Czech Technical University in Prague

Faculty of Transportation Sciences

Department of Applied Mathematics

Modeling of Extensive Files of Discrete Data

DOCTORAL THESIS

Ing. Šárka Jozová

Supervisor: doc. Ing. Ivan Nagy, CSc.

Study Programme: Engineering Informatics

Field of Study: Engineering Informatics in Transportation and Telecommunications

Prague 2023

Acknowledgment

I would like to thank my supervisor doc. Ing. Ivan Nagy, CSc., who has supported me throughout my doctoral studies and whose cooperation and expertise in mathematical modeling helped me to write this thesis. His tolerant and caring attitude made it possible for me to complete this thesis even during my parenthood, so it was an honor to work with such a brilliant personality.

I would also like to thank my whole family and especially my partner for his helpfulness and patience. And also to my parents for their support during my studies and for taking care of my son during the writing of this thesis. Additional thanks to Miroslav Vaniš, for his valuable advice during my doctoral studies.

Declaration

I hereby declare that this doctoral thesis Modeling of Extensive Files of Discrete Data was written solely by me under the professional guidance of my supervisor and with the use of literature and other sources of information, all of which are cited in the text and listed at the end of the thesis.

In Prague, June 30, 2023 .....

Ing. Šárka Jozová

Abstract

Data analysis is an important method for many decision-making processes that are used to extract information from data. Common data sources are questionnaires, which provide primarily discrete data. The analysis of discrete data often fails due to the high dimension and the large number of parameters. Therefore, finding solutions to these problems is essential and useful.

The thesis deals with the prediction of discrete data from questionnaires in the field of transportation and medicine. The proposed solution is based on modeling of the explanatory variables using marginal mixtures (models of individual variables under the assumption of their independence) and the construction of categorical prediction models locally on found clusters. This approach reduces the number of parameters and the overall dimension of the model by assuming the independence of the mixtures and using the binomial distribution in the components of these mixtures.

To verify the accuracy of the constructed prediction model, experiments are performed using real data. The results are then compared with existing prediction methods, specifically k-nearest neighbor, decision tree, neural networks, logistic regression, naive Bayes, and fuzzy rules.

Keywords

Accident data, binomial model, discrete data, extensive files, marginal mixtures, mixture of binomial distributions, multimodal data, prediction.

Contents

List of notations

Notations in the theoretical part	Notations in the Scilab code	Meaning of the notations
t	t	discrete time
T	nd	number of data
x_t	x(t)	explanatory variable at time t
y_t	y(t)	target variable at time t
c_t	c(t)	pointer at time t
$d_t = \{y_t, x_t\}$	(*)	pairs of target and explanatory variables at time t
$d\,(t) = \{d_0, d_1, \cdots, d_t\}$	(*)	sequence of all data
$\hat{x}$	(*)	measured data
$\hat{y}$	yp	prediction of target y
Θ, p	C(j).p (*)	model parameters
$\bullet_{k\|lm}$	–	multiindex (2.2)
S_t	C(j).S X(i).c(j).S	statistics
κ	X(i).ka	counter
$\hat{p}$	C(j).pE X(i).c(j).pE	point estimate of parameter p
w	X(i).w	weight
α	–	stationary probability weight

(*) basic notations modified according to specific situation.

The thesis deals with variables in the form of random sequences, i.e. random variables indexed by discrete time $t = 0, 1, 2, \cdots$ where $t = 0$ relates to prior data (or generally prior information). E.g. x_t, y_t are explanatory and target variables monitored at time t (and at the same time their values measured at time t), respectively. $f\,(y_t|x_t)$ is a model of the target y depending on the explanatory variable x, all at time t. In the following text, if the variables occur all in the same time (expressions without the time evolution), the index t will be omitted for clarity. Thus, $f\,(y_t|x_t)$ and $f\,(y|x)$ have the same meaning if t is just a time index without any relation to previous and next indexes.

List of models

Models in the theoretical part	Explanation of the models
$f(y,x)$	joint model
$f(x)$	marginal model
$f(y\|x)$	conditional model
$f(\Theta\|d(t))$	parameter model dependent on data
$f(x_i\|c_j,p) = f_j(x_i\|p)$	model of j-th component in i-th variable
$f(\hat{x}_i\|c_j,\hat{p}) = f_j(\hat{x}_i\|\hat{p})$	proximity of j-th component in i-th variable
$f(x_i\|y,c_j,\Theta) = f_j(x_i\|y,\Theta)$	local model

For clarity $i \in \{1,2,\cdots,n\}$ denotes the variables, where n (nv in Scilab) is the number of variables, and $j \in \{1,2,\cdots,m\}$ denotes the components, where m (nc in Scilab) is the number of components.

Chapter 1

Introduction

The motivation for this thesis was the task of accident analysis in the Prague area with the aim of classifying traffic accidents according to their severity depending on the circumstances of the accident. The circumstances were e.g. weather conditions, time of day, presence of alcohol, etc. The data were obtained from the records of the Czech Police in the form of a questionnaire (as a set of answers to questions from one type of questionnaire). The questions are prescribed and specific answers are given according to the accident in a question, so it is almost entirely discrete data - the number of possible answers for each question is finite and the answer is simply marked. For a good set of discrete data from a questionnaire, it is important to have appropriately chosen questions that match what we want to find out. It is also necessary to have a large number of records and the most accurate information (answering all questions). This results in extensive files of discrete data with a high dimension. While solving this particular task of traffic accident classification, it became clear how few suitable methods are available for analyzing discrete (survey) data. The same problem arose later in the analysis of the medical data obtained again from the questionnaire survey.

The basis of this thesis is the elaboration of the marginal mixtures method for the analysis of extensive files of discrete data. This means that the range of values of these data is countable and the model takes on enormous dimensions. Data analysis using descriptive statistics is simple, but the aim of this thesis is to look for relationships between variables. In practice, much attention is paid to continuous models, especially linear regression, so that continuous data are explored more than discrete data. The most common description of discrete data is through categorical models that can be written in the form of a table. Each table entry corresponds to a configuration of the values of the

variables that appear in the model, and the model assigns a probability to each configuration. A great advantage of such models is their general form - each situation that can occur in a modeled system is described separately. However, the big disadvantage of the categorical models is their size, which means a high dimension of the table expressing this distribution for more variables and values and their overparameterization. For example, a model with say 10 variables, each with 8 different values on average, has dimension 8^{10}. This is called the curse of dimensionality [1]. This makes common categorical models almost unusable in larger practical tasks. Therefore, it is necessary to create a new discrete prediction model that reduces the number of parameters and thus the dimension of the model. Correct choice of the model and estimation of its parameters is the basic presumption for successful solution of the target task. If the model is designed with the wrong structure (selection of variables that affect the modeled variable) or the model is misspecified (insufficient or inappropriate data), the result is likely to be incorrect. Therefore, the design of the model and the collection of sufficient valid data for correct estimation are essential.

The basic assumption that in general allows for a reduction of the model dimension is the independence of the variables from each other. For the aforementioned case of 10 variables with 8 possible values, the assumption of independence reduces the dimension from 8^{10} to $8 * 10$. However, the independence has to be justified somehow. In our approach, we assume that the variables are multimodal. This means that the generation process works in several different modes, and we assume independence within these working models. Then we estimate the variable models in the form of mixtures. Each component of the mixture reflects the data of individual working modes - so called data clusters. A mixture models not only the data clusters by its components but also a pointer variable that describes the transitions between the components (working modes). And while the data in the clusters are assumed to be independent, the dependency of the pointers of individual variables is left. The assumption is that the behavior of the variables is projected into these pointers, and through them the important connection between the variables is preserved.

