.

The following curves are showing the maximization process of $Q_1(t)$ for every method on Dataset 1.
In Figure 61 the curve progression of the maximum $Q_1(t)$ that corresponds to the best individual in the population of iteration t using Method 1 is depicted. Because of the elitism ($e = 1$), the curve stays constant or grows. The maximum value is already achieved at $t \approx 150$. However, the results of Method 1 vary from run to run and a higher number of iterations can not degrade the result. Therefore, $T = 300$ was chosen as maximum number of iterations.

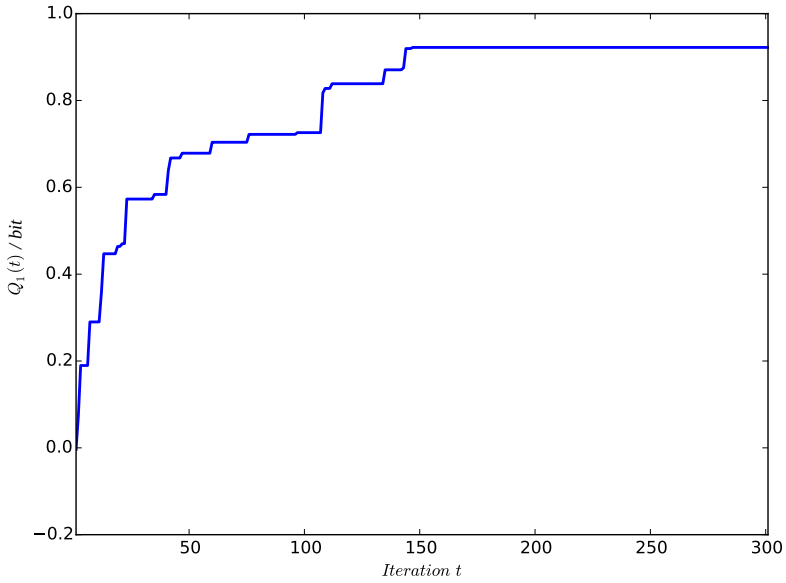

Figure 61: Curve of the optimization process of Dataset 1 using Method 1 to maximize Q_1, which shows the highest value of Q_1 for every iteration (or generation).

In Figure 62 four curves of the optimization process using Method 2 in dependency of the current number of iterations t are depicted. The black curve shows the value of $Q_1(t)$ in every iteration t for relaxed sector assignments. The value $Q_1(t)$ that corresponds to the relaxed sector assignments exceeds the value $1bit$ at the end, while the final value after the assignment quantization is smaller ($Q_1 = 0.8908bit$). The blue curve shows the penalty function $f_p(t)$ for every iteration, which is a measure for constraint violation. At the beginning the constraint of a definite sector assignments is violated at most because all sector assignments are initialized with 0.5, which means that every sector corresponds equally to feature subspace 1 and feature subspace 0. While the optimization the penalty function $f_p(t)$ falls continuously and is finally smaller than 0.1, while $Q_1(t)$ keeps growing in spite of taking the constraints into account more and more. The green curve shows the negated cost function $Q_1(t) \cdot (1 - r(t) \cdot f_p(t))$ which is the actual value that is maximized by the method, while considering $Q_1(t)$ and the constraint violation at the same time. However, the constraint violations have no significant influence in the area $t < 8$ due to the parameter $c_r = 8$, which causes a late increase of $r(t)$ that weights the influence of penalty function $f_p(t)$ in the cost function.
The red curve shows for every iteration the factor $(1 - r(t) \cdot f_p(t))$ which has to be multiplied with Q_1 to create the negated cost function. For $t > 8$ the factor starts to fall significantly below 1, which decreases the negated cost function, while $Q_1(t)$ keeps growing slowly.

In Figure 63 the curve of $Q_1(t)$ during the optimization process using Method 3 in dependency of the current number of iterations t is depicted. The growing value $Q_1(t)$ which is calculated for relaxed sector assignments exceeds the value $1bit$, while the final value after the assignment quantization is smaller ($Q_1 = 0.9718bit$).

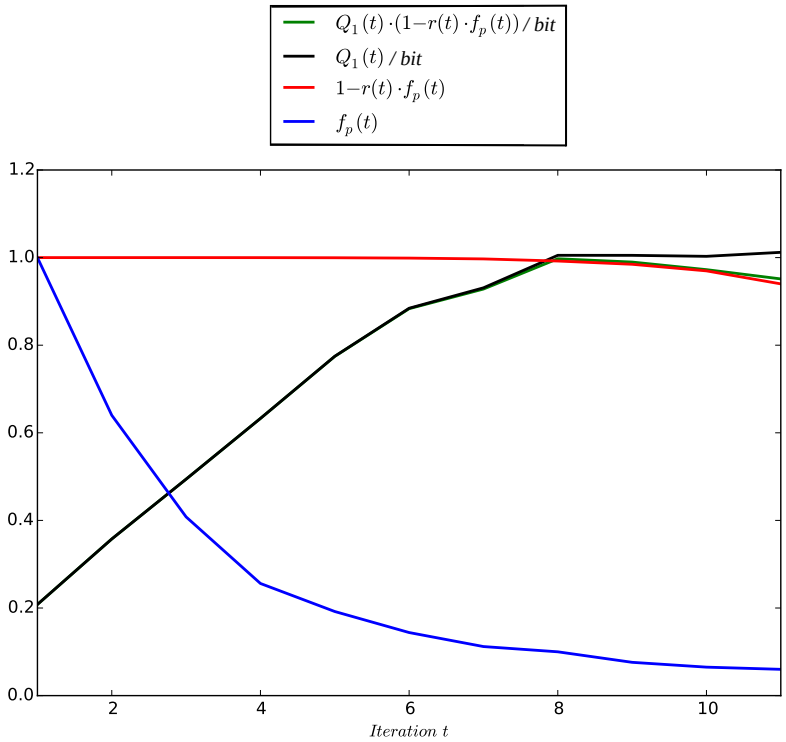

Figure 62: Curves of the optimization process of Dataset 1 using Method 2 to maximize Q_1, where the black curve shows the value of Q_1 in every iteration. The blue curve shows the penalty function for every iteration, which is a measure for constraint violation. The green curve shows the negated cost function that is maximized by the method, while considering Q_1 and the constraint violation at the same time. The red curve shows for every iteration the factor which has to be multiplied with Q_1 to create the negated cost function.
Note: The value Q_1 which is calculated for relaxed sector assignments exceeds the value 1bit, while the final value after the assignment quantization is smaller ($Q_1 = 0.8908bit$).

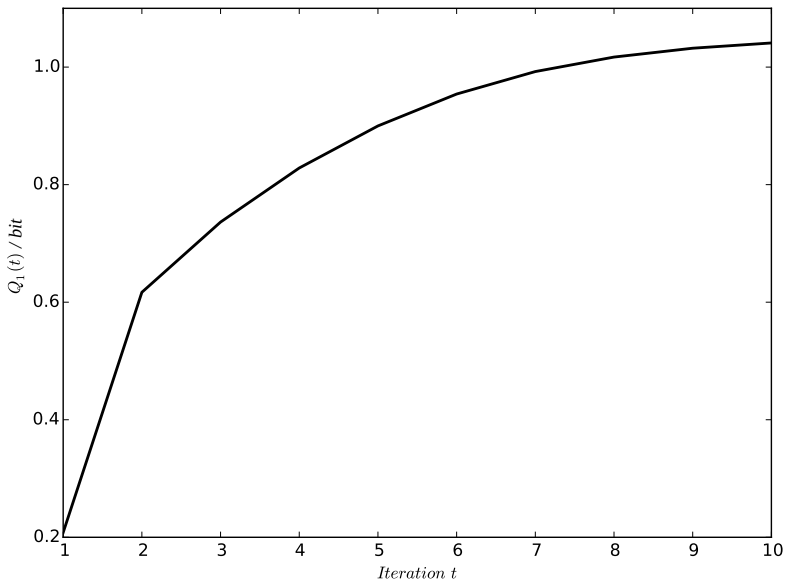

Figure 63: Curve of the optimization process of Dataset 1 using Method 3 to maximize Q_1, which shows the value of Q_1 in every iteration.
Note: The value Q_1 which is calculated for relaxed sector assignments exceeds the value 1bit, while the final value after the assignment quantization is smaller ($Q_1 = 0.9718bit$).

Summary of the results

The results of Q_1 of all dataset, method and goal combinations are summarized in Table 18. The values of Q_1 from the manual partitioning in Section 4.1 were added as a baseline.

Table 18: Summary of the feature space partitioning results

Results					
		Method 1	Method 2	Method 3	Manual
Dataset	Goal	Q_1	Q_1	Q_1	Q_1
1	maximize Q_1	0.9222 bit	0.8908 bit	0.9718 bit	0.9222 bit
2	maximize Q_1	2.9503 bit	2.8995 bit	2.9900 bit	2.9676 bit
3	maximize Q_1	2.0000 bit	2.0000 bit	2.0000 bit	2.0000 bit
1	minimize Q_1	-2.0000 bit	-1.8843 bit	-1.8586 bit	-2.0000 bit
2	minimize Q_1	-2.5408 bit	-0.1040 bit	-0.1402 bit	0.0000 bit
3	minimize Q_1	-2.0000 bit	-2.0000 bit	-2.0000 bit	-2.0000 bit

Discussion of the results

As shown in Table 18 all tested methods produce similar results at maximizing Q_1 on Dataset 1. Method 3 provides an even higher value than the manual partitioning. In Figure 58 on the left side the results for Method 1 (top), Method 2 (middle) and Method 3 (bottom) are shown. The feature subspace of Method 3 with the best result consists of four unconnected parts. Using a larger sector size e.g. 2x2 would produce a subspace of connected sectors, which is the result of Method 1. On larger datasets, where larger sectors sizes are recommended, the isolated assignment of single data samples would not longer occur. Because Method 1 is a genetic algorithm, the new results are produced by a recombination of previous results. All good results contain samples that are in the area $1 \leq x \leq 4$ there are not much results possible that allow a higher value than the value Q_1 of Method 1. Hence, the most results in the population are similar, with slight differences. However, removing single sectors as shown in Method 2 leads to a smaller value of Q_1, although the same sectors ($x = y = 1$, $x = y = 3$) that are removed in Method 2 are also removed in Method 3. For the result of Method 3 the half of 16 sectors has to be removed in order to reach a very unique and better result than Method 1.
In contrast to Method 1, Method 2 and 3 are converging very fast, therefore, they converge to a local optimum. Early optimization steps strongly influence the final result. For example, Method 2 was not able to pick up the two sectors at $x = y = 1$ and $x = y = 3$ because they are not useful at the beginning. In the *Interaction Information Map* (see Figure 45 on page 76) those two sectors at $x = y = 1$ and $x = y = 3$ are negative shares of the *Interaction Information*. If only the subspace $1 \leq x \leq 4$ is considered then those shares are zero, but taking them into account increases the remaining shares, therefore, the result of Method 1 is better than the result of Method 2, although the result of Method 1 is comprehensible. The result of Method 3 is that different because it does not take the constraint of a definite sector assignment into account (no penalty function) and at the end this disadvantage is reduced by testing several thresholds for the assignment and the best threshold is used. A high threshold is the reason for the small number of sectors that are colored in magenta.
If we take a short look into the appendix (Figures 70, 72, 74) we can see that the *Redundancy* of the remaining sectors ($Q_0 = -0.4450bit$) for Method 1 is the highest. Of course Q_0 was not considered while the optimization, however, the result of Method 1 is the best result from the three methods for a feature space partitioning with two local models of different complexity.

At minimizing Q_1 on Dataset 1 the results of all methods are similar, too (see Table 18). But Method 1 is the only one that achieves the same value as the manual partitioning. In Figure 58 on the right side the

results for Method 1 (top), Method 2 (middle) and Method 3 (bottom) are shown. Non of the methods found the simple result of the manual partitioning, which is simply the area ($5 \leq x \leq 8$) that consists of only four sectors. However, Method 1 found a result with the same value of $Q_1 = -2bit$. The results of all methods consist of diagonal and horizontal sector chains. In the area ($1 \leq x \leq 7$) the partitioning of Method 1 and 3 is identical, while Method 2 has missed the sector at position ($x = 4$, $y = 2$). In the area ($5 \leq x \leq 12$) the result of Method 2 and 3 is equal, while Method 1 did not take the sectors at $x = 8$, $x = 9$ and $x = 12$. In the area ($13 \leq x \leq 16$) the results of all methods are different. Method 1 has a diagonal chain and Method 2 a horizontal chain of sectors, while Method 3 did not take sectors from that area.
If the goal is to minimize Q_1, we are interested in diagonal chains rather than in horizontal chains. A pure *Redundancy* requires that all three variables contain the same information, which is only possible with diagonal chains. A horizontal chain e.g. in area ($8 \leq x \leq 12$) is rather a *Non-Interaction* because two variables contain the same information (Z and X) and Y contains no information about X and Z. Therefore, it is obvious that Method 1 achieved the best result of the *Redundancy* search because it has the largest number of diagonal sector chains.
If we observe the *Interaction Information Map* again (Figure 45 on page 76), we can see that the shares of the *Interaction Information* are negative in the horizontal chain ($9 \leq x \leq 12$) if the whole feature space is taken into account. Method 2 and 3 have reinforced that area at the beginning of the optimization and have kept it until the end.

At maximizing Q_1 on Dataset 2 the results of all methods are similar (see Table 18). But Method 3 is again the only method that achieves a higher value of Q_1 than the manual partitioning. In Figure 59 on the left side the results for Method 1 (top), Method 2 (middle) and Method 3 (bottom) are shown. As we can see the sector size is 2x2 for Dataset 2. The sector assignment of Method 3 is very similar to the sector assignment of the manual partitioning, where the feature subspace is simply the area ($1 \leq y \leq 10$). But Method 3 removed two more sectors (at $x = y = 1$ and $x = y = 10$) to achieve an even better result. If we remember the *Interaction Information Map* see Figure 46 on page 76, we can see that already the unpartitioned feature space has a diagonal shape of the most synergetic positions labeled with "++". This shape is even more noticeable in the result of Method 2, where Q_1 is smaller than in the simple manual partitioning. Although Method 3 is using the *Interaction Information Map* while the optimization, the implemented threshold tests find a reasonable threshold and avoids a more diagonal shape by making the decision to keep even those sectors that are less synergetic using a smaller threshold. Method 1 was able to find almost the same result as Method 3, but the removed sector is at ($x = 3$, $y = 1$) instead of ($x = 1$, $y = 1$). Because of the random components (e. g. mutation) in Method 1 the result of Method 3 could also be a result of Method 1. Due to the *Non-Interaction* in the area $11 \leq y \leq 20$, no method has used sectors from that area to create a synergetic subspace.