In addition to independence assumptions, another way to reduce model dimensionality is to replace the categorical distribution with another discrete distribution with a smaller number of parameters while maintaining model quality. The binomial distribution was chosen as a suitable one because it is determined by only one parameter and this allows to shape the probability function of the binomial distribution well.

The presented research is based on the construction of a model that estimates (predicts) the value of

the target variable on the basis of measured explanatory variables (circumstances of the phenomenon), based on the above mentioned simplifications (independence assumptions, distribution replacement). This value determines the class to which the measured vector of values of explanatory variables will be classified. Thus, the prediction of a discrete target variable is understood here as a classification in the data space of explanatory variables.

The process is as follows: We observe a certain phenomenon that is described by a target variable. The values of the target variable are influenced by explanatory variables. All variables (target and explanatory) are discrete. The goal of the problem is to construct a model that classifies a given vector of measured explanatory variables into a class given by the values of the target variable. For example, in the task of analyzing traffic accidents, it will be: Target variable - traffic severity (e.g. with values: light, heavy, with injury, with death). Explanatory variables - road surface, lighting conditions, time of day, accident speed, etc. Each variable has multiple values and continuous variables are discretized - e.g. speed: by regulation, exceeded. Classes correspond to the values of the target variable. For example, the vector “smooth road surface”, “dusk”, “evening”, “speed exceeded” is likely to be classified as “light”. However, this is only a guess. The data will reveal the real classification.

The principle of the proposed method is already described, but it is necessary to show how useful this method can be in practice: We plan a new route of a traffic road. At any (suspicious) location, we can enter the corresponding values of the explanatory variables and find out whether this location will be safe or what degree of danger is associated with it. At the same time, we can experimentally determine which values of the explanatory variables need to be changed to make the location safer.

Data from other fields such as medicine, aviation, sociological research, and many others can be used in a similar way. This means that the proposed method can be used in the other fields where data are collected from questionnaires. Therefore, not only accident data but also car data and medical data were used for the experiments in this thesis.

1.1 State of the art

The thesis is based on the modeling of discrete questionnaire data using marginal mixtures, especially in the area of traffic accidents. The following sources [2, 3, 4, 5, 6] deal with the analysis of discrete accident data. The first of these studies [2] investigates the relationship between real-time traffic data and crash risk of reduced visibility related (VR) crashes. The measured data are collected from (i) the Automatic Vehicle Identification (AVI) sensors and (ii) loop/radar detectors (LDs). The study

also solves the problem of of data suitability for predicting VR crashes. The next study [3] describes the conceptual and mathematical development of an accident occurrence model that incorporates accident and exposure data in a mathematically consistent level of disaggregation using principles of survival theory. The model predicts the probability of being involved in an accident at a specific time, given that a vehicle has survived to that time. Several alternative functional forms are discussed, including additive, proportional hazards, and accelerated failure time models. Study [4] is devoted to the development of methods for analyzing highway accident data. It provides guidance in defining these challenges and opportunities by first reviewing the evolution of methodological applications and available data in highway accident research. Based on this review, directions for future methodological developments are identified and that new data sources will play the role in defining these directions. The same author published another study [5], also dealing with highway accidents. It presents a detailed discussion of the problem typically referred to an unobserved heterogeneity of traffic accidents in the context of accident data and analysis. Various statistical approaches available to deal with this unobserved heterogeneity are presented along with their strengths and weaknesses. The paper concludes with a summary of fundamental issues and directions for future methodological work dealing with this topic. The aim of the latter study [6] is to investigate the severity of incidents as a function of different accident circumstances. The description of these circumstances leads to the use of a large number of different variables (about 50 variables) and most of them are discrete. The majority of statistical methods that deal with discrete variables use a frequency table. This approach is not suitable for traffic data due to its high dimension. This paper offers several methods are proposed to solve the problem with high-dimensional traffic data.

In general, the analysis of questionnaire data is affected by the uncertainty of responses obtained without direct interaction with respondents. This uncertainty leads to a limited number of response options that may not be appropriate for all of these respondents [7]. The analysis is also affected by missing data [8], measurement error [9], unrepresentative samples [10], unfavorable data heterogeneity [5], etc. As mentioned in the introduction, the huge amount of traffic accident questionnaire data with many values leads to a high dimension of the whole model, which is a common problem in this field [11]. A similar problem is posed by a large number of discrete explanatory variables, resulting in a low probability of several levels of the accident severity in the predictive model [12]. Since we want to reduce the dimension of the model while preserving the information in unreduced tables, it is not appropriate to use categorical models. Therefore, we use conditional probability functions and mixture

models, which reduce the dimension but do not lead to loss of information from the questionnaires [13]. A mixture model can be created from any distribution. A commonly used distribution for modeling discrete data with mixtures is the Poisson distribution. This is addressed in the study [14], which uses Poisson mixture estimation with the goal of prediction. The estimation can also be based on a mixture of gamma distributions [15], as well as others. Therefore, we have previously conducted a study [16] that selected the binomial distribution as suitable for modeling and estimation. Subsequently, it was verified that the binomial mixture distribution is ideal for constructing a predictive model for large discrete data sets while maintaining its tolerable dimension, since the chosen distribution uses only one parameter [17].

Approaches to the analysis of discrete questionnaire data vary widely. In the literature, there are simple solutions such as the use of hypothesis testing or proportion estimation to advanced classification methods that depend on the specific task. In this book [18], the proposed model is based on the analysis of discrete data using the Bayesian approach and general mixture theory, as well as on mixtures specifically focused on discrete data [19]. The theory of Bayesian mixture estimation is also developed in [20, 21]. The approach for estimating recursive dynamic mixtures for different types of distributions was then generalized and derived for binomial mixtures [22]. Recursion allows algorithms to continuously use the available explanatory variables used for prediction.

In the field of discrete data analysis, it is also possible to use classification methods, specifically data mining methods such as k-nearest neighbors [23], decision tree [24], neural networks [25], logistic regression [26], naive Bayes classifier [23], fuzzy rules [27], and others. However, for modeling and estimation of these mixtures, studies are based on prediction using the iterative expectation-maximization (EM) algorithm [28, 29, 30, 31] or the Bayesian approach [14, 15, 32, 33]. The EM algorithm works offline, i.e. it requires a complete data set for evaluation, and the results cannot be obtained sequentially because the convergence time is not guaranteed. However, we also want to obtain results in real time, so it is appropriate to use only a Bayesian approach, which is iteration free, data can be added sequentially, and results can be obtained at any time.

1.2 Summary of the presented problem

The objective of this thesis is to develop a method for modeling and estimating of extensive files of discrete data with the goal of classifying the target variable (e.g., accident severity). Existing categorical models are not suitable for this reason due to their high dimension, commonly known as

the curse of dimensionality. The proposed solution basically consists of two main parts. The first, which is crucial, is the modeling of the explanatory data by a model of independent mixtures, where each variable is described by a scalar mixture of binomial components. The second part deals with the construction of local predictive models based on data from individual clusters and corresponding target variables. The proposed solution is tested on both simulated and real data.

1.3 The dissertation structure

In order to successfully meet the goals of the thesis, this section determines the general procedure to achieve them. The chosen approach is based on processing data consisting of explanatory variables x and target variable y.

1. Construction of an algorithm for estimation and classification of discrete data:
   (a) creation of the joint model of independent explanatory variables and system output in the factorized form
   $$f(y,x) = f(x)\, f(y|x),$$
   (b) analysis of the data space x,
   (c) creation of the local models $f(x|y)$,
   (d) construction of predictive model $f(y|x)$ using the naive Bayes principle.
2. Simulation testing of the algorithm:
   (a) verification of the theoretical part and development (debugging) of the program,
   (b) the simulation is an imitation of the real situation (e.g. 2 or 3 modes).
3. Verification of correctness of the proposed model - whether the data meet the assumptions.
4. Testing programs on real data.
5. Comparison of results with known classification methods.

Chapter 2

Preliminaries

This chapter describes important terms needed to understand the problem presented in Section 1.2. These terms include chain and Bayes rule, Bayesian and discrete models. Another important term is the classification, which is understood here as the estimation of the values of the discrete target variable y. These estimates classify the explanatory variables x.

2.1 Chain rule

Chain rule performs factorization of a joint probability density function $f(x_1, x_2, \cdots, x_n)$ to the product of conditional ones

$$f(x_1, x_2, \cdots, x_n) = f(x_1|x_2, x_3, \cdots, x_n) f(x_2|x_3, x_4, \cdots, x_n) \cdots f(x_n).$$

It can be obtained by recursively using the definition of the conditional probability density function $f(A|B) = f(A, B)/f(B) \rightarrow f(A, B) = f(A|B) f(B)$. Let x_1 be A and $\{x_2, x_3, \cdots, x_n\}$ be B, then

$$f(x_1, x_2, \cdots, x_n) = f(A|B) f(B) = f(x_1|x_2, x_3, \cdots, x_n) f(x_2, x_3, \cdots, x_n)$$

and again let $x_2 = A_1$ and $\{x_3, x_4, \cdots, x_n\} = B_1$ and we can continue

$$f(x_1, x_2, \cdots, x_n) = f(x_1|x_2, x_3, \cdots, x_n) f(x_2, x_3, \cdots, x_n) = f(x_1|x_2, x_3, \cdots, x_n) f(A_1|B_1) f(B_1) =$$

$$= f(x_1|x_2, x_3, \cdots, x_n) f(x_2|x_3, x_4, \cdots, x_n) f(x_3, x_4, \cdots, x_n)$$

and if we continue for $x_3, x_4, \cdots, x_n$ we obtain the chain rule [34].

2.2 Bayes rule

The standard Bayes rule performs the reversal of causality. Normally, we have a cause and an effect, but with this rule, we estimate the cause when we know the effect. From our point of view, the cause is the parameters and the effect is the output data. Using Bayes rule, we reverse this statement and use the data to determine the parameters that induced the data. Thus, in this thesis, we use Bayes rule to estimate the parameters of models that are described by the probability density function

$$f\left(y_t|x_t,\Theta\right),$$

where y_t is the target variable at time t, x_t is the regression vector containing the explanatory variables at time t on which the target variable depends, and Θ is a collection of model parameters. The prior and posterior probability density functions $f\left(\Theta|d\left(t-1\right)\right) \rightarrow f\left(\Theta|d\left(t\right)\right)$ are also essential [22], where $d\left(t\right)=\left\{d_0,d_1,\cdots,d_t\right\}$ is the sequence of all data, with d_0 denoting prior information and $d_t=\left\{y_t,x_t\right\}$ describing pairs of target and explanatory variables at time t. The formula of the Bayes rule has the form

$$\underbrace{f\left(\Theta|d\left(t\right)\right)}_{posterior} \propto \underbrace{f\left(y_t|x_t,\Theta\right)}_{model}\underbrace{f\left(\Theta|d\left(t-1\right)\right)}_{prior}. \tag{2.1}$$

This formula includes a normalization constant corresponding to the predictive probability density function of the data $f\left(d_t|d\left(t-1\right)\right)$, which describes the conditional probability of the current data given the knowledge of previously measured data. This term is omitted and replaced by the proportional sign $\propto$.

Naive Bayes rule

For data prediction, we need the prediction distribution $f\left(y|x\right)=f\left(y|x_1,x_2,\cdots,x_n\right)$, where y is the variable to be predicted and x_i are explanatory variables that influence the variable y. If the variables x_i are independent, we can use the Naive Bayes principle as follows [35]

$$f\left(y|x_1,x_2,\cdots,x_n\right) \propto f\left(x_1,x_2,\cdots,x_n|y\right)f\left(y\right) = f\left(y\right)\prod_{i=1}^{n} f\left(x_i|y\right).$$

The dimension of the used distributions is significantly reduced by this principle.

2.3 Bayesian model

The model in Bayesian statistics has a form of conditional probability density function $f(y_t|x_t, \Theta)$. The model describes the output as a random variable in the form of a conditional probability (density) function. The model generally contains two indeterminate elements - noise and unknown parameters. Bayesian statistics treats these two elements as random variables described by their probability density functions. We use a prior knowledge and measured data for estimation [22].

The model can be static or dynamic, but for the analysis of discrete questionnaire data, a dynamic model does not make sense because we are not interested in the evolution of variables (we do not observe the dependence of a person's answers on previous answers). Therefore, only the static model is used in this thesis, which takes two different forms that are important and need to be defined:

- descriptive - description of the space of x

$$f(x) = f(x_1, x_2, \cdots, x_n),$$

- explanatory - connection between x and y

$$f(y|x) = f(y|x_1, x_2, \cdots, x_n).$$

Formulas are written for known model parameters. Models in practice will still contain unknown parameters in the state that need to be estimated.

2.4 Discrete models

Due to the input data, only a discrete model is used in this thesis. Discrete model can be used if all the variables entering the model are discrete. Then there is a finite number of value configurations of the data vector $[y_t, x_t]$.

Categorical model

The categorical model allows us to assign a probability to each configuration separately, and the model is

$$f(y = k|x = l, \Theta) = \Theta_{k|l}, \tag{2.2}$$

where $k|l$ is the multiindex with k denoting the value of the target variable y and l is the vector of the values of the explanatory variable $x = [x_1, x_2, \cdots, x_n]$, and Θ is the model parameter.

The discrete model can also be used for a discretized continuous variable.

There are several distributions to describe the discrete model, but the categorical distribution is the most commonly used. The categorical model assigns a probability to each combination of values of the variables involved. If there are too many combinations, the model has a high dimension and a large number of parameters. There is always an effort to avoid this. Therefore, in our case, we choose another discrete distribution with less parameters.Based on the previous research and the study [16], we decided to use the binomial distribution due to its best estimation results.

Binomial model

The probability function of the binomial distribution is

$$f\left(x|p\right) = \binom{N}{x} p^x \left(1-p\right)^{N-x},$$

where N is the number of Bernoulli trials in the binomial experiment, x is the total number of positive trails, and p is the probability of a positive trail in each performed Bernoulli trial [36].

2.5 Estimation and classification

Model parameters are generally unknown and must be estimated from the measured data. Recursive parameter estimation is based on recomputing distributions describing unknown parameters according to the Bayes rule. The goal of modeling is often to achieve an optimal prediction of the output, which requires estimation of unknown model parameters.

The Bayes rule (2.1) is based on two types of distributions. The first is a distribution of the output, called the system model

$$f\left(y_t|x_t,\Theta\right)$$

and the second is a distribution of the parameter (prior and posterior)

$$f\left(\Theta|d\left(t-1\right)\right),\ f\left(\Theta|d\left(t\right)\right),$$

where $f\left(\Theta|d\left(t-1\right)\right)$ is the description of the parameter Θ based on the old data with respect to the current time t and $f\left(\Theta|d\left(t\right)\right)$ uses information from all available data, including those measured at time t. The estimation follows the Bayes rule. It explains how to obtain a new (posterior) parameter

distribution $f(\Theta|d(\tau))$ from the old (prior) one $f(\Theta|d(\tau-1))$ using the system model $f(y_\tau|x_\tau,\Theta)$ for $\tau \in \{1,2,\cdots,t\}$

$$f(\Theta|d(\tau)) \propto f(y_\tau|x_\tau,\Theta)\, f(\Theta|d(\tau-1)).$$

It shows the evolution of the parameter probability density function over time as new measured data are supplied

$$f(\Theta|d(0)) \underbrace{\rightarrow}_{d_1=\{y_1,x_1\}} f(\Theta|d(1)) \underbrace{\rightarrow}_{d_2=\{y_2,x_2\}} \cdots \underbrace{\rightarrow}_{d_t=\{y_t,x_t\}} f(\Theta|d(t))$$

with the initial (prior) probability density function $f(\Theta|d(0))$ constructed from prior data or specified by an expert and $d_t = \{y_t, x_t\}$ describing pairs of target and explanatory variables at time t [37].