At minimizing Q_1 on Dataset 2 the results of all methods exceptionally not similar (see Table 18). Method 2 and 3 found results near zero because Dataset 2 was intended to have a *Synergy* part and *Non-Interaction* part, but no *Redundancy* part as shown in the manual partitioning in Section 4.1. In contrast, Method 1 found a features subspace with an outstanding small value of Q_1. In Figure 59 on the right side the results for Method 1 (top), Method 2 (middle) and Method 3 (bottom) are shown. The magenta sectors of Method 1 are distributed across the entire feature space and do not form a connected area. This is a very good result with respect to the goal. Dataset 2 was not intended to contain a *Redundancy* and the baseline value was zero. However, there is no particular large area where the variable relations are redundant. Method 1 creates the *Redundancy* itself by picking particular sectors from the whole feature space. The sector distribution has a slightly diagonal shape, which allows to create the *Redundancy*. This kind of result leads to the smallest *Synergy* in the remaining feature subspace 0 ($Q_0 = 2.0042bit$) comparing the three methods (see Figures 77, 79 and 81 in the appendix). Therefore, it is not the best result in order to use two local models of different complexity, a larger sector size or an

alternative goal (optimization criterion), would avoid that behavior.
The partitioning results of Method 2 and 3 are almost the opposite of the result from the Q_1 maximization. If we look at *Interaction Information Map* (see Figure 46 on page 76), we can observe that there is no negative *Interaction Information* share on the map. Hence, the search for a *Redundancy* was started in the areas, where the shares are the smallest, which includes the *Non-Interaction* area. In contrast to Method 1, Method 2 and 3 are treating sectors equally in the current optimization step if they have currently the same influence on Q_1. Consequently, they would not remove particular sectors from an area with homogenous *Interaction Information* shares.

On Dataset 3 the Q_1 results of all methods are exactly the same as the Q_1 results of the manual partitioning (see Table 18). In Figure 60 six partitioning results are shown. The results of the Q_1 maximization goal are on the left side of the figure and the results of the Q_1 minimization goal are on the right side of the figure. The results for Method 1 are on top, the results of Method 2 in the middle and the results of Method 3 at the bottom of the figure. Almost all methods are partitioning the feature space in a pure *Synergy* subspace ($1 \leq x \leq 4, 1 \leq y \leq 4$) and a pure *Redundancy* subspace ($5 \leq x = y \leq 8$) in the same manner as the manual partitioning from Section 4.1. The only exception can be observed in the result of Method 1 at minimizing Q_1, where an alternative result, with the same value $Q_1 = -2bit$ as in the other methods, is shown. The sectors are forming a diagonal chain in the *Synergy* subspace. In Dataset 3 the samples in the *Redundancy* subspace ($5 \leq x = y \leq 8$) have a frequency of 4, while the samples in the *Synergy* subspace ($1 \leq x \leq 4, 1 \leq y \leq 4$) have a frequency of 1 (see also Figure 36 on page 68). Thus, the four samples at position $x = y = 5$ and $z = 1$ can be replaced by four samples at the positions $x = y = 1, ..., 4$ and $z = 1$. The entropy $H(Z) = 2bit$ does not change because all z values on the new diagonal chain are 1. The entropy limits the *Interaction Information* $|Q_1| \leq H(Z)$. However, the result of Method 1 in feature subspace 0 ($Q_0 = 0.3774bit$) is much smaller than the result of the other methods ($Q_0 = 2bit$) (see Figures 83, 85 and 87 in the appendix). Therefore, the alternative result is less reasonable for using local models of different complexity. It is probable that the result of the other methods was also contained in the final population of Method 1, however, an arbitrary result from the set of best results in the population is chosen as final result.
For Method 1 and 2 the results are rather comprehensible due to the special properties of Dataset 3. In the *Interaction Information Map* (see Figure 47 on page 77) we can see that at the beginning the *Interaction Information* shares are positive and equal in the *Synergy* subspace ($1 \leq x \leq 4, 1 \leq y \leq 4$), while the shares are negative and equal in the *Redundancy* subspace ($5 \leq x = y \leq 8$). Hence, the relaxed sector assignments would be pushed in the correct direction in the first optimization step. The final result can already be found using only one iteration. If more than one iteration is used the assignment becomes clearer, but the optimization steps of the weights will not change their direction. For example, if the subspace with the maximum *Synergy* is the goal, the assignment weights that correspond to the sectors of the *Synergy* subspace ($1 \leq x \leq 4, 1 \leq y \leq 4$) are increased towards 1 and the assignment weights that correspond to the sectors in the *Redundancy* subspace ($5 \leq x = y \leq 8$) are decreased towards 0.

Summary of the discussion
All methods were able to achieve good results which were in the most cases similar to the manual partitioning results that were used as a baseline. Method 3 was always slightly better than Method 2 with respect to the goal, while the partitioning of both methods looks similar. Method 1 was in all cases the best at maximizing Q_1, while Method 1 was always the best in minimizing Q_1. The best results at maximizing or minimizing Q_1 do not necessarily lead to the best partitioning for using two local models of different complexity. In order to find a result that is most suitable, additional measures can be built into the optimization criterion (see Section 3.2 for more details). If a fast method is required Method 3 is the best choice because it is the fastest of the methods. If the partitioning is done only once and much time is available, Method 1 with a high number of iterations can be used. If the result of Method 1 is finally available, it might be a good idea to compare the result with the result from Method 3 at the end.

5 Conclusion

By utilizing and extending the concept of *Interaction Information*, methods were developed that are able to partition the feature space by evaluating the *Interaction Information* in two-dimensional feature subspaces. The partitioning aims to separate one feature subspace, where a *Synergy* between the variables is dominating, from the remaining feature space, where usually a Redundancy between the variables is dominating.
Furthermore, *Information Maps* were introduced, which can be used as a visualization technique that decomposes information measures in local shares. The visualization can be used to understand the local and global dependencies and interactions of variables.
It was shown that the global relevance of a feature in the entire feature space differs from the relevance of the feature in certain feature subspaces. Hence, the relevance of a feature can be different at different locations in the feature space. Ignoring that circumstance, global feature selection methods evaluate the average relevance of features across the whole feature space. The partitioning allows to take advantage of the better generalization capabilities in areas where less features are relevant, while all features can be considered in areas where more features, especially for synergetic effects, are needed. If synergetic areas in the feature space are small compared to areas that are not synergetic, the overall *Synergy* could be removed by global feature selection techniques. That can be avoided by separating synergetic areas from the rest of the feature space. An overall improved prediction can be achieved depending on the *Interaction Information* in the feature subspaces. As soon as a partitioning of the feature space was determined, the local feature selection can either be applied by the prediction model if it is capable to do that (e.g. linear perceptron) or the features can be removed using the rules based on information measures (see Section 3.2).

6 Outlook

Because the quantification of *Redundancy* and *Synergy* using the *Interaction Information* is restricted to three variables, the partitioning as described in Section 3.2 is not designed for datasets of higher dimensionality. In order to use the methods for higher dimensions of data, the partitioning could be performed for every possible pair of features and the output variable to produce a partitioning for each feature pair. The partitioning results could be evaluated by their feature relevance contrast between the first and the second feature subspace. The pair with the most promising partitioning can be used to partition the high dimensional feature space into two subspaces. However, this approach would at most remove one feature in a feature subspace, while there could be more than one locally irrelevant feature in the high dimensional data.
Another approach could be to decompose the feature space with $N_f > 2$ dimensions in high dimensional sectors and find a subspace where the sum of all pairwise *Interaction Information* is small $\sum_{i<j} Q(X_i;X_j;Y),\ i,j \in \{1,...,N_f\}$, and another feature subspace where the sum is large. In the subspace where the sum is small a feature selection is expected to be reasonable if no interaction of higher order exists.
There are also relatively unusual approaches to decompose the Multivariate Information in nonnegative measures which are based on a new definition of *Redundancy* [24]. If *Redundancy* and *Synergy* can be measured locally for higher dimensions, the same methods that were used for the partitioning using the *Interaction Information* could be used for an alternative measure that handles higher dimensions in a similar way. However, the number of sector assignment combinations increases exponentially with the number of dimensions in the feature space.
Furthermore, the criterion of the partitioning could be used as a criterion for an imbalanced decision tree, that separates local *Redundancy* and *Synergy* at the branches.

A Interaction Information: Examples

Interaction Information

In this section the *Interaction Information* the manual calculation is illustrated for four small datasets, which are examples for *Synergy*, *Redundancy*, *Non-Interaction* and *Compensation*.
For the calculation of $Q(X;Y;Z) = H(X,Y) + H(X,Z) + H(Y,Z) - H(X) - H(Y) - H(Z) - H(X,Y,Z)$ several entropies have to be calculated. Therefore probability distributions of the variables and the joint variables are needed. Thus, they will be approximated by using the given data because they are usually unknown.
Assuming the given data consists of N samples and every sample is an observed combination of the discrete values that the three random variables X, Y, Z can take (x_i, y_j, z_k), the probability distributions can be estimated as follows:

$$p(x_i) \approx \frac{1}{N} f_o(x_i) \tag{222}$$
$$p(y_j) \approx \frac{1}{N} f_o(y_j) \tag{223}$$
$$p(z_k) \approx \frac{1}{N} f_o(z_k) \tag{224}$$
$$p(x_i, y_j) \approx \frac{1}{N} f_o(x_i, y_j) \tag{225}$$
$$p(x_i, z_k) \approx \frac{1}{N} f_o(x_i, z_k) \tag{226}$$
$$p(y_j, z_k) \approx \frac{1}{N} f_o(y_j, z_k) \tag{227}$$
$$p(x_i, y_j, z_k) \approx \frac{1}{N} f_o(x_i, y_j, z_k) \tag{228}$$
$$\forall i \in \{1, ..., n\},\ j \in \{1, ..., m\},\ k \in \{1, ..., q\}$$

where n is the number of discrete values that X can take, m the number of discrete values that Y can take and q the number of discrete values that Z can take. $f_o(v)$ is the frequency of occurrence of a discrete value or value combination v in the data.

For the calculation of the entropies we assume that the estimation of the probabilities is correct (replacing "≈" by "=").

Synergy

$z = x \oplus y$		
x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

$N = 4$
$x_1 = 0, x_2 = 1$
$y_1 = 0, y_2 = 1$
$z_1 = 0, z_2 = 1$

Calculation:

$$p(x_1) = \frac{1}{N} f_o(x_1) = \frac{1}{4} 2 = \frac{1}{2}$$
$$p(x_2) = \frac{1}{N} f_o(x_2) = \frac{1}{4} 2 = \frac{1}{2}$$
$$H(X) = -p(x_1) \cdot log_2(p(x_1)) - p(x_2) \cdot log_2(p(x_2))$$
$$-\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(y_1) = \frac{1}{N} f_o(y_1) = \frac{1}{4} 2 = \frac{1}{2}$$
$$p(y_2) = \frac{1}{N} f_o(y_2) = \frac{1}{4} 2 = \frac{1}{2}$$
$$H(Y) = -p(y_1) \cdot log_2(p(y_1)) - p(y_2) \cdot log_2(p(y_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(z_1) = \frac{1}{N} f_o(z_1) = \frac{1}{4} 2 = \frac{1}{2}$$
$$p(z_2) = \frac{1}{N} f_o(z_2) = \frac{1}{4} 2 = \frac{1}{2}$$
$$H(Z) = -p(z_1) \cdot log_2(p(z_1)) - p(z_2) \cdot log_2(p(z_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(x_1, y_1) = \frac{1}{N} f_o(x_1, y_1) = \frac{1}{4}$$
$$p(x_1, y_2) = \frac{1}{N} f_o(x_1, y_2) = \frac{1}{4}$$
$$p(x_2, y_1) = \frac{1}{N} f_o(x_2, y_1) = \frac{1}{4}$$
$$p(x_2, y_2) = \frac{1}{N} f_o(x_2, y_2) = \frac{1}{4}$$
$$H(X,Y) = -p(x_1, y_1) \cdot log_2(p(x_1, y_1)) - p(x_1, y_2) \cdot log_2(p(x_1, y_2))$$
$$-p(x_2, y_1) \cdot log_2(p(x_2, y_1)) - p(x_2, y_2) \cdot log_2(p(x_2, y_2))$$
$$= -4\frac{1}{4} \cdot log_2\left(\frac{1}{4}\right) = log_2(4) = 2bit$$

$$p(x_1, z_1) = \frac{1}{N} f_o(x_1, z_1) = \frac{1}{4}$$
$$p(x_1, z_2) = \frac{1}{N} f_o(x_1, z_2) = \frac{1}{4}$$
$$p(x_2, z_1) = \frac{1}{N} f_o(x_2, z_1) = \frac{1}{4}$$
$$p(x_2, z_2) = \frac{1}{N} f_o(x_2, z_2) = \frac{1}{4}$$
$$H(X,Z) = H(X,Y) = 2bit$$

$$p(y_1,z_1)=\frac{1}{N}f_o(y_1,z_1)=\frac{1}{4}$$
$$p(y_1,z_2)=\frac{1}{N}f_o(y_1,z_2)=\frac{1}{4}$$
$$p(y_2,z_1)=\frac{1}{N}f_o(y_2,z_1)=\frac{1}{4}$$
$$p(y_2,z_2)=\frac{1}{N}f_o(y_2,z_2)=\frac{1}{4}$$
$$H(Y,Z)=H(X,Y)=2bit$$

$$p(x_1,y_1,z_1)=\frac{1}{N}f_o(x_1,y_1,z_1)=\frac{1}{4}$$
$$p(x_1,y_2,z_2)=\frac{1}{N}f_o(x_1,y_2,z_2)=\frac{1}{4}$$
$$p(x_2,y_1,z_2)=\frac{1}{N}f_o(x_2,y_1,z_2)=\frac{1}{4}$$
$$p(x_2,y_2,z_1)=\frac{1}{N}f_o(x_2,y_2,z_1)=\frac{1}{4}$$
$$H(X,Y,Z)=H(X,Y)=2bit$$

$$Q(X;Y;Z)=H(X,Y)+H(X,Z)+H(Y,Z)-H(X)-H(Y)-H(Z)-H(X,Y,Z)=6bit-5bit=1bit>0bit$$

Note: The probabilities of events (value combinations) which are not observed in the data are considered as zero (e.g. $p(x_2,y_2,z_2)=0$). Impossible events ($p=0$) do not affect the entropy.