There are two imaginary levels in creating algorithms for estimation and classification. The first level is based on probability densities and this way is only general. The second level makes these densities more specific and takes into account the special distribution and its statistics.

The estimation algorithms for single models are presented in the following chapter.

Chapter 3

Single models

In this chapter, the single models are described in detail with explanations based on simple examples. The examples are demonstrated in Scilab (www.scilab.org), where parameters and prior information can be changed in the codes to show the principle of these models. The basic discrete categorical model is introduced first, followed by the binomial model.

3.1 Categorical model

The discrete categorical model can describe target and explanatory variables

$$f(y,x) = f(x)\, f(y|x)\,.$$

The decomposition of this model produces the descriptive model $f(x)$ and the explanatory model $f(y|x)$. The descriptive model $f(x)$ for two-valued variables $x = [x_1, x_2]$ can be described by the following Table 3.1.

Table 3.1: Descriptive part of the categorical model

x_1	1	1	2	2
x_2	1	2	1	2
Θ	Θ_{11}	Θ_{12}	Θ_{21}	Θ_{22}

The explanatory model $f(y|x)$ for two-valued variables $x = [x_1, x_2]$ and y has parameters $\Theta_{y|x}$. The model can be given in the form of a Table 3.2.

Table 3.2: Explanatory part of the categorical model

x_1	1	1	2	2
x_2	1	2	1	2
$y=1$	$\Theta_{1\|11}$	$\Theta_{1\|12}$	$\Theta_{1\|21}$	$\Theta_{1\|22}$
$y=2$	$\Theta_{2\|11}$	$\Theta_{2\|12}$	$\Theta_{2\|21}$	$\Theta_{2\|22}$

The entries of the table denote all configurations of the data vector, each of them is assigned its own parameter $\Theta_{i|jk}$ with multiindex $i|jk$, where the target variables y with values $i \in \{1,2\}$ are separated from the explanatory variable $x = [x_1, x_2]_t$ where $j \in \{1,2\}$ and $k \in \{1,2\}$ [22].

The practical use of the categorical model can be demonstrated by an example involving a T-junction.

The experiment with the T-junction is described by a binary categorical variable x with the values 1 (turn to the left) and 2 (turn to the right). If we generally admit that only left or right turns are possible at a junction, then

$$P(y=1) = p_1 \text{ and } P(y=2) = p_2$$

where, indeed, it holds: $p_1 \geq 0$, $p_2 \geq 0$ and $p_1 + p_2 = 1$.

The categorical model $f(y) = p_y$ with $y \in \{1,2\}$ is given in Table 3.3.

Table 3.3: Categorical model in the form of a table

y	1	2
$f(y)$	p_1	p_2

This experiment is well known, and according to the statistical definition of probability, we can experimentally determine the estimates of the parameters p_1 and p_2 : After performing a sufficient the number M of experiments, we count number M_1 of results with $y = 1$ and M_2 as the number of results with $y = 2$. Then it holds

$$p_1 = \frac{M_1}{M_1 + M_2}, \; p_2 = \frac{M_2}{M_1 + M_2}$$

or

$$[p_1, p_2] = \frac{[M_1, M_2]}{M} \tag{3.1}$$

which is a normalization to a sum equal to one [38].

The estimation can also be performed recursively. To do this, we define initial statistics

$$M_{1;0} = 0 \text{ and } M_{2;0} = 0\,.$$

During the online measurement of data for $t \in \{1, 2, \cdots, T\}$, the statistics are updated

$$M_{1;t} = M_{1;t-1} + 1, \text{ if } y_t = 1\,,$$

$$M_{2;t} = M_{2;t-1} + 1, \text{ if } y_t = 2\,.$$

The current update of the parameter estimates (or for $t = T$ it is the last update) is performed by normalization (3.1).

The T-junction example with simulated data is described in detail in the experiments in Subsection 6.1.1.

Note: *In experiments, there are sometimes situations where it is appropriate to introduce prior information, e.g., 20% of the cars turned left and 80% turned right. This corresponds to the probabilities* $p = [0,2\ \ 0,8]$. *If the same information comes from 10 data records, it means that 2 cars turn left and 8 turn right. Then the value of the variable* $ka = 10$ *(used in Scilab) and this variable determines the strength of the prior information (number of a prior steps). The probability* p *is then multiplied by the variable ka to produce the initial summary statistic* $S = p * ka$. *Conversely, if we divide the prior knowledge (2 cars turned left and 8 turned right) by the variable ka, we get the value of the point estimates* p *that we want. This principle can be implemented similarly for other models.*

3.1.1 Descriptive categorical model

The general form of the model is

$$f\,(y) = p_y$$

for y defined integers. This model is very general because it assigns a probability to each situation that occurs. This has advantages (mainly accuracy), but also disadvantages (high dimensionality and overparameterization).

This is a similar example to the T-junction experiment, except that y takes on more values, e.g., 5 instead of 2. The probability density for this descriptive categorical model with 5 values is shown in

Table 3.4, which has the same form as Table 3.1 (which is for 2 variables).

Table 3.4: Categorical model with five values

y	1	2	3	4	5
$f(y)$	$0,10$	$0,27$	$0,18$	$0,41$	$0,04$

It should be noted that the values of p_y are probabilities, i.e. they are non-negative, and their sum is equal to 1. The number of values y in Table 3.4 corresponds to the the 5 arm roundabout example in Subsection 6.1.1.

3.1.2 Explanatory categorical model

The general form of the explanatory categorical model is

$$f(y|x) = p_{y|x}\,.$$

An example is a model with the following parameters $y \in \{1,2\}$ and $x = [x_1, x_2]$; $x_1 \in \{1,2,3\}$ and $x_2 \in \{1,2\}$. The model parameters can be seen in Table 3.5.

Table 3.5: Example of explanatory categorical model

x_1	1	1	2	2	3	3
x_2	1	2	1	2	1	2
$y=1$	$p_{1\|11}$	$p_{1\|12}$	$p_{1\|21}$	$p_{1\|22}$	$p_{1\|31}$	$p_{1\|32}$
$y=2$	$p_{2\|11}$	$p_{2\|12}$	$p_{2\|21}$	$p_{2\|22}$	$p_{2\|31}$	$p_{2\|32}$

The descriptive model describes how often particular combinations of variables occur, while the explanatory model determines what the relationship between x and y is. Given a certain combination of explanatory variables x, we search for the probability of the target variable y. The explanatory categorical model is therefore very important because it allows us to observe behavior. The probabilities $p_{y|x}$ of this model may look like this:

$$p_{1|x} = [0,34\ \ 0,17\ \ 0,65\ \ 0,49\ \ 0,02\ \ 0,22]\,,$$

$$p_{2|x} = [0,66\ \ 0,83\ \ 0,35\ \ 0,51\ \ 0,98\ \ 0,78]\,.$$

In contrast to the descriptive model, the probabilities in the columns are normalized, and the sum of the values of y for a given a combination of x must be equal to one.

3.2 Binomial model

The binomial model describes an experiment consisting of a serial of N independent Bernoulli trials where the outcome is a number of successes. It has the probability function

$$f\left(x|p\right) = \binom{N}{x} p^x \left(1-p\right)^{N-x}.$$

Here, we assume that the number of experiments is known and the only model parameter is p. The mean value is $E\left[x\right] = Np$, the variance is $D\left[x\right] = p\left(1-p\right)$.

For fixed N, the binomial distribution belongs to the exponential family. The statistic for estimating the parameter p is S - sum of realizations of x and κ - number of these realizations. Its recursive update is (index t denotes discrete time of data measurement)

$$S_t = S_{t-1} + x_t$$
$$\kappa_t = \kappa_{t-1} + 1$$

with initial values S_0 and κ_0. In these initial values, it is possible to express a prior knowledge of the parameter p.

The point estimate of parameter p, estimated from data up to time t, is

$$\hat{p}_t = \frac{S_t}{\kappa_t N}.$$

The main advantage of the binomial distribution is a fixed N and a low dimension of the model. For a known number of trails, this distribution is defined by only one scalar parameter p, which ensures the flexibility of the shape of the probability function. It is also important to note that the binomial distribution does not exist for a vector variable and therefore the conditional probability function cannot be written. Figure 3.1 shows the flexibility of the binomial distribution for different values of the parameter p [39, 40].

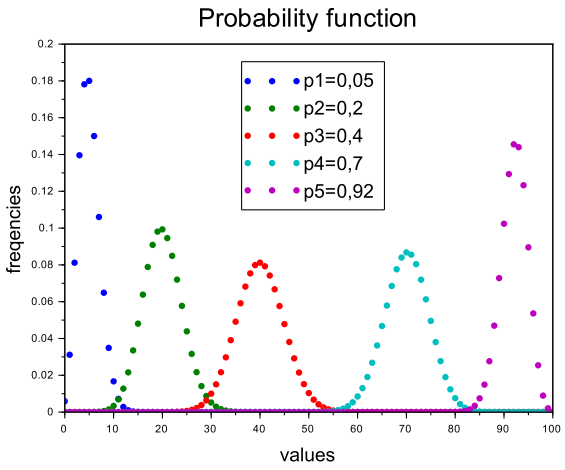

Figure 3.1: Histogram of the probability function of a binomial distribution with different parameters

Chapter 4

Mixtures

Modeling of single models with uncertainty from the exponential family of distributions in Bayesian statistics is quite simple, because the model is in the form of a conditional probability function that defines only one mode. However, real applications involve data with a multimodal character, so it is necessary to use mixtures of distributions that model all modes of the data file. Therefore, mixtures are generally used to describe these multimodal systems that switch between a finite number of different working modes. A mixture is composed of a set of ordinary models (here binomial) and a pointer model [22]. The histogram in Figure 4.1 represents a mixture of a binomial distribution with two components.

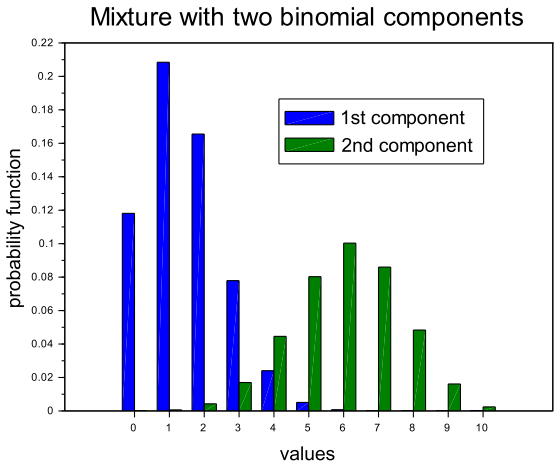

Figure 4.1: Histogram of a mixture of a binomial distributions

In Figure 4.1, the first binomial component has the parameter $p_1 = 0,15$, the second has the

parameter $p_2 = 0,6$. In addition, a stationary probability weights (based only on old data) are assigned to each component by using the parameter α. The parameter is set to $\alpha_1 = 0,6$ for the first component and $\alpha_2 = 0,4$ for the second component. Each component describes one working mode of the system, and the pointer indicates the active component at any given time. The estimation of the pointer can be used for data classification. However, we are interested in estimating the mixture parameters in the space of explanatory variables x to form the clusters that we will work with in later phases of the thesis.

In general, a mixture of binomial components is defined in the form of probability functions

$$f_j\left(x|p_j\right) = \binom{N}{x} p_j^x \left(1-p_j\right)^{N-x}, \;\; x \in \{0,1,\cdots,N\}$$

for $j \in \{1,2,\cdots,m\}$, where m is the number of components, N is the maximum of x, which is fixed and known (so it does not appear in the model conditions).

The pointer model has the following form

$$f\left(c=j|\alpha\right) = \alpha_j\,,$$

where c is the aforementioned pointer indicating the active component and α_j for $j \in \{1,2,\cdots,m\}$ is a stationary probability weight.

The model of the unknown variables is

$$f\left(x, c=j|p_j,\alpha\right) = f\left(x|c=j,p_j\right) f\left(c=j|\alpha\right) = f_j\left(x|p_j\right)\alpha_j$$

and it is decomposed into component and pointer models. Some objects in conditions disappear due to the assumed independence.

Consequently, the mixture model has the form

$$f\left(x|p,\alpha\right) = \sum_{i=j}^{m} \alpha_j f_j\left(x|p_j\right)$$

for $j \in \{1,2,\cdots,m\}$, where m is the number of components.

Mixtures can be modeled and estimated in several ways, but the thesis focuses on a Bayesian modeling approach, specifically quasi-Bayesian [41]. Mixture estimation consists of classifying the measured data records with respect to the individual components and then the updating of the statistics with weights

$$w_j = f_j\left(x_t|\hat{p}_j\right)$$

for $j \in \{1, 2, \cdots, m\}$.

Note: *If there was no weighted classification, the data would be dragged around (the first component would pull the record from the first component, as well as the second and third components), and all the data would cluster in the middle and nothing would come of it. Thus, estimation without classification of the active component would not work.*

If the switching of the components, i.e. the current values of the pointer, is known, the situation is easy. We would simply use the data records to update the active component and leave the rest of them unchanged. In practice, we do not know, which working regime generated the data record. So we have to estimate it. Uncertainty causes, that there is no one hundred percent active component, but we assign probabilities (weights) of activity to each component and with these probabilities (in the sense of a part from the whole measured data) we use the data element for updating all components. The weights are equal to proximities normalized so that their sum is equal to one and these weights are used in the update of the component statistics. The proximity is defined as a value of the component with inserted the actual data record and the current point estimate of its parameter [33, 42, 43].

In this thesis, we are interested in the estimation of the mixture with the mentioned binomial components. The estimation of these components is based on the standard estimation of the single (binomial) model (described in Section 3.2) with weighted data. The full derivation of the mixture estimate is given in Appendix A, but the abbreviated estimation algorithm for binomial components is as follows:

for $t \in \{1, 2, \cdots, T\}$ do:

- measure x_t,
- substitute x_t into all components with current estimates of parameters $\hat{p}_j$ for $j \in \{1, 2, \cdots, m\}$

$$q_j = f_j\left(x_t|\hat{p}_j\right)\hat{\alpha}_{t-1} \doteq f_j\left(x_t|\hat{p}_j\right) ,$$

  where $\hat{\alpha}$ has practically no influence and it can be ignored,

- the component values q_j are normalized to the unit sum and denoted by $w_t = [w_1, w_2, \cdots, w_m]_t$

$$w_{j;t} = \frac{q_j}{\sum_{k=1}^{m} q_k} ,$$

- with these weights we recalculate the statistics of all components for $j \in \{1, 2, \cdots, m\}$

$$S_{j;t} = S_{j;t-1} + w_{j;t} x_t \,,$$
$$\kappa_{j;t} = \kappa_{j;t-1} + w_{j;t} \,,$$

- finally, we compute point estimates of the parameters of all components for $j \in \{1, 2, \cdots, m\}$

$$\hat{p}_{j;t} = \frac{S_{j;t}}{\kappa_{j;t} N} \,.$$

An experiment for estimation of the mixture $f(x_1, x_2|c)$ with binomial x is shown in Subsection 6.1.3.