Redundancy

$z=\overline{x}=y$		
x	y	z
1	0	0
0	1	1

$$N=2$$
$$x_1=0,x_2=1$$
$$y_1=0,y_2=1$$
$$z_1=0,z_2=1$$

Calculation:

$$p(x_1)=\frac{1}{N}f_o(x_1)=\frac{1}{2}$$
$$p(x_2)=\frac{1}{N}f_o(x_2)=\frac{1}{2}$$
$$H(X)=-p(x_1)\cdot log_2(p(x_1))-p(x_2)\cdot log_2(p(x_2))$$
$$=-\frac{1}{2}\cdot log_2\left(\frac{1}{2}\right)-\frac{1}{2}\cdot log_2\left(\frac{1}{2}\right)=-log_2\left(\frac{1}{2}\right)=log_2(2)=1bit$$

$$p(y_1) = \frac{1}{N} f_o(y_1) = \frac{1}{2}$$
$$p(y_2) = \frac{1}{N} f_o(y_2) = \frac{1}{2}$$
$$H(Y) = -p(y_1) \cdot log_2(p(y_1)) - p(y_2) \cdot log_2(p(y_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(z_1) = \frac{1}{N} f_o(z_1) = \frac{1}{2}$$
$$p(z_2) = \frac{1}{N} f_o(z_2) = \frac{1}{2}$$
$$H(Z) = -p(z_1) \cdot log_2(p(z_1)) - p(z_2) \cdot log_2(p(z_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(x_2, y_1) = \frac{1}{N} f_o(x_2, y_1) = \frac{1}{2}$$
$$p(x_1, y_2) = \frac{1}{N} f_o(x_1, y_2) = \frac{1}{2}$$
$$H(X,Y) = -p(x_2, y_1) \cdot log_2(p(x_2, y_1)) - p(x_1, y_2) \cdot log_2(p(x_1, y_2))$$
$$= -2\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(x_2, z_1) = \frac{1}{N} f_o(x_2, z_1) = \frac{1}{2}$$
$$p(x_1, z_2) = \frac{1}{N} f_o(x_1, z_2) = \frac{1}{2}$$
$$H(X,Z) = H(X,Y) = 1bit$$

$$p(y_1, z_1) = \frac{1}{N} f_o(y_1, z_1) = \frac{1}{2}$$
$$p(y_2, z_2) = \frac{1}{N} f_o(y_2, z_2) = \frac{1}{2}$$
$$H(Y,Z) = H(X,Y) = 1bit$$

$$p(x_1, y_2, z_2) = \frac{1}{N} f_o(x_1, y_2, z_2) = \frac{1}{2}$$
$$p(x_2, y_1, z_1) = \frac{1}{N} f_o(x_2, y_1, z_1) = \frac{1}{2}$$
$$H(X,Y,Z) = H(X,Y) = 1bit$$

$$Q(X;Y;Z) = H(X,Y) + H(X,Z) + H(Y,Z) - H(X) - H(Y) - H(Z) - H(X,Y,Z) = 3bit - 4bit = 0bit < 0$$

Non-Interaction

X is independent from Y, Z and also from the combination of Y and Z.

$z = \overline{y}$		
x	y	z
0	0	1
0	1	0
1	0	1
1	1	0

$$N = 4$$
$$x_1 = 0, x_2 = 1$$
$$y_1 = 0, y_2 = 1$$
$$z_1 = 0, z_2 = 1$$

Calculation:

$$p(x_1) = \frac{1}{N} f_o(x_1) = \frac{1}{4} 2 = \frac{1}{2}$$
$$p(x_2) = \frac{1}{N} f_o(x_2) = \frac{1}{4} 2 = \frac{1}{2}$$
$$H(X) = -p(x_1) \cdot log_2(p(x_1)) - p(x_2) \cdot log_2(p(x_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(y_1) = \frac{1}{N} f_o(y_1) = \frac{1}{4} 2 = \frac{1}{2}$$
$$p(y_2) = \frac{1}{N} f_o(y_2) = \frac{1}{4} 2 = \frac{1}{2}$$
$$H(Y) = -p(y_1) \cdot log_2(p(y_1)) - p(y_2) \cdot log_2(p(y_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(z_1) = \frac{1}{N} f_o(z_1) = \frac{1}{4} 2 = \frac{1}{2}$$
$$p(z_2) = \frac{1}{N} f_o(z_2) = \frac{1}{4} 2 = \frac{1}{2}$$
$$H(Z) = -p(z_1) \cdot log_2(p(z_1)) - p(z_2) \cdot log_2(p(z_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(x_1, y_1) = \frac{1}{N} f_o(x_1, y_1) = \frac{1}{4}$$
$$p(x_1, y_2) = \frac{1}{N} f_o(x_1, y_2) = \frac{1}{4}$$
$$p(x_2, y_1) = \frac{1}{N} f_o(x_2, y_1) = \frac{1}{4}$$
$$p(x_2, y_2) = \frac{1}{N} f_o(x_2, y_2) = \frac{1}{4}$$
$$H(X, Y) = -4\frac{1}{4} \cdot log_2\left(\frac{1}{4}\right) = log_2(4) = 2bit$$

$$p(x_1,z_2)=\frac{1}{N}f_o(x_1,z_2)=\frac{1}{4}$$
$$p(x_1,z_1)=\frac{1}{N}f_o(x_1,z_1)=\frac{1}{4}$$
$$p(x_2,z_2)=\frac{1}{N}f_o(x_2,z_2)=\frac{1}{4}$$
$$p(x_2,z_1)=\frac{1}{N}f_o(x_2,z_1)=\frac{1}{4}$$
$$H(X,Z)=H(X,Y)=2bit$$

$$p(y_1,z_2)=\frac{1}{N}f_o(y_1,z_2)=2\frac{1}{4}=\frac{1}{2}$$
$$p(y_2,z_1)=\frac{1}{N}f_o(y_2,z_1)=2\frac{1}{4}=\frac{1}{2}$$
$$H(Y,Z)=-\frac{1}{2}\cdot log_2\left(\frac{1}{2}\right)-\frac{1}{2}\cdot log_2\left(\frac{1}{2}\right)=-log_2\left(\frac{1}{2}\right)=log_2(2)=1bit$$

$$p(x_1,y_1,z_2)=\frac{1}{N}f_o(x_1,y_1,z_1)=\frac{1}{4}$$
$$p(x_1,y_2,z_1)=\frac{1}{N}f_o(x_1,y_2,z_2)=\frac{1}{4}$$
$$p(x_2,y_1,z_2)=\frac{1}{N}f_o(x_2,y_1,z_2)=\frac{1}{4}$$
$$p(x_2,y_2,z_1)=\frac{1}{N}f_o(x_2,y_2,z_1)=\frac{1}{4}$$
$$H(X,Y,Z)=H(X,Y)=2bit$$

$$Q(X;Y;Z)=H(X,Y)+H(X,Z)+H(Y,Z)-H(X)-H(Y)-H(Z)-H(X,Y,Z)=5bit-5bit=0bit$$

Alternatively the *Interaction Information* can be calculated utilising the independence of X:

$$I(Y,Z;X)=I(Y;X)=I(Z;X)=0bit$$
$$Q(X;Y;Z)=I(X,Y;Z)-[I(X;Z)+I(Y;Z)]=I(Y,Z;X)-[I(X;Y)+I(X;Z)]$$
$$I(Y,Z;X)=H(Y,Z)+H(X)-H(X,Y,Z)=1bit+1bit-2bit=0bit$$
$$I(X;Z)=H(X)+H(Z)-H(X,Z)=1bit+1bit-2bit=0bit$$
$$I(X;Y)=H(X)+H(Y)-H(X,Y)=1bit+1bit-2bit=0bit$$

$$Q(X;Y;Z)=I(Y,Z;X)-[I(Y;X)+I(Z;X)]=0bit-[0bit+0bit]=0bit$$

Table 19: Data and the corresponding entropies and information

$z = \overline{x \cdot y}$		
x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

Entropy						
$H(X)$	$H(Y)$	$H(Z)$	$H(X,Y)$	$H(X,Z)$	$H(Y,Z)$	$H(X,Y,Z)$
1.0000 bit	1.0000 bit	0.8113 bit	2.0000 bit	1.5000 bit	1.5000 bit	2.0000 bit

Information						
$I(X;Y)$	$I(X;Z)$	$I(Y;Z)$	$I(X,Y;Z)$	$I(X,Z;Y)$	$I(Y,Z;X)$	$Q(X;Y;Z)$
0.0000 bit	0.3113 bit	0.3113 bit	0.8113 bit	0.5000 bit	0.5000 bit	0.1887 bit

The NAND Operation usually contains a slight *Synergy* because the information about Z is only complete if the information of X and Y in combination is known. However, the information about Z which X and Y contain separately is not zero. If you assume that $z = \overline{x}$ or $z = \overline{y}$, you will predict z correctly in 3 of 4 cases. In 2 of 4 samples x and y are the same ($x = y$). If these samples are duplicated, x and y are the equal in 4 of 6 samples. Thus the *Redundancy* in the data increases slightly and compensates the *Synergy*.
Hence, it is important to know that the *Interaction Information* also takes into account how often a sample in a dataset exists. If a blackbox system is observed and the output for every possible input is checked, the data usually will not contain information about the frequency of occurrence of a specific input.

$z = \overline{x \cdot y}$		
x	y	z
0	0	1
0	0	1
0	1	1
1	0	1
1	1	0
1	1	0

$$N = 6$$
$$x_1 = 0, x_2 = 1$$
$$y_1 = 0, y_2 = 1$$
$$z_1 = 0, z_2 = 1$$

Calculation:

$$p(x_1) = \frac{1}{N} f_o(x_1) = \frac{1}{6} 3 = \frac{1}{2}$$
$$p(x_2) = \frac{1}{N} f_o(x_2) = \frac{1}{6} 3 = \frac{1}{2}$$
$$H(X) = -p(x_1) \cdot log_2(p(x_1)) - p(x_2) \cdot log_2(p(x_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(y_1) = \frac{1}{N} f_o(y_1) = \frac{1}{6} 3 = \frac{1}{2}$$
$$p(y_2) = \frac{1}{N} f_o(y_2) = \frac{1}{6} 3 = \frac{1}{2}$$
$$H(Y) = -p(y_1) \cdot log_2(p(y_1)) - p(y_2) \cdot log_2(p(y_2))$$
$$= -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = -log_2\left(\frac{1}{2}\right) = log_2(2) = 1bit$$

$$p(z_1) = \frac{1}{N} f_o(z_1) = \frac{1}{6} 2 = \frac{1}{3}$$
$$p(z_2) = \frac{1}{N} f_o(z_2) = \frac{1}{6} 4 = \frac{2}{3}$$
$$H(Z) = -p(z_1) \cdot log_2(p(z_1)) - p(z_2) \cdot log_2(p(z_2))$$
$$= -\frac{1}{3} \cdot log_2\left(\frac{1}{3}\right) - \frac{2}{3} \cdot log_2\left(\frac{2}{3}\right) \approx 0.9182 bit$$

$$p(x_1, y_1) = \frac{1}{N} f_o(x_1, y_1) = 2\frac{1}{6} = \frac{1}{3}$$
$$p(x_1, y_2) = \frac{1}{N} f_o(x_1, y_2) = \frac{1}{6}$$
$$p(x_2, y_1) = \frac{1}{N} f_o(x_2, y_1) = \frac{1}{6}$$
$$p(x_2, y_2) = \frac{1}{N} f_o(x_2, y_2) = 2\frac{1}{6} = \frac{1}{3}$$
$$H(X, Y) = -2\frac{1}{3} \cdot log_2\left(\frac{1}{3}\right) - 2\frac{1}{6} \cdot log_2\left(\frac{1}{6}\right)$$
$$= -\frac{2}{3} \cdot log_2\left(\frac{1}{3}\right) - \frac{1}{3} \cdot log_2\left(\frac{1}{6}\right) = \frac{2}{3} \cdot log_2(3) + \frac{1}{3} \cdot log_2(6) \approx 1.9182 bit$$

$$p(x_1, z_2) = \frac{1}{N} f_o(x_1, z_2) = 3\frac{1}{6} = \frac{1}{2}$$
$$p(x_2, z_2) = \frac{1}{N} f_o(x_2, z_2) = \frac{1}{6}$$
$$p(x_2, z_1) = \frac{1}{N} f_o(x_2, z_1) = 2\frac{1}{6} = \frac{1}{3}$$
$$H(X, Z) = -\frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{6} \cdot log_2\left(\frac{1}{6}\right) - \frac{1}{3} \cdot log_2\left(\frac{1}{3}\right)$$
$$= \frac{1}{2} \cdot log_2(2) + \frac{1}{6} \cdot log_2(6) + \frac{1}{3} \cdot log_2(3) \approx 1.4591 bit$$

$$p(y_1, z_2) = \frac{1}{N} f_o(y_1, z_2) = 3\frac{1}{6} = \frac{1}{2}$$
$$p(y_2, z_2) = \frac{1}{N} f_o(y_2, z_2) = \frac{1}{6}$$
$$p(y_2, z_1) = \frac{1}{N} f_o(y_2, z_1) = 2\frac{1}{6} = \frac{1}{3}$$
$$H(Y, Z) = H(X, Z) \approx 1.4591 bit$$

$$p(x_1, y_1, z_2) = \frac{1}{N} f_o(x_1, y_1, z_2) = 2\frac{1}{6} = \frac{1}{3}$$
$$p(x_1, y_2, z_1) = \frac{1}{N} f_o(x_1, y_2, z_2) = \frac{1}{6}$$
$$p(x_2, y_1, z_2) = \frac{1}{N} f_o(x_2, y_1, z_2) = \frac{1}{6}$$
$$p(x_2, y_2, z_1) = \frac{1}{N} f_o(x_2, y_2, z_1) = 2\frac{1}{6} = \frac{1}{3}$$
$$H(X, Y, Z) = -2\frac{1}{3} \cdot log_2\left(\frac{1}{3}\right) - 2\frac{1}{6} \cdot log_2\left(\frac{1}{6}\right)$$
$$= \frac{2}{3} log_2(3) + \frac{1}{3} log_2(6) \approx 1.9182 bit$$

$$Q(X;Y;Z) = H(X,Y) + H(X,Z) + H(Y,Z) - H(X) - H(Y) - H(Z) - H(X,Y,Z) = 0bit$$

Alternatively the *Interaction Information* can be calculated as follows, simultaneously we can see that the variables are not independent from each other:

$$I(X,Y;Z) = H(X,Y) + H(Z) - H(X,Y,Z) \approx 1.9182bit + 0.9182bit - 1.9182bit = 0.9182bit > 0$$
$$I(X;Y) = H(X) + H(Y) - H(X,Y) \approx 1bit + 1bit - 1.9182bit = 0.0818bit > 0bit$$
$$I(X;Z) = H(X) + H(Z) - H(X,Z) \approx 1bit + 0.9182bit - 1.4591bit = 0.4591bit > 0bit$$
$$I(Y;Z) = H(Y) + H(Z) - H(Y,Z) \approx 1bit + 0.9182bit - 1.4591bit = 0.4591bit > 0bit$$

$$Q(X;Y;Z) = I(X,Y;Z) - [I(X;Z) + I(Y;Z)] = 0.9182bit - [0.4591bit + 0.4591bit] = 0bit$$

B Graphs of Logarithmic Functions

The graphs of the logarithmic functions (Figure 64-66) are useful for a better understanding of the entropy introduced in Section 2.1. The graphs correspond to $log_2(x)$ functions with a base of 2. The graph of a function with $\log_b(x) = \log_2(x) \cdot \log_b(2)$ would only scale the y-axis by $\log_b(2)$.