4.1 Initialization of mixture estimation

Mixture estimation consists of estimating parameters of individual components and the pointer model. It is assumed that the measured data come sequentially from different working modes of the system and thus belong to different components of the mixture model. The farther the data record is from the center of a component, the smaller the proximity value will be; which means the smaller is the probability that the data record belongs to that component. The weights of the components are normalized proximities. If the data are far from the initial components, the weights will be virtually zero and no estimation will occur. Therefore, there is a need to:

1. set the initial distributions of the components to be in the region where the data occur,
2. prevent any component from moving away or overlapping with the components at the beginning of the estimate.

This is done by initializing the mixture estimation [21, 44, 45] - i.e., positioning the initial components appropriately and partially fixing them (forcing them to change more slowly) at the beginning of the estimation. In doing so, we assume that we have a prior sample of data (i.e., data obtained in the past that are available prior to the start of the estimation from continuously measured data).

The general principles of initialization are as follows:

1. Find a region where the measured data occur. For example, find out the minima and maxima for each variable, or better, look at their histograms.

2. Set the initial values of the parameter estimates as well as possible (using prior data and expert information).

3. Hold the prior estimates of the component centers at the beginning of the estimation so that they do not run too far or overlap.

4. Keep small covariances fixed for the components (if we care about the shape of the clusters, we start estimating them later, when the component centers are more or less correctly determined).

5. Run the estimation repeatedly on the same data sample. In this case, it is necessary to specify the previous estimates instead of the prior parameters, and to reset the statistics to their prior values (so that they are not tight).

6. Determine artificial data vectors and use them for initialization. Artificial data vectors are not measured, but determined by an expert.

7. Perform expert classification on several prior or artificially generated data and use them for initialization.

Each point is shown theoretically under the same number in Appendix B.

An example with simulated data to illustrate the initialization properties is described in the experiments in Subsection 6.1.1.

4.2 Mixture applied to real problem

Some of the terms may be difficult to understand, so here is an example from practice that explains the important terms of mixtures and their estimation.

We monitor the level of traffic in the city. During the day, there are two different periods, namely rush hour and off-peak hour, where the situation changes significantly. The rush hour is a period of time when the intensity of the traffic flow is greater than the average of the observed intensities during the entire monitoring interval. The off-peak hour is the opposite of the rush hour, between which the intensity of traffic flow is lower than the average of the observed period [46]. The morning and afternoon rush hours usually alternate with the off-peak hours during the day.

The traffic level is divided into five groups according to the average speed of vehicles, the composition of the traffic flow, the density and intensity of the traffic, etc. The groups represent continuous traffic (1), thickening traffic (2), heavy traffic (3), convoy formation (4) and traffic collapse (5) [47].

It is assumed that during off-peak hours the city traffic level is 1. When the rush hour comes, the traffic starts to thicken and usually occurs in levels 3-5. It may seem, that the switching of components (working points: rush hour and off-peak hours) is known. However, an exceptional off-peak evening may behave like a rush hour if there is a football match in the monitored area. Another example is an off-peak car accident that causes rush hour traffic. This means that we do not know the actual level of traffic, so we have to estimate it based on the circumstances. In this example, the circumstances are discretized speed (maximum permitted, reduced, low, and almost zero speed) and discretized intensity (large, medium, and small intensity). These circumstances are variables from which we estimate the level of traffic. If the circumstances change a little within a working regime, we still stay in the same data cluster (described by one component). The predicted period depends only on the level of traffic, not on the specific value of the circumstances.

Assignment of terms:

- the level of traffic is modeled (clustered) variable,
- circumstances (discretized speed of vehicles and intensity of traffic) are discrete explanatory variables,
- the data groups are clusters,
- the description of clusters is done by components,
- the switching of the regimes is indicated by the values of the pointer (which are estimated = classification).

Experts will use speeds and intensities in specific areas of the city to determine the overall level of traffic in Prague. Based on this, we will create a model that will be able to predict the level of traffic depending on the explanatory variables. The result of this model can be seen as clustering. During the prediction, we find out to which cluster the relevant circumstances belong and then we determine the level of traffic. It is also important that the circumstances within one working mode can be assumed to have a noise character - the changes are caused only by random events, such as a slow vehicle on some street or irregular driving caused by less experienced driver. It does not cause significant and synchronized changes of the explanatory variables. In clusters, we assume the independence of measured variables within individual circumstances. This means that random minor changes do not affect the level of traffic. A switch to another cluster is caused by a significant change. Only this switch gives information about the change in traffic level.

For example, we consider a driving school car to be a discrete noise in a cluster. This car is slightly slower than other cars, but it will not cause a change of cluster, and the traffic level will remain the same. Only a significant change, such as a traffic accident, will cause a change to another cluster, even if a cluster with traffic level 1 is assumed.

Chapter 5

Marginal mixtures

This chapter deals with the core of the thesis, which aims to analyze and model extensive files of discrete data based on the newly developed method called estimation of marginal mixtures, or marginal mixtures for short. It is now necessary to state the basic formulas on which the estimation is based.

The data set is denoted by

$$\{y_t, x_t\}_{t=1}^{T} ,$$

where T is the number of measured data, y_t is a discrete scalar target variable, and x_t is a vector of n explanatory variables (circumstances)

$$x = [x_1, x_2, \cdots, x_n] ,$$

where $x_{i;t}$ are discrete or discretized continuous variables.

The mutual description of these variables is

$$f(y, x) = f(x) f(y|x) .$$

This model with joint probability function can be divided into the circumstances model $f(x)$ and the prediction model $f(y|x)$. After the learning phase (when both x and y are measured), only the vector of variables x is known, and we are interested in predicting the discrete target variable $\hat{y}$ based on the measured values of $\hat{x}$. The predicted value $\hat{y}$ classifies the actual data record into the corresponding component. This is achieved by estimating marginal mixtures, which are divided into three parts:

- creating clusters in data space x,
- construction of local models for classification,
- classification using naive Bayes.

This method is mainly based on dimension reduction, and the individual parts are described in detail in the following sections.

5.1 Creating clusters in data space x

The circumstance model $f(x)$ aims at clustering the data space x under the assumption of independence of the explanatory variables in the individual clusters (the experiment in Subsection 6.1.3 showed that the independence assumption may not be fully satisfied and yet the results are very good, so this procedure can be used in all cases).

The description of modeling with independent (marginal) mixtures is as follows. We have n explanatory variables $x = [x_1, x_2, \cdots, x_n]$. We assume that these variables come from a strongly multimodal system. By this we understand that the system has several working points (modes) and works relatively easily in each of them, but when all modes are mixed together, everything is lost. Therefore, we want to use clustering to find local models where we expect simpler links between the explanatory and target variables. For such a system, it holds that its variables are independent under the condition of a fixed mode.

The independence of variables in clusters reduces the dimension by reducing the huge table describing the data set (Section 3.1) to vectors. Therefore, only one variable x_i, $i \in \{1, 2, \cdots, n\}$ is always modeled separately.

The corresponding local model is

$$f_j(x_i|p_{ij}),$$

where j is the component and i is the variable.

The creation of clusters in the data space x is based on Bayesian mixture estimation, which is described in detail in Chapter 4. Specifically, it is a mixture of binomial distributions. This estimation requires initialization, and the proposed approach with independence has a great advantage in this respect. Each variable and each of its components has its own scalar model, which can be easily initialized using a histogram. The probability functions of individual variables are then one-dimensional

normalized histograms of the frequencies of the values for each variable separately. In the histograms, we can easily see the individual modes and their corresponding tops, from which we can determine the parameters p_{ij} for each variable and component of the binomial mixture. This procedure is performed separately for each variable, so that the entire data space x is divided into clusters, where each variable can have a different number of clusters. The principle of cluster creation is shown in Figure 5.1.

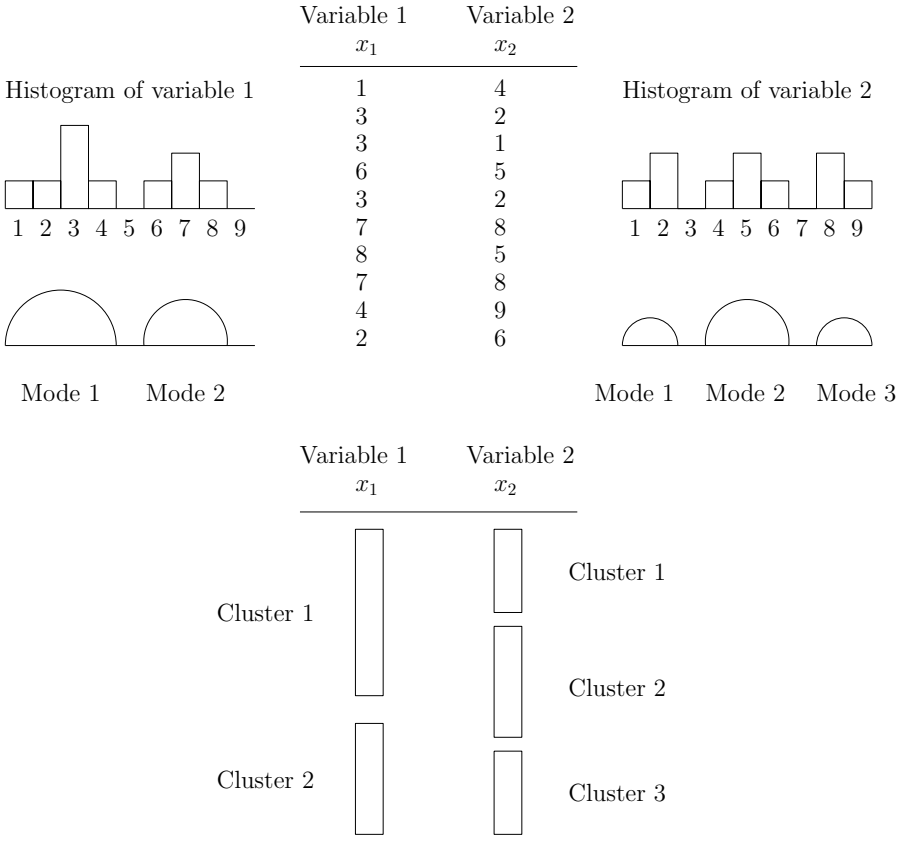

Figure 5.1: Clustering of data space x

In practice, the files of discrete data are very extensive, but only a small sample of these data is used in Figure 5.1 to describe the clustering principle of the data space x. Therefore, the example contains only two explanatory variables x_1 and x_2, and each of these variables has only ten records, as shown at the top of this figure. The first histogram to the left of the data is now created from the values of the first variable x_1. The second histogram to the right of the data is generated from the values of the second variable x_2. Both of these variables are generated from a mixture of binomial distributions, which are defined by a vector of binomial parameters p_{ij}. The parameter vector is

obtained by mixture estimation (see Chapter 4). With these parameters it is now possible to read directly from the histogram where the modes are located and where their tops are (the marking of the modes under the histograms is only for illustration). This information read from the data can be used to locate the components. The histogram of the first variable on the left side of the figure shows that this variable x_1 has two modes (i.e., it is a mixture with two binomial components). The second variable x_2, whose histogram is shown on the right, has even three modes (i.e. it is a mixture with three binomial components). The individual components (modes) symbolically describe the clusters indicated at the bottom of the figure, i.e. the first cluster is described by the first component in the first variable x_1 with the values $1, 2, 3, 4$ and the second cluster is described by the second component of this variable with the values $6, 7, 8$. The same applies to the second variable x_2, where three clusters are symbolically denoted in the lower part of the figure. The examined data space x is now divided into initial clusters and also described by an initial binomial mixture for individual variables.

5.2 Construction of local models for classification

The classification we aim at is based on the model $f(y|x)$. To take advantage of the independence of x_i, we use the Naive Bayes principle, which is based on the product of the models $f(x_i|y)$. Therefore, the basis for classification is now established by constructing all local categorical models $f_j(x_i|y)$ in each cluster $c_i = j$ for all variables x_i. In the first step, we take the first cluster $c_1 = 1$ of the first variable x_1 and see which values of x_1 belong to this cluster. Then we select the corresponding values of the discrete target scalar variable y that belong to the selected values of x_1. These data are used to construct the model $f_1(x_1|y)$, which has the form of a normalized frequency table similar to Table 3.5. This procedure is performed for all clusters $c_i = j \in \{1, 2, \cdots, m_i\}$ in all variables x_i, $i \in \{1, 2, \cdots, n\}$. For the subsequent classification, it is also necessary to create the model $f(y)$, which is formed simply by a normalized histogram of all y values.

The described principle of constructing local categorical models is shown graphically in Figure 5.2, which follows the previous Figure 5.1, describing two explanatory variables x_1 and x_2 with ten records. Each record of the variable x_1 is assigned a corresponding cluster. For x_1, there are only two options, $c_1 = 1$ or $c_1 = 2$, which are shown to the left of the data. The first cluster $c_1 = 1$ and its associated values of y are marked in black. Then the model created from this set has the form $f_1(x_1|y)$. The same is done for the rest of the records of the variable x_1 that belong to the second cluster $c_1 = 2$. This second cluster, together with the associated values y, is marked in red, and the resulting model

has the following form $f_2(x_1|y)$. This procedure is also performed for the second variable x_2, which has three clusters $c \in \{1, 2, 3\}$ marked to the right of the records of this variable. Records belonging to the first cluster $c_2 = 1$ and their associated y are marked in black. The model has the form $f_1(x_2|y)$. The second cluster $c_2 = 2$ and its corresponding y are marked in red and the third cluster $c_2 = 3$ and its corresponding y are marked in green. In this way, all local categorical models are determined. On the right side of the figure there are the numbers of records, where each record has the corresponding y. Furthermore, the model $f(y)$ is constructed from the set of all values of y.

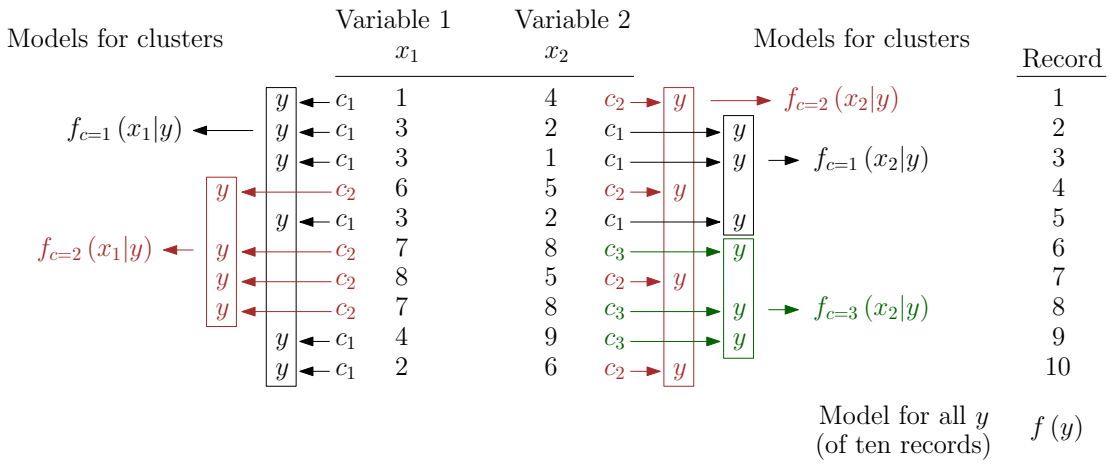

Figure 5.2: Creation of local models

Categorical models $f_j(x_i|y)$ are determined for each cluster in each variable, and the model $f(y)$ is defined for each record in the entire data set. These models do not have a large dimension because they are created for individual variables x_i, which are scalars.