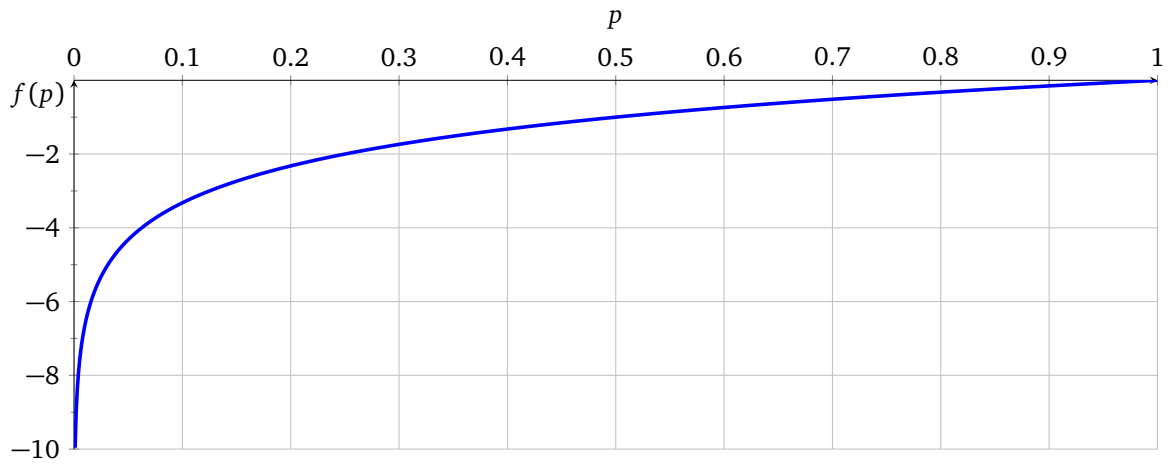

Figure 64: Graph of the function $f(p) = log_2(p)$, where $0 \leq p \leq 1$

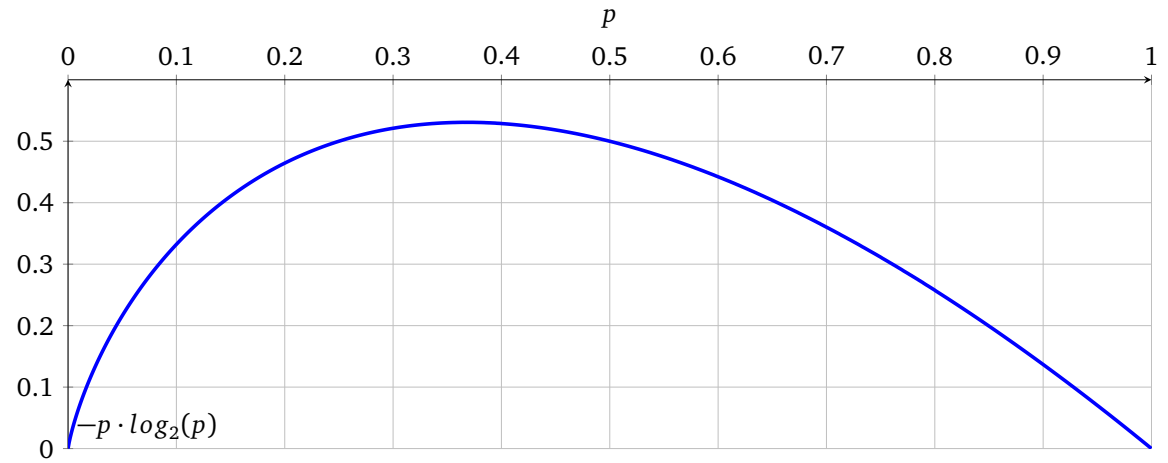

Figure 65: Graph of the function $-p \cdot log_2(p)$

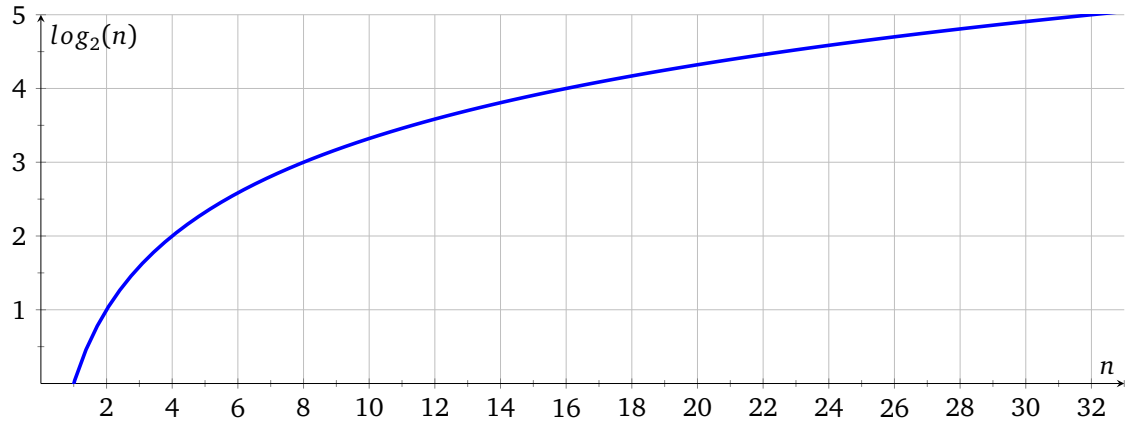

Figure 66: Graph of the function $log_2(n)$, where $1 \leq n$

C Derivation of the Interaction Information by relaxed data sample assignments

Derivation of *Interaction Information* $Q(\widetilde{X};\widetilde{Y};\widetilde{Z})$ by a weighting vector $\boldsymbol{w}$ that weights the influence of all data samples:

$$Q(\widetilde{X};\widetilde{Y};\widetilde{Z}) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q}\widetilde{a}_{i,j,k} \tag{229}$$

$$\widetilde{a}_{i,j,k} = \widetilde{p}_{i,j,k}\left(\log_2(\widetilde{p}_{i,j,k}) - \log_2(\widetilde{p}_{i,j}) - \log_2(\widetilde{p}_{i,k}) - \log_2(\widetilde{p}_{j,k}) + \log_2(\widetilde{p}_i) + \log_2(\widetilde{p}_j) + \log_2(\widetilde{p}_k)\right) \tag{230}$$

The weighting vector consists of h weights because h samples are given.

$$\boldsymbol{w} = (w_1, w_2, ..., w_h)^T,\ 0 \leq w_\nu \leq 1\ \forall\ \nu = 1, ..., h \tag{231}$$

Table 20 shows for every weight the corresponding data sample that is weighted in its frequency of occurrence.

Table 20: Data samples with weighted frequency of occurrence

Weighted data samples			
Frequency	Data samples		
w_1	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
w_2	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_l	$s_{l,1}$	$s_{l,2}$	$s_{l,3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_h	$s_{h,1}$	$s_{h,2}$	$s_{h,3}$

The following functions are designed to allow a consideration of the data samples in the calculation of the probabilities that are needed later in the derivation.

$$\delta(x) = \begin{cases} 0, & x \neq 0 \\ 1, & x = 0 \end{cases} \tag{232}$$

$$d_{l,(i,j,k)} = \delta(s_{l,1} - x_i) \cdot \delta(s_{l,2} - y_j) \cdot \delta(s_{l,3} - z_k) \tag{233}$$
$$d_{l,(i,j)} = \delta(s_{l,1} - x_i) \cdot \delta(s_{l,2} - y_j) \tag{234}$$
$$d_{l,(i,k)} = \delta(s_{l,1} - x_i) \cdot \delta(s_{l,3} - z_k) \tag{235}$$
$$d_{l,(j,k)} = \delta(s_{l,2} - y_j) \cdot \delta(s_{l,3} - z_k) \tag{236}$$
$$d_{l,(i)} = \delta(s_{l,1} - x_i) \tag{237}$$
$$d_{l,(j)} = \delta(s_{l,2} - y_j) \tag{238}$$
$$d_{l,(k)} = \delta(s_{l,3} - z_k) \tag{239}$$
$$\forall\ i = 1, ..., n,\ j = 1, ..., m,\ k = 1, ..., q, l = 1, ..., h$$

The derivation of the *Interaction Information* can be decomposed as follows:

$$\nabla Q(\widetilde{X};\widetilde{Y};\widetilde{Z}) = \begin{pmatrix} \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_1} \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_2} \\ \vdots \\ \frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_h} \end{pmatrix} \tag{240}$$

$$\frac{\partial Q(\widetilde{X};\widetilde{Y};\widetilde{Z})}{\partial w_l} = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q}\frac{\partial \widetilde{a}_{i,j,k}}{\partial w_l} \tag{241}$$

$$\frac{\partial \widetilde{a}_{i,j,k}}{\partial w_l} = \frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i,j,k})}{\partial w_l} + \frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i})}{\partial w_l} + \frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{j})}{\partial w_l} + \frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{k})}{\partial w_l} \tag{242}$$

$$- \frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i,j})}{\partial w_l} - \frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i,k})}{\partial w_l} - \frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{j,k})}{\partial w_l} \tag{243}$$

$$\forall\, l = 1,...,h \tag{244}$$

The seven summands are derived separately.

Summand 1:

$$\frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i,j,k})}{\partial w_l} = \log_2(\widetilde{p}_{i,j,k})\cdot\frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} + \widetilde{p}_{i,j,k}\cdot\frac{\partial \log_2(\widetilde{p}_{i,j,k})}{\partial w_l} \tag{245}$$

$$\frac{\partial \log_2(\widetilde{p}_{i,j,k})}{\partial w_l} = \frac{\partial \log_2(\widetilde{p}_{i,j,k})}{\partial \widetilde{p}_{i,j,k}}\cdot\frac{\partial p_{i,j,k}}{\partial w_l} \tag{246}$$

$$\frac{\partial \log_2(\widetilde{p}_{i,j,k})}{\partial \widetilde{p}_{i,j,k}} = \frac{1}{\ln(2)\cdot\widetilde{p}_{i,j,k}} \tag{247}$$

$$\frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i,j,k})}{\partial w_l} = \left(\log_2(\widetilde{p}_{i,j,k}) + \frac{1}{\ln(2)}\right)\cdot\frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} \tag{248}$$

Summand 2:

$$\frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i,j})}{\partial w_l} = \log_2(\widetilde{p}_{i,j})\cdot\frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} + \widetilde{p}_{i,j,k}\cdot\frac{\partial \log_2(\widetilde{p}_{i,j})}{\partial w_l} \tag{249}$$

$$\frac{\partial \log_2(\widetilde{p}_{i,j})}{\partial w_l} = \frac{\partial \log_2(\widetilde{p}_{i,j})}{\partial \widetilde{p}_{i,j}}\cdot\frac{\partial p_{i,j}}{\partial w_l} \tag{250}$$

$$\frac{\partial \log_2(\widetilde{p}_{i,j})}{\partial \widetilde{p}_{i,j}} = \frac{1}{\ln(2)\cdot\widetilde{p}_{i,j}} \tag{251}$$

$$\frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i,j})}{\partial w_l} = \log_2(\widetilde{p}_{i,j})\cdot\frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} + \frac{\widetilde{p}_{i,j,k}}{\ln(2)\cdot\widetilde{p}_{i,j}}\cdot\frac{\partial p_{i,j}}{\partial w_l} \tag{252}$$

Summand 3:

$$\frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i,k})}{\partial w_l} = \log_2(\widetilde{p}_{i,k})\cdot\frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} + \frac{\widetilde{p}_{i,j,k}}{\ln(2)\cdot\widetilde{p}_{i,k}}\cdot\frac{\partial p_{i,k}}{\partial w_l} \tag{253}$$

Summand 4:

$$\frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{j,k})}{\partial w_l} = \log_2(\widetilde{p}_{j,k})\cdot\frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} + \frac{\widetilde{p}_{i,j,k}}{\ln(2)\cdot\widetilde{p}_{j,k}}\cdot\frac{\partial p_{j,k}}{\partial w_l} \tag{254}$$

Summand 5:

$$\frac{\partial \widetilde{p}_{i,j,k}\log_2(\widetilde{p}_{i})}{\partial w_l} = \log_2(\widetilde{p}_{i})\cdot\frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} + \frac{\widetilde{p}_{i,j,k}}{\ln(2)\cdot\widetilde{p}_{i}}\cdot\frac{\partial p_{i}}{\partial w_l} \tag{255}$$

Summand 6:

$$\frac{\partial \widetilde{p}_{i,j,k} \log_2(\widetilde{p}_j)}{\partial w_l} = \log_2(\widetilde{p}_j) \cdot \frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} + \frac{\widetilde{p}_{i,j,k}}{\ln(2) \cdot \widetilde{p}_j} \cdot \frac{\partial p_j}{\partial w_l} \tag{256}$$

Summand 7:

$$\frac{\partial \widetilde{p}_{i,j,k} \log_2(\widetilde{p}_k)}{\partial w_l} = \log_2(\widetilde{p}_k) \cdot \frac{\partial \widetilde{p}_{i,j,k}}{\partial w_l} + \frac{\widetilde{p}_{i,j,k}}{\ln(2) \cdot \widetilde{p}_k} \cdot \frac{\partial p_k}{\partial w_l} \tag{257}$$

Calculation of $\widetilde{p}_{i,j,k}$ and $\frac{\partial}{\partial w_l}\widetilde{p}_{i,j,k}$ using the "Quotient rule"

$$\widetilde{p}_{i,j,k}(w_1, ..., w_h) = \frac{1}{\left(\sum_{v=1}^{h} w_v\right)} \cdot \sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)} = \frac{w_l \cdot d_{l,(i,j,k)} + b_l}{w_l + c_l} \tag{258}$$

$$c_l = \sum_{v=1, v\neq l}^{h} w_v\,, \quad \frac{\partial c_l}{\partial w_l} = 0 \tag{259}$$

$$b_l = \sum_{v=1, v\neq l}^{h} w_v \cdot d_{v,(i,j,k)}\,, \quad \frac{\partial b_l}{\partial w_l} = 0 \tag{260}$$

$$\frac{\partial \widetilde{p}_{i,j,k}(w_1, ..., w_h)}{\partial w_l} = \frac{\partial}{\partial w_l}\left(\frac{w_l \cdot d_{l,(i,j,k)} + b_l}{w_l + c_l}\right) \tag{261}$$

$$= \frac{d_{l,(i,j,k)} \cdot (w_l + c_l) - (w_l \cdot d_{l,(i,j,k)} + b_l) \cdot 1}{(w_l + c_l)^2} = \frac{c_l \cdot d_{l,(i,j,k)} - b_l}{(w_l + c_l)^2} \tag{262}$$

$$= \frac{\sum_{v=1, v\neq l}^{h} w_v \cdot d_{l,(i,j,k)} - \sum_{v=1, v\neq l}^{h} w_v \cdot d_{v,(i,j,k)}}{(w_l + \sum_{v=1, v\neq l}^{h} w_v)^2} \tag{263}$$

$$= \frac{\sum_{v=1, v\neq l}^{h} w_v \cdot d_{l,(i,j,k)} - \sum_{v=1, v\neq l}^{h} w_v \cdot d_{v,(i,j,k)}}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{264}$$

$$= \frac{\left(\sum_{v=1}^{h} w_v\right) \cdot d_{l,(i,j,k)} - w_l \cdot d_{l,(i,j,k)} - \sum_{v=1, v\neq l}^{h} w_v \cdot d_{v,(i,j,k)}}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{265}$$

$$= \frac{\left(\sum_{v=1}^{h} w_v\right) \cdot d_{l,(i,j,k)} - w_l \cdot d_{l,(i,j,k)} - \left(\sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)}\right) + w_l \cdot d_{l,(i,j,k)}}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{266}$$

$$= \frac{\left(\sum_{v=1}^{h} w_v\right) \cdot d_{l,(i,j,k)} - \left(\sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)}\right)}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{267}$$

$$= \frac{\sum_{v=1}^{h} w_v \cdot (d_{l,(i,j,k)} - d_{v,(i,j,k)})}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{268}$$

$$\forall\ i = 1, ..., n,\ j = 1, ..., m,\ k = 1, ..., q, l = 1, ..., h$$

Calculation of $\frac{\partial}{\partial w_l}\widetilde{p}_{i,j,k}$ using the "Product rule" as approval for the previous result

$$\widetilde{p}_{i,j,k}(w_1,...,w_h) = \frac{1}{\left(\sum_{v=1}^{h} w_v\right)} \cdot \sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)} \tag{269}$$

$$\frac{\partial}{\partial w_l}\left(\frac{1}{\left(\sum_{v=1}^{h} w_v\right)}\right) = \frac{-1}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{270}$$

$$\frac{\partial}{\partial w_l}\left(\sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)}\right) = d_{l,(i,j,k)} \tag{271}$$

$$\frac{\partial \widetilde{p}_{i,j,k}(w_1,...,w_h)}{\partial w_l} = \frac{d_{l,(i,j,k)}}{\left(\sum_{v=1}^{h} w_v\right)} - \frac{\sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)}}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{272}$$

$$= \frac{\left(\sum_{v=1}^{h} w_v\right) \cdot d_{l,(i,j,k)} - \sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)}}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{273}$$

$$= \frac{\left(\sum_{v=1}^{h} w_v\right) \cdot d_{l,(i,j,k)} - \sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)}}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{274}$$

$$= \frac{\sum_{v=1}^{h} w_v \cdot (d_{l,(i,j,k)} - d_{v,(i,j,k)})}{\left(\sum_{v=1}^{h} w_v\right)^2} \tag{275}$$

$$\forall\ i = 1,...,n,\ j = 1,...,m,\ k = 1,...,q, l = 1,...,h$$

Using equations (270) and (269), the derivation can be written in relation to $\widetilde{p}_{i,j,k}$:

$$\frac{\partial \widetilde{p}_{i,j,k}(w_1,...,w_h)}{\partial w_l} = \frac{d_{l,(i,j,k)} - \widetilde{p}_{i,j,k}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \tag{276}$$

$$\forall\ i = 1,...,n,\ j = 1,...,m,\ k = 1,...,q, l = 1,...,h$$

This result can be interpreted as follows:
If the l-th data sample $(s_{l_1}, s_{l_2}, s_{l_3})$ has the exact values (x_i, y_j, z_k), the probability $\widetilde{p}_{i,j,k}$ that the value combination (x_i, y_j, z_k) occurs increases if w_l increases, unless $\widetilde{p}_{i,j,k}$ is already 100%. In that case the derivation is positive or zero (see equation (277)).
In contrast, if the l-th data sample $(s_{l_1}, s_{l_2}, s_{l_3})$ does not exactly contain the values (x_i, y_j, z_k), an increase of w_l would decrease $\widetilde{p}_{i,j,k}$ because the probability of another value combination increases, unless $\widetilde{p}_{i,j,k}$ is already 0% (see equation (278)).

$$\frac{\partial \widetilde{p}_{i,j,k}(w_1,...,w_h)}{\partial w_l} = \frac{1 - \widetilde{p}_{i,j,k}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \geq 0 \text{ case of } (s_{l_1}, s_{l_2}, s_{l_3}) = (x_i, y_j, z_k) \tag{277}$$

$$\frac{\partial \widetilde{p}_{i,j,k}(w_1,...,w_h)}{\partial w_l} = -\frac{\widetilde{p}_{i,j,k}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \leq 0 \text{ case of } (s_{l_1}, s_{l_2}, s_{l_3}) \neq (x_i, y_j, z_k) \tag{278}$$

$$\forall\ i = 1,...,n,\ j = 1,...,m,\ k = 1,...,q, l = 1,...,h$$

Calculation of all probabilities and derivations that are required for the seven summands:

$$\widetilde{p}_{i,j,k}(w_1,...,w_h) = \frac{\sum_{v=1}^{h} w_v \cdot d_{v,(i,j,k)}}{\left(\sum_{v=1}^{h} w_v\right)} \tag{279}$$

$$\frac{\partial \widetilde{p}_{i,j,k}(w_1,...,w_h)}{\partial w_l} = \frac{\sum_{v=1}^{h} w_v \cdot \left(d_{l,(i,j,k)} - d_{v,(i,j,k)}\right)}{\left(\sum_{v=1}^{h} w_v\right)^2} = \frac{d_{l,(i,j,k)} - \widetilde{p}_{i,j,k}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \tag{280}$$

$$\widetilde{p}_{i,j}(w_1,...,w_h) = \frac{\sum_{v=1}^{h} w_v \cdot d_{v,(i,j)}}{\left(\sum_{v=1}^{h} w_v\right)} \tag{281}$$

$$\frac{\partial \widetilde{p}_{i,j}(w_1,...,w_h)}{\partial w_l} = \frac{\sum_{v=1}^{h} w_v \cdot \left(d_{l,(i,j)} - d_{v,(i,j)}\right)}{\left(\sum_{v=1}^{h} w_v\right)^2} = \frac{d_{l,(i,j)} - \widetilde{p}_{i,j}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \tag{282}$$

$$\widetilde{p}_{i,k}(w_1,...,w_h) = \frac{\sum_{v=1}^{h} w_v \cdot d_{v,(i,k)}}{\left(\sum_{v=1}^{h} w_v\right)} \tag{283}$$

$$\frac{\partial \widetilde{p}_{i,k}(w_1,...,w_h)}{\partial w_l} = \frac{\sum_{v=1}^{h} w_v \cdot \left(d_{l,(i,k)} - d_{v,(i,k)}\right)}{\left(\sum_{v=1}^{h} w_v\right)^2} = \frac{d_{l,(i,k)} - \widetilde{p}_{i,k}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \tag{284}$$

$$\widetilde{p}_{j,k}(w_1,...,w_h) = \frac{\sum_{v=1}^{h} w_v \cdot d_{v,(j,k)}}{\left(\sum_{v=1}^{h} w_v\right)} \tag{285}$$

$$\frac{\partial \widetilde{p}_{j,k}(w_1,...,w_h)}{\partial w_l} = \frac{\sum_{v=1}^{h} w_v \cdot \left(d_{l,(j,k)} - d_{v,(j,k)}\right)}{\left(\sum_{v=1}^{h} w_v\right)^2} = \frac{d_{l,(j,k)} - \widetilde{p}_{j,k}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \tag{286}$$

$$\widetilde{p}_{i}(w_1,...,w_h) = \frac{\sum_{v=1}^{h} w_v \cdot d_{v,(i)}}{\left(\sum_{v=1}^{h} w_v\right)} \tag{287}$$

$$\frac{\partial \widetilde{p}_{i}(w_1,...,w_h)}{\partial w_l} = \frac{\sum_{v=1}^{h} w_v \cdot \left(d_{l,(i)} - d_{v,(i)}\right)}{\left(\sum_{v=1}^{h} w_v\right)^2} = \frac{d_{l,(i)} - \widetilde{p}_{i}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \tag{288}$$

$$\widetilde{p}_{j}(w_1,...,w_h) = \frac{\sum_{v=1}^{h} w_v \cdot d_{v,(j)}}{\left(\sum_{v=1}^{h} w_v\right)} \tag{289}$$

$$\frac{\partial \widetilde{p}_{j}(w_1,...,w_h)}{\partial w_l} = \frac{\sum_{v=1}^{h} w_v \cdot \left(d_{l,(j)} - d_{v,(j)}\right)}{\left(\sum_{v=1}^{h} w_v\right)^2} = \frac{d_{l,(j)} - \widetilde{p}_{j}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \tag{290}$$

$$\widetilde{p}_{k}(w_1,...,w_h) = \frac{\sum_{v=1}^{h} w_v \cdot d_{v,(k)}}{\left(\sum_{v=1}^{h} w_v\right)} \tag{291}$$

$$\frac{\partial \widetilde{p}_{k}(w_1,...,w_h)}{\partial w_l} = \frac{\sum_{v=1}^{h} w_v \cdot \left(d_{l,(k)} - d_{v,(k)}\right)}{\left(\sum_{v=1}^{h} w_v\right)^2} = \frac{d_{l,(k)} - \widetilde{p}_{k}(w_1,...,w_h)}{\left(\sum_{v=1}^{h} w_v\right)} \tag{292}$$

$$\forall\ i = 1,...,n,\ j = 1,...,m,\ k = 1,...,q, l = 1,...,h$$

D Old method: Mutation based feature space partitioning algorithm

This method was implemented for the first feature space partitioning tests.
Because the other methods provide better and more robust results, this method was removed from the feature space partitioning section.

Method description
The following simple algorithm was used for the first feature space partitioning experiments and was planed to be improved by including some knowledge about the relations between the feature space partitioning and the resulting values of the *Interaction Information* in the feature subspaces. The only knowledge that was found is that a small change of a feature subspace leads to a small change in the corresponding *Interaction Information* because the *Interaction Information* is influenced by all data samples in the feature subspace. For example, if 5% of the samples in the data of the feature subspace are changed, the *Interaction Information* will be similar to the value before because it is still dominated by the old 95% of the data samples. Apart from that, the change of the *Interaction Information* that is caused by adding or removing a data sample to a feature subspace depends on the cooperation with the rest of the considered data samples. Hence, we make the following assumption:

In two feature subspaces with similar corresponding data samples the **change** of the *Interaction Information* that is caused by the removal or the addition of a certain data sample is similar.

Because of this assumption, it is more promising to test whether the removal or addition of a certain sector brings a desired change in the *Interaction Information* if the last test of this sector was made with a feature subspace that is quite different from the current feature subspace. Therefore, a sector that was removed or added in the current iteration of the algorithm that is causing a benefit, should stay in this current state until a certain number of other sectors in the feature subspace has changed. As consequence an age of a sector assignment was implemented in the algorithm. And a new assignment of a sector is not possible until a at least a certain age of the last assignment change was reached. This assumption restricts the freedom of the otherwise completely random mutation of the sector assignment in a reasonable way. Further knowledge can be included to find additional rules that decrease the freedom of the mutation to increase the convergence of the Algorithm 2.

Algorithm 2 Search Redundancy Subspace

1: **procedure** SEARCH REDUNDANCY SUBSPACE
2: Decompose the entire feature space in sectors of same size and shape.
3: Consider those N sectors which contain at least one data sample and assign every sector $i \in \{1, ..., N\}$ either to the *Redundancy Subspace* ($a_i = 1$) or to the *Rest* ($a_i = 0$), where $1 \leq \sum_{i=1}^{N} a_i \leq (N-1)$ (Random Initialization)
4: Set minimum age Ag_{min} that a sector assignment a_i requires to be able to mutate.
5: At the beginning every *Assignment* is set to an age, where it is ready to mutate $Ag_i = Ag_{min} \forall i \in \{1, ..., N\}$
6: The maximum number of iterations iwp_{max} that are performed without progress before the loop is broken is set.
7: The current number of iterations without progress is set to $iwp = 0$.
8: Calculate the *Interaction Information* Q_{best} in the current *Redundancy Subspace*.
9: **while** $iwp < iwp_{max}$ **do**
10: Calculate $s = \sum_{i=1}^{N} a_i$, which is the number of sectors that are assigned to the *Redundancy Subspace*.
11: Determine the index subset $M \subset \{1, ..., N\}$ of the *Assignments* a_i that can be inverted. The elements of M satisfy $(Ag_i \geq Ag_{min}) \wedge [(2 \leq s \leq N-2) \vee ((s < 2) \wedge \neg a_i) \vee ((s > N-2) \wedge a_i)]$
12: Invert a random *Assignment* a_r, $r \in M$
13: Calculate the *Interaction Information* Q_{new} in the *Redundancy Subspace* after the mutation
14: Increase age of all *Assignments* $Ag_i = Ag_i + 1 \; \forall i \in \{1, ..., N\}$
15: **if** $(Q_{best} > Q_{new}) \vee ((Q_{best} == Q_{new}) \wedge (a_r == 1))$ **then**
16: $Ag_r = 0$
17: $iwp = 0$
18: $Q_{best} = Q_{new}$
19: **else**
20: Invert the *Assignment* a_r to undo the last mutation
21: $iwp = iwp + 1$
22: **end if**
23: **end while**
24: **end procedure**

Advantages of Algorithm 2:
1.) The optimization criterion can easily be changed by replacing line 15. For example, if the ">" sign is replaced by "<" the search *Redundancy* search is modified to a *Synergy* search.
2.) Larger feature subspaces are preferred over smaller feature subspaces if the *Redundancy* value is greater or equal.
Note: A larger *Redundancy* feature subspace leads to a more frequent usage of the simple system model.
3.) Simple implementation and easy modification
4.) Includes the knowledge that is used for the assumption that was explained above.
5.) Stops if no progress is detected for a certain number of iterations.
6.) The complexity of the optimization can be scaled by the number of used sectors.

Disadvantages of Algorithm 2:
1.) The quality of the result differs from run to run.
2.) Due to the mutation of only one sector per iteration, the result converges to a local best result, that strongly depends on the random initialization.

E More fitness function examples for the genetic algorithm

In section 3.2.1 about the genetic algorithm, only two fitness function examples are listed. The rest of the fitness function examples was moved to the appendix:

3.) Goal: *Non-Interaction* in feature subspace 1:

$$f(i) = f(\mathbf{D_{0,i}}, \mathbf{D_{1,i}}, \mathbf{x_i}) = -|Q(\mathbf{D_{1,i}})| - w \cdot \min\{I(X_{1,i}; Y_{1,i}), I(X_{1,i}; Z_{1,i}), I(Y_{1,i}; Z_{1,i})\} \tag{293}$$

where w is the weight of the second term and $X_{1,i}$, $Y_{1,i}$, $Z_{1,i}$ are the variables from $\mathbf{D_{1,i}}$.

4.) Goal: Maximum *Redundancy* in feature subspace 1, while the size of the feature subspace 1 is maximized, too:

$$f(i) = f(\mathbf{D_{0,i}}, \mathbf{D_{1,i}}, \mathbf{x_i}) = -|Q(\mathbf{D_{1,i}})| + w \cdot \sum_{j=1}^{b} x_{i,j} \tag{294}$$

where w is the weight of the second term and $x_{i,j}$ is one entry of $\mathbf{x_i}$.

5.) Goal: Maximum *Redundancy* in feature subspace 0 and maximum *Synergy* in feature subspace 1:

$$f(i) = f(\mathbf{D_{0,i}}, \mathbf{D_{1,i}}, \mathbf{x_i}) = -Q(\mathbf{D_{0,i}}) + Q(\mathbf{D_{1,i}}) \tag{295}$$

where w is the weight of the second term and $x_{i,j}$ is the j-th entry of $\mathbf{x_i}$.

6.) Goal: Maximum *Synergy* in feature subspace 1, while the number of sectors should not fall below n:

$$f(i) = f(\mathbf{D_{0,i}}, \mathbf{D_{1,i}}, \mathbf{x_i}) = Q(\mathbf{D_{1,i}}) + w \cdot \left(\left(1 + e^{-\frac{100}{b} \cdot (x+1-n)}\right)^{-1} + \frac{1}{15} \cdot log_2\left(10^{-5} + \frac{s_i}{100b}\right)\right) \tag{296}$$

$$s_i = \sum_{j=1}^{b} x_{i,j} \tag{297}$$

where w is the weight of the second term and $x_{i,j}$ is the j-th entry of $\mathbf{x_i}$. Figure 69 shows the curve of the second term, which is the sum of the curves from Figure 67 and 68. A minimum number of sectors can be used to force a minimum size of a feature subspace.

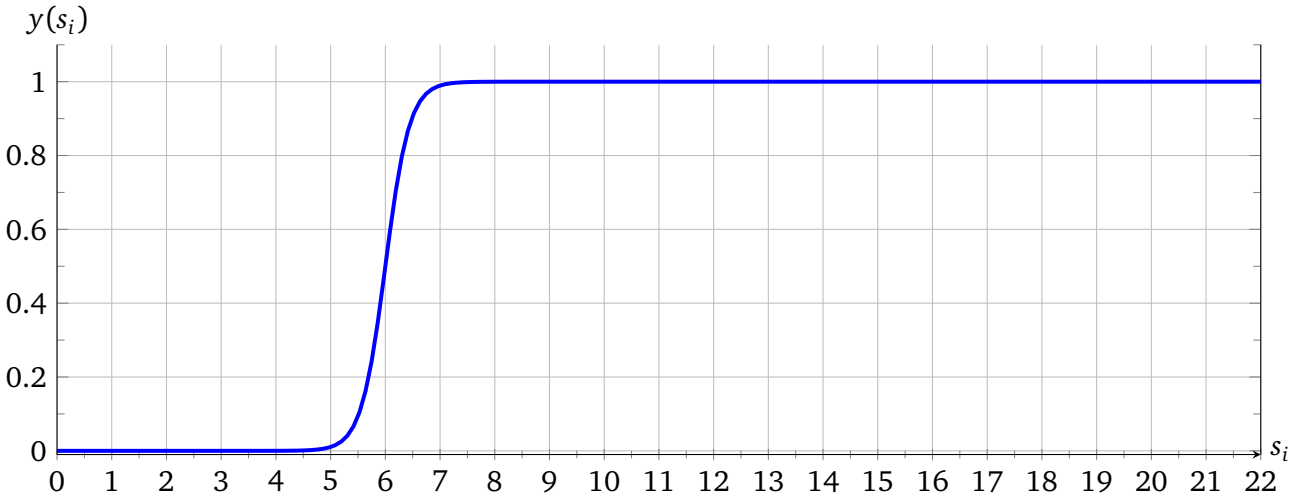

Figure 67: Graph of the sigmoid function $y(s_i) = \left(1 + e^{-\frac{100}{b} \cdot (s_i + 1 - n)}\right)^{-1},\ b = 22,\ n = 7$

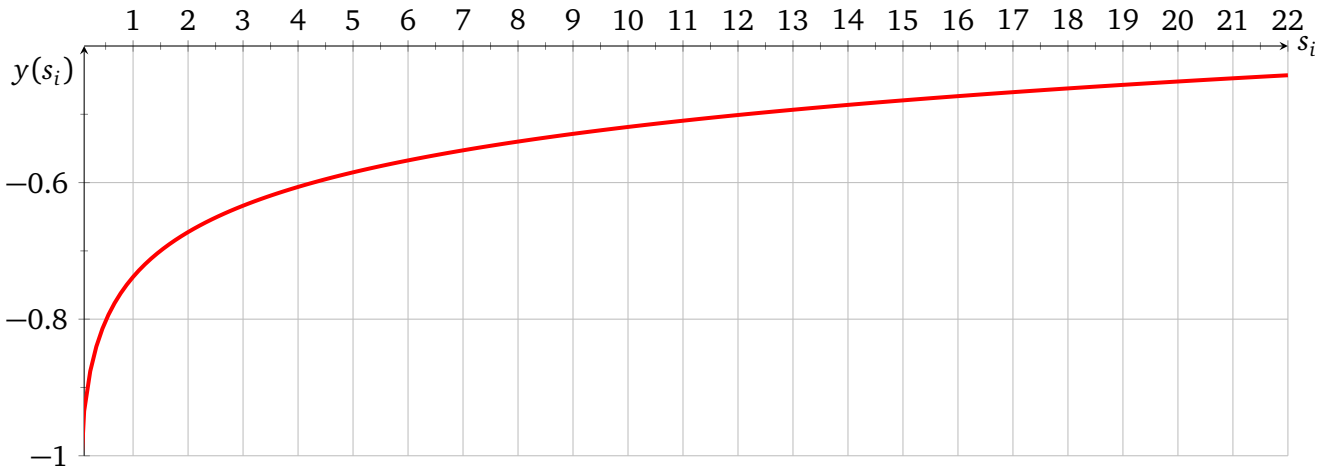

Figure 68: Graph of the logarithmic function $y(s_i) = \frac{1}{15} \cdot log_2(10^{-5} + \frac{s_i}{100b}),\ b = 22,\ n = 7$

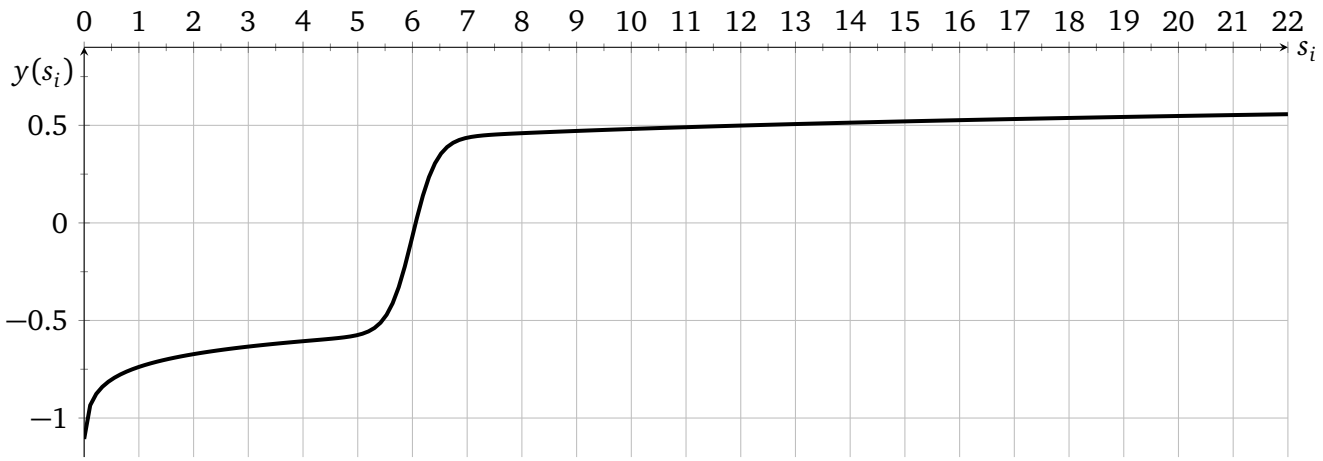

Figure 69: Graph of the function $y(s_i) = \left(1 + e^{-\frac{100}{b} \cdot (s_i + 1 - n)}\right)^{-1} + \frac{1}{15} \cdot log_2(10^{-5} + \frac{s_i}{100b}),\ b = 22,\ n = 7$, which is the second term of a fitness function in equation (296).

F Complete collection of pictures from the feature space partitioning experiment

Due to the large number of pictures that were produced while the partitioning of the feature space using the developed methods in section 4.4.2, the complete collection of 18 pictures (3 datasets) x (3 methods) x (2 goals) were moved to the appendix.

The meaning of the parameters that are noted in the upper part of the pictures is summarized in the following list:

Parameter overview of the feature space partitioning methods (see section 3.2 for more details)

Method 1: Genetic Algorithm
μ: Number of parents, which is equal to the size of the initial population and the number of offspring $\lambda = \mu$.
e: Number of elitists
T: Number of iterations
S: Selection pressure
n_{point}: Number of random crossover points while the recombination
p_m: Probability for every assignment bit in the offspring that it is inverted

Method 2: Gradient Descent
γ: Step size in an optimization step
sgn: If yes then the gradient is used in a signum function.
Δw: The change of every relaxed sector assignment for the approximation of the gradient using the difference quotient
c: Is a parameter of the penalty function that measures the constraint violations of the relaxed sector assignments. $c > 1$ allows a stronger weighted penalty for greater constraint violations.
c_r: The higher c_r the later the penalty function f_p takes a significant effect in the optimization process.
T: Number of iterations

Method 3: Map Reinforcement
γ: Step size in an optimization step
$w_{norm,min}$: Smallest value that occurs in $\hat{\boldsymbol{w}}$ after the normalization of $\hat{\boldsymbol{w}}$.
n_{thr}: Number of w-thresholds that are tested, before the values of $\hat{\boldsymbol{w}}$ are quantized to 0 or 1.
T: Number of iterations

The partitioning results are shown in the following pictures. The sectors that are colored in magenta are the sectors of feature subspace 1, where the value the *Interaction Information* Q_1 is optimized, while Q_0 is the *Interaction Information* of the remaining feature space.

Partitioning of Dataset 1

The following pictures are showing the partitioning results for Dataset 1.

Dataset 1: Results for Method 1:
The partitioning result using Method 1 to maximize Q_1 is shown in Figure 70, while the result using Method 1 to minimize Q_1 is shown in Figure 71. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

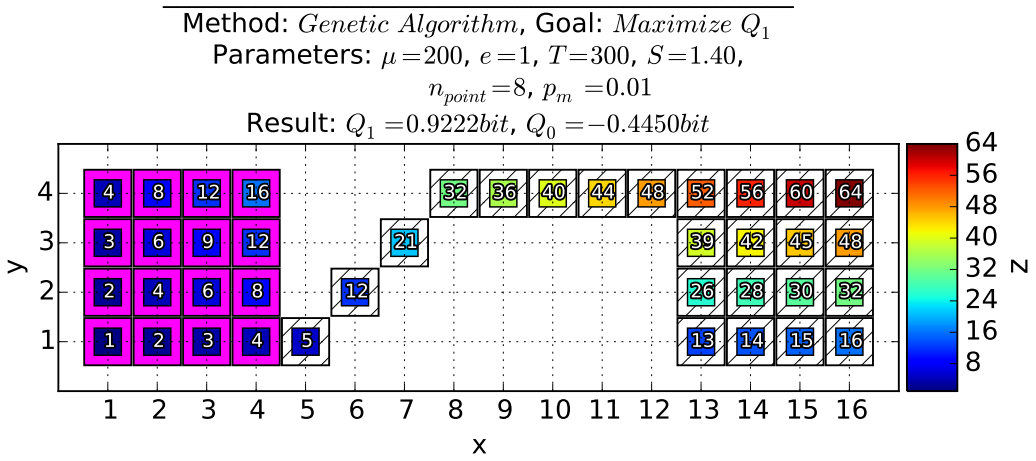

Figure 70: Partitioning of Dataset 1 using Method 1 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

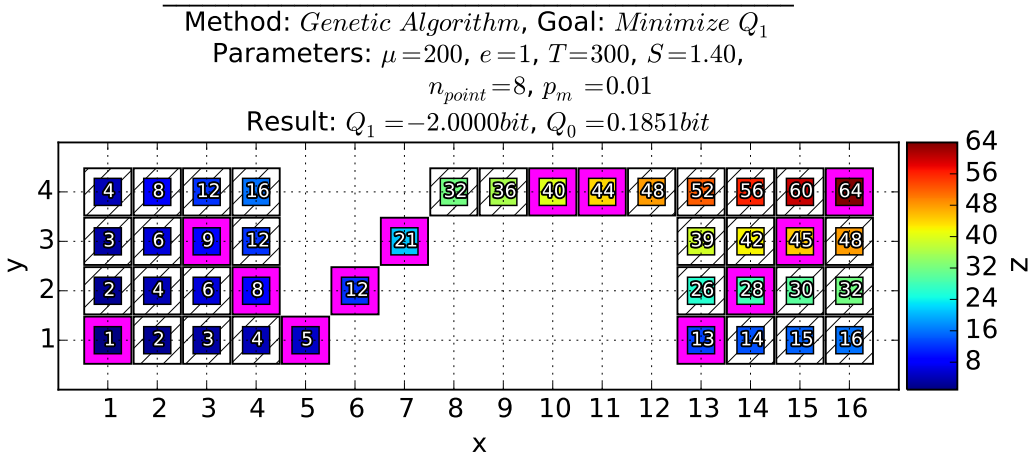

Figure 71: Partitioning of Dataset 1 using Method 1 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

Dataset 1: Results for Method 2:

The partitioning result using Method 1 to maximize Q_1 is shown in Figure 72, while the result using Method 1 to minimize Q_1 is shown in Figure 73. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

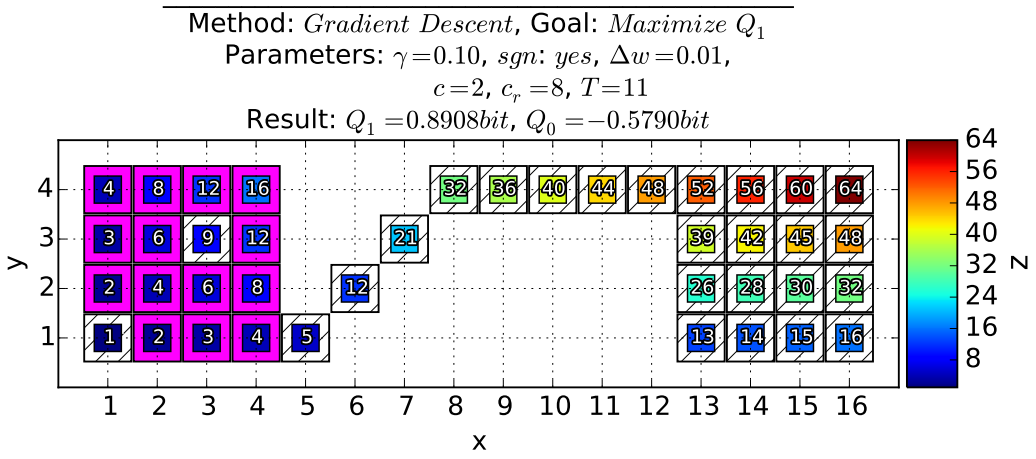

Figure 72: Partitioning of Dataset 1 using Method 2 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

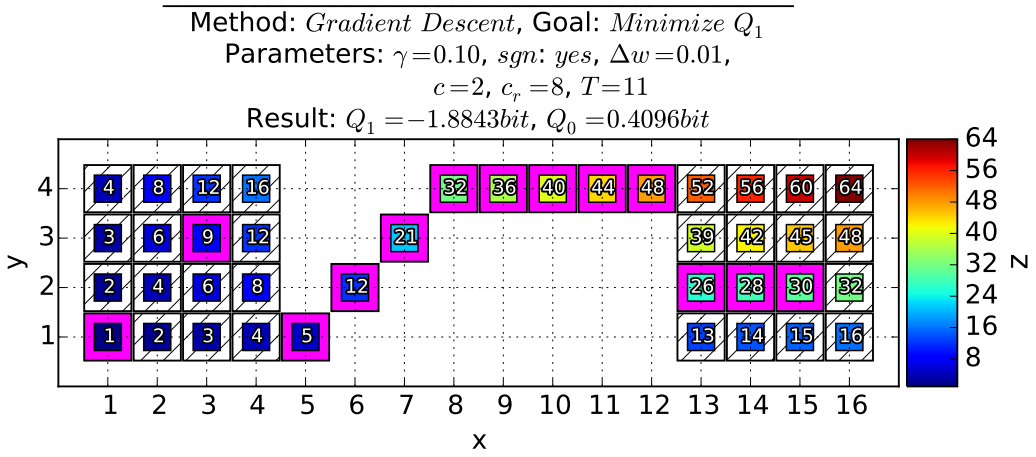

Figure 73: Partitioning of Dataset 1 using Method 2 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

Dataset 1: Results for Method 3:

The partitioning result using Method 1 to maximize Q_1 is shown in Figure 74, while the result using Method 1 to minimize Q_1 is shown in Figure 75. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

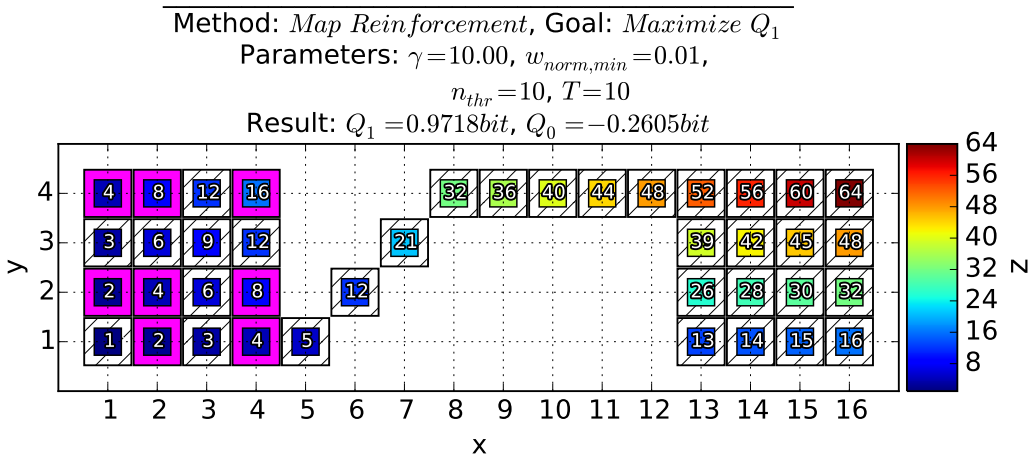

Figure 74: Partitioning of Dataset 1 using Method 3 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

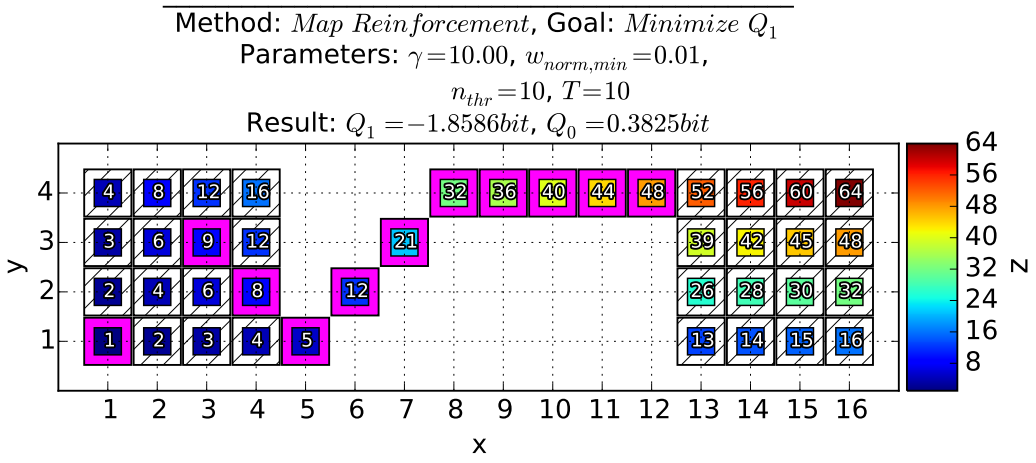

Figure 75: Partitioning of Dataset 1 using Method 3 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

Partitioning of Dataset 2

Dataset 2 has much more data sample positions in the feature space than Dataset 1 and 3. In order to reduce the computation time, the sector size 2x2 were chosen.

Dataset 2: Results for Method 1:
The partitioning result using Method 1 to maximize Q_1 is shown in Figure 76, while the result using Method 1 to minimize Q_1 is shown in Figure 77. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

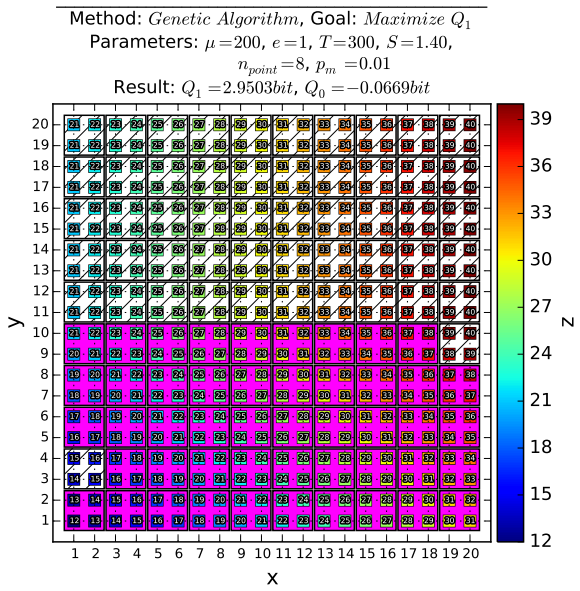

Figure 76: Partitioning of Dataset 2 using Method 1 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

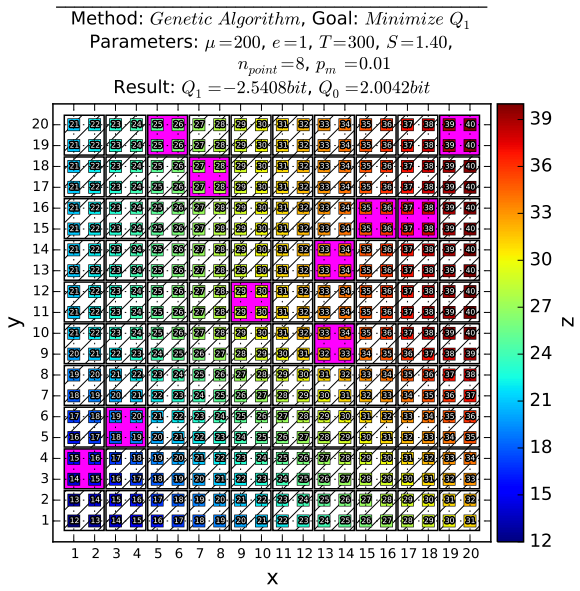

Figure 77: Partitioning of Dataset 2 using Method 1 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

Dataset 2: Results for Method 2:

The partitioning result using Method 1 to maximize Q_1 is shown in Figure 78, while the result using Method 1 to minimize Q_1 is shown in Figure 79. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

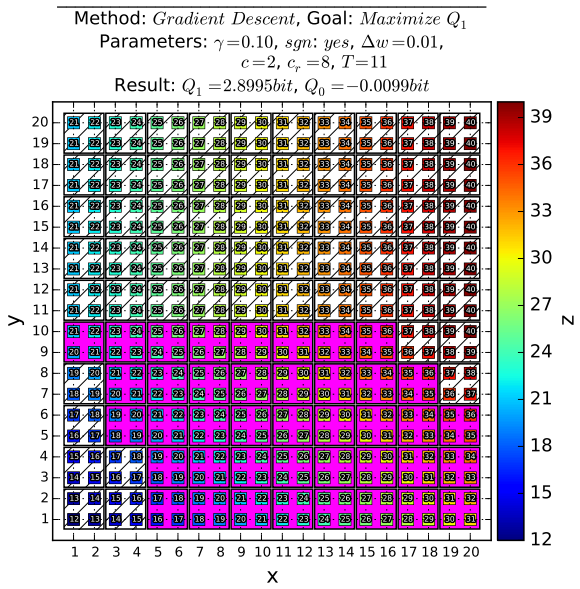

Figure 78: Partitioning of Dataset 2 using Method 2 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

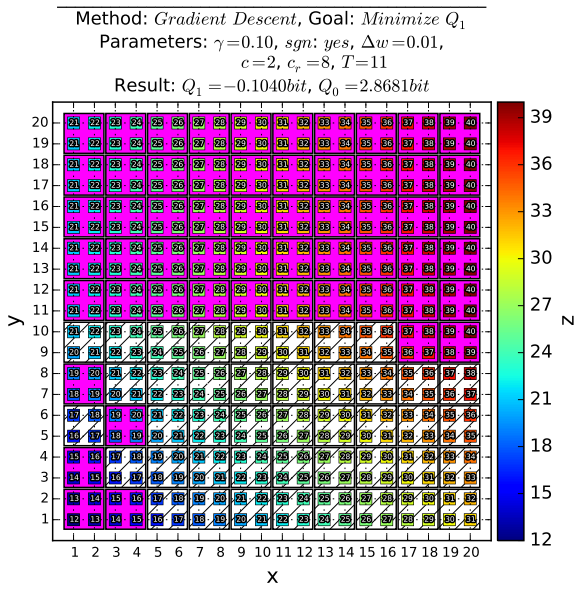

Figure 79: Partitioning of Dataset 2 using Method 2 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

Dataset 2: Results for Method 3:

The partitioning result using Method 1 to maximize Q_1 is shown in Figure 80, while the result using Method 1 to minimize Q_1 is shown in Figure 81. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

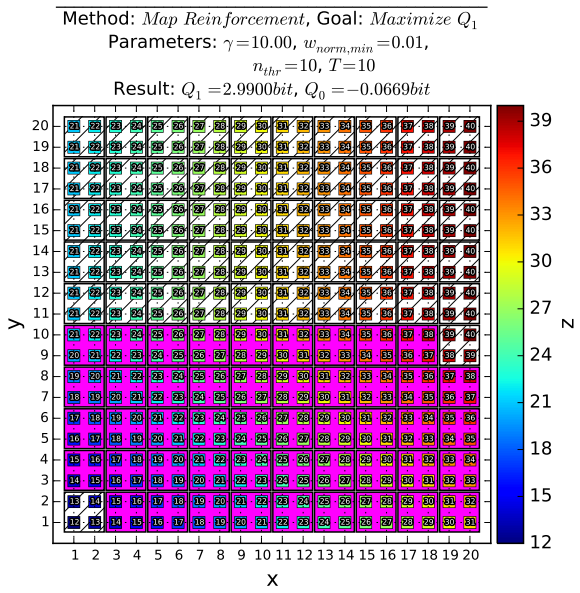

Figure 80: Partitioning of Dataset 2 using Method 3 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

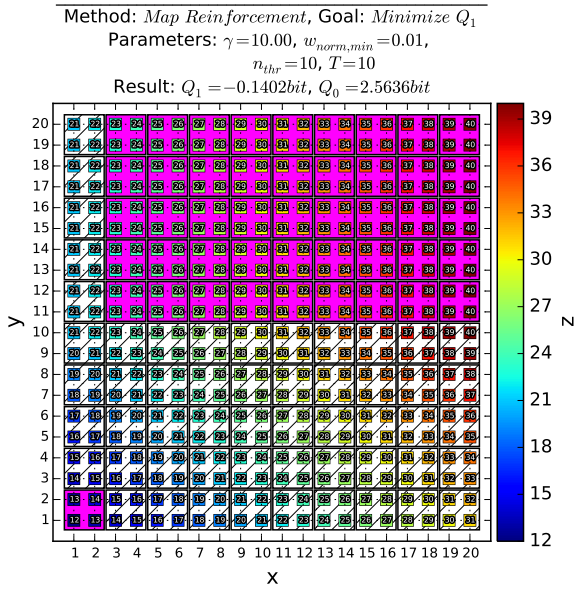

Figure 81: Partitioning of Dataset 2 using Method 3 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

Partitioning of Dataset 3

Dataset 3: Results for Method 1:

The partitioning result using Method 1 to maximize Q_1 is shown in Figure 82, while the result using Method 1 to minimize Q_1 is shown in Figure 83. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding

to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

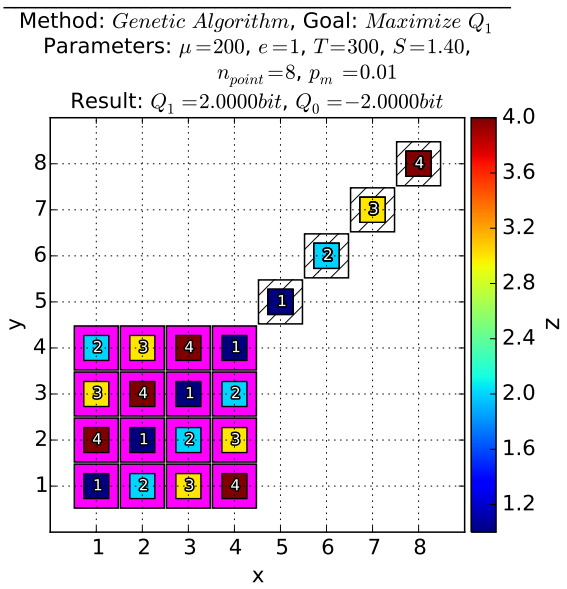

Figure 82: Partitioning of Dataset 3 using Method 1 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

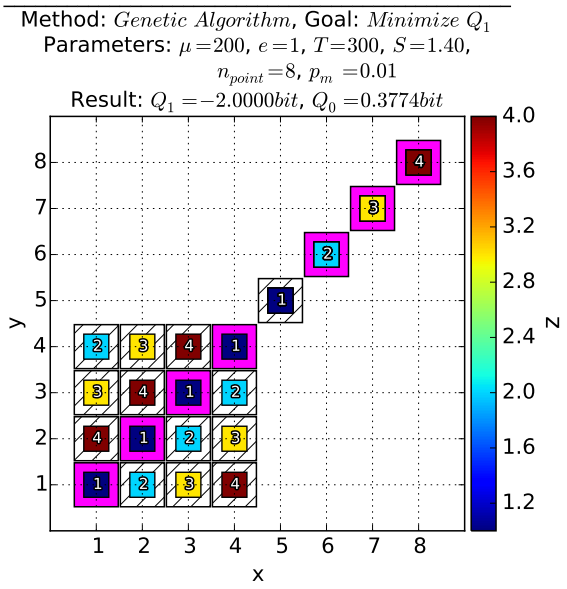

Figure 83: Partitioning of Dataset 3 using Method 1 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

Dataset 3: Results for Method 2:

The partitioning result using Method 1 to maximize Q_1 is shown in Figure 84, while the result using Method 1 to minimize Q_1 is shown in Figure 85. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding

to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

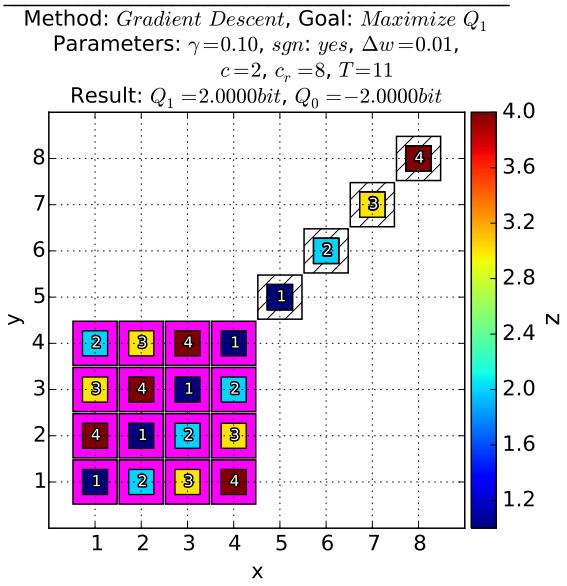

Figure 84: Partitioning of Dataset 3 using Method 2 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

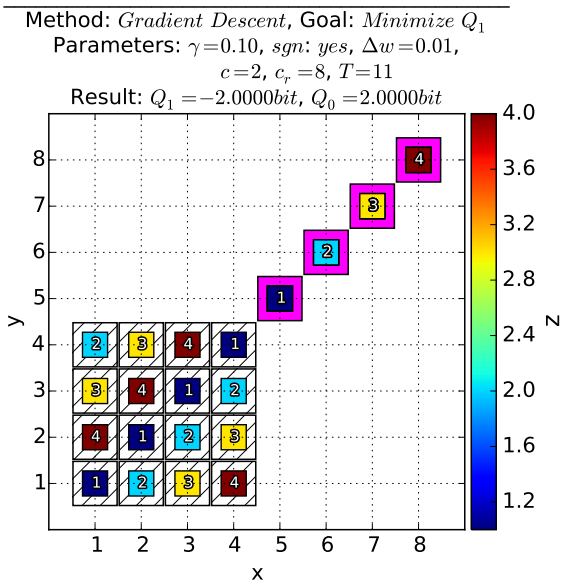

Figure 85: Partitioning of Dataset 3 using Method 2 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

Dataset 3: Results for Method 3:

The partitioning result using Method 1 to maximize Q_1 is shown in Figure 86, while the result using Method 1 to minimize Q_1 is shown in Figure 87. In the upper part of the pictures the results of Q_1 and Q_0 and the exact parameters, that were given to the method, are listed. In both pictures Q_1 is corresponding

to the feature subspace 1 which consists of the magenta colored sectors and Q_0 is corresponding to the feature subspace 0 which consists of the remaining sectors.

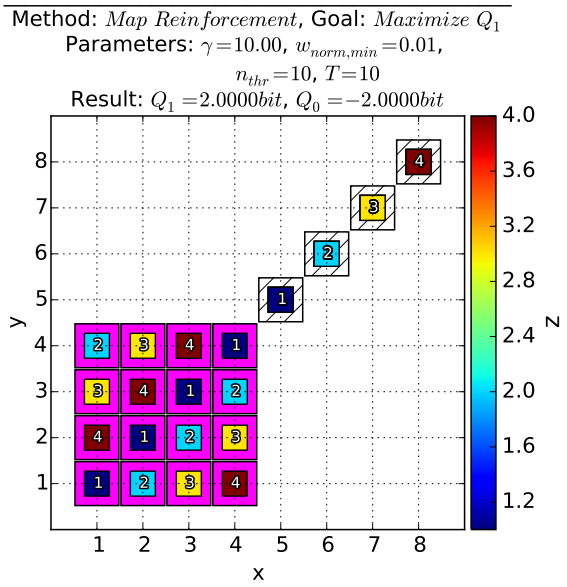

Figure 86: Partitioning of Dataset 3 using Method 3 to maximize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

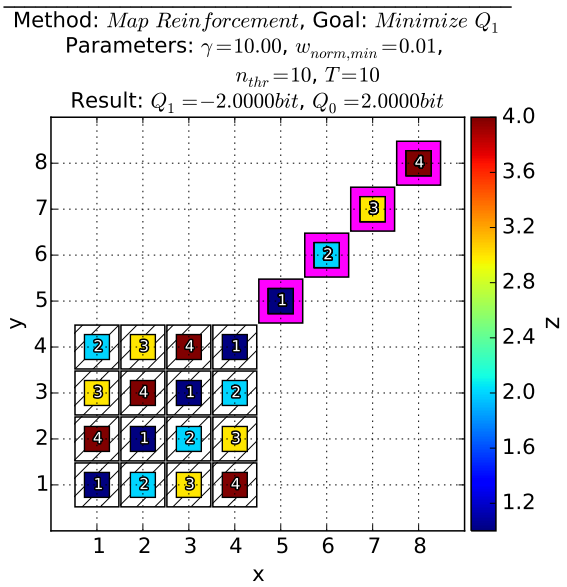

Figure 87: Partitioning of Dataset 3 using Method 3 to minimize Q_1 that corresponds to feature subspace 1 which consists of the magenta colored sectors. Q_0 corresponds to feature subspace 0 which consists of the remaining sectors. The parameters that were used for the method are displayed above.

References

[1] Prof. Dr.-Ing. J. Adamy. Fuzzy Logik, Neuronale Netze und Evolutionäre Algorithmen. Number ISBN 978-3-8440-0397-0. Shaker, 2011.

[2] Ali El Akadi, Abdeljalil El Ouardighi, and Driss Aboutajdine. A powerful feature selection approach based on mutual information. International Journal of Computer Science and Network Security, pages 116–121, 2008.

[3] Hussein Almuallim and Thomas G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. Artificial Intelligence, 69:279–305, 1994.

[4] Dimitris Anastassiou. Computational analysis of the synergy among multiple interacting genes. Molecular Systems Biology 3 (2007): 83. PMC, Web. 12 Aug. 2015.

[5] Anthony J. Bell. The co-information lattice. In in Proc. 4th Int. Symp. Independent Component Analysis and Blind Source Separation, pages 921–926, 2003.

[6] Lars Gräning. Shape Mining: Knowledge Extraction from Engineering Design Data. PhD thesis, 2014.

[7] Lars Gräning and Thomas Ramsay. Flow field data mining based on a compact stream line representation. In SAE 2015 World Congress & Exhibition. SAE International, April 2015.

[8] Robert M. Gray. Entropy and Information Theory. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[9] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. J. Mach. Learn. Res., 3:1157–1182, March 2003.

[10] Jiawei Han. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[11] Aleks Jakulin and Ivan Bratko. Analyzing attribute dependencies. In PKDD 2003, volume 2838 of LNAI, pages 229–240. Springer-Verlag, 2003.

[12] Klaus Krippendorff. Ross ashby's information theory: a bit of history, some solutions to problems, and what we face today. Int. J. General Systems, 38(2):189–212, 2009.

[13] Ariel Kulik and Hadas Shachnai. On lagrangian relaxation and subset selection problems. In In Proc. 6th Workshop on Approximation and Online Algorithms, pages 160–173, 2009.

[14] Mehal Patel. Bayesian classification using probabilistic graphical models. 2014.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[16] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27:1226–1238, 2005.

[17] Duc Truong Pham and Gonzalo A. Ruz. Unsupervised training of bayesian networks for data clustering. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 465(2109):2927–2948, 2009.

[18] Michael Rath. Information theoretic approach to the investigation of design and flow feature interactions. Master's thesis, 2011.

[19] Lior Rokach and Oded Maimon. Data Mining with Decision Trees: Theory and Applications. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2008.

[20] Mehran Sahami. Learning limited dependence bayesian classifiers. In In KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pages 335–338. AAAI Press, 1996.

[21] Tom Schaul, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. PyBrain. Journal of Machine Learning Research, 11:743–746, 2010.

[22] Claude Shannon. A mathematical theory of communication. Bell System Technical Journal, 27:379–423, 623–656, July, October 1948.

[23] Nicholas Timme, Wesley Alford, Benjamin Flecker, and John M. Beggs. Towards resolving the multivariate information controversy. CoRR, abs/1111.6857, 2011.

[24] Paul L. Williams and Randall D. Beer. Nonnegative decomposition of multivariate information. CoRR, abs/1004.2515, 2010.

[25] Chen Yanover, Talya Meltzer, Yair Weiss, P. Bennett, and Emilio Parrado-hernandez. Linear programming relaxations and belief propagation - an empirical study. Journal of Machine Learning Research, 7:2006, 2006.