5.3 Classification using naive Bayes

The last part of the marginal mixtures method is based on the use of the naive Bayes principle with the independence assumption on x_i, which is used to achieve the desired prediction model $f(y|x) = f(y|x_1, x_2, \cdots, x_n)$ in order to subsequently estimate the values of the target variable y. The procedure of classification is as follows. In the learning phase, we assume that we have a data sample of both paired variables, i.e. y and the corresponding x_i. In the testing phase, we assume that only the vector $x = \hat{x} = [\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n]$ is measured and the corresponding y is estimated. The ultimate goal is prediction, i.e. estimation of the value of y based on the value of the measured $\hat{x}$.

In the testing phase, the value of $\hat{x}_1$ is inserted into all components of the first variable, creating proximities of x_1 to all components of the first variable. These proximities are created according to the principle described in Appendix A. In the same way, the value of $\hat{x}_2$ is inserted into all the components of the second variable, creating proximities for the second variable x_2. This is done for all the measured variables of the measured sample $\hat{x}$ and their corresponding components. After normalizing the proximities for each variable i, we obtain the weights $w_i = [w_{1;i}, w_{2;i}, \cdots, w_{m_i;i}]$ for $i \in \{1, 2, \cdots, n\}$. With the weights and the local models constructed in the previous section, it is now possible to create the model $f(x_i|y)$ for each variable (not just for clusters within the variable) using the formula

$$f(\hat{x}_i|y) = \sum_{j=1}^{m_i} w_{j;i} f_j(\hat{x}_i|y).$$

The prediction model $f(y|x)$ is a model of the target variable y depending on x. The naive Bayes formula (described in Section 2.2) is used to determine this model

$$f(y|\hat{x}) \propto f(y) \prod_{i=1}^{n} f(\hat{x}_i|y).$$

The measured $\hat{x}$ is substituted into the formula to obtain a prediction of the values y and their probabilities. In the last step, the most probable value $\hat{y}$ is calculated for the measured $\hat{x}$, and this point prediction can be determined as the index of the maximum value of the predictive probability function $f(y|\hat{x})$

$$\hat{y} = \arg\max f(y|\hat{x}).$$

The classification procedure is shown in Figure 5.3. The right part of the figure illustrates again the two already known explanatory variables x_1 and x_2 together with their clusters. The weights for each variable are now determined by inserting the first measured value of $\hat{x}_1$ into the first cluster $c_1 = 1$ of the first variable x_1 in the following form $f_1(\hat{x}_1)$. This gives the proximity of the value $\hat{x}_1$ to the first component $f_1(x_1)$. The same measured value $\hat{x}_1$ is then inserted into the second component $c_1 = 2$ of the first variable x_1 in the form $f_2(\hat{x}_1)$, and it creates the second proximity. If we now normalize these proximities so that their sum is equal to 1, we obtain the weights w_{11} and w_{12} of the components in the first variable x_1. In the same way, the weights of the second component x_2 are determined using the measured value $\hat{x}_2$, which is inserted into all three components to obtain the proximities in the form of $f_j(\hat{x}_2)$ and then the weights w_{12}, w_{22} and w_{32}.

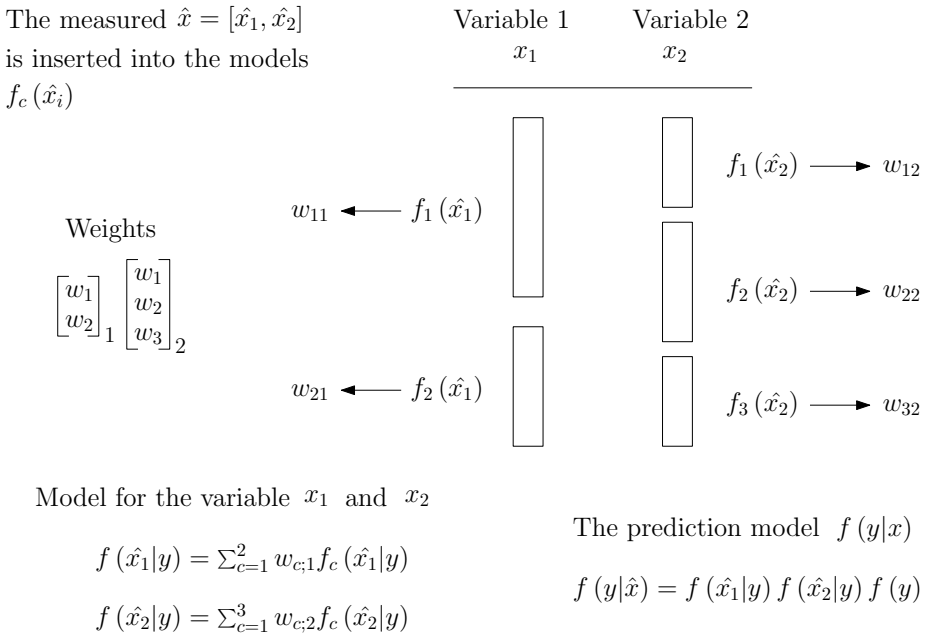

Figure 5.3: Construction of the prediction model

5.4 Algorithm for estimation and classification with marginal mixtures

The individual steps of the developed algorithm of point estimation and classification with marginal mixtures are described in detail and illustrated graphically in this Chapter 5. For clarity, the entire proposed algorithm is summarized below.

For each time instant t perform:

1. clustering data in each independent variable x_i of the data space x:
   (a) independence reduces the dimension and allows each variable x_i, $i \in \{1, 2, \cdots, n_{x_i}\}$ to be modeled separately,
   (b) in the individual variables x_i, the components are searched for using a mixture of binomial distributions,
   (c) a set of components is created in each variable,
   (d) components describe the clusters.

2. preparation of the local models $f_j(x_i|y)$ and $f(y)$:

   (a) $f_j(x_i|y)$ is constructed using data from individual clusters $c_i = j$ and variables x_i,

   (b) probability functions $f(y)$ are constructed for all individual record,

3. testing using prediction:

   (a) $x = \hat{x}$ is measured,

   (b) the weights w_{ij} are determined by substitution $\hat{x}_i$ into the models $f_j(\hat{x}_i)$ of individual components with actually estimated parameters,

   (c) the model $f(x_i|y)$ of individual variables is determined as follows

$$f(\hat{x}_i|y) = \sum_{j=1}^{m_i} w_{j;i} f_j(\hat{x}_i|y),$$

   (d) the prediction model $f(y|x)$ is calculated according to the principle of naive Bayes

$$f(y|\hat{x}) \propto f(y) \prod_{i=1}^{n} f(\hat{x}_i|y),$$

   (e) the point prediction $\hat{y}$ can be determined as the index of the maximum value of $f(y|\hat{x})$

$$\hat{y} = \arg\max f(y|\hat{x}).$$

The developed algorithm was first tested on simulated data and the output in the form of Scilab code is presented in Section 6.2. The algorithm was then applied to real data and the complete code is presented in Appendix F.

Chapter 6

Experiments

Experiments are first performed on simulated data to verify the correctness of the proposed procedure. The final algorithm is then applied to real data. Scilab is used for this purpose, and all the codes below can be easily run by copying and pasting them into this program.

6.1 Basic experiments

Basic experiments are used to demonstrate the theory described in Chapter 3 and 4.

6.1.1 Estimation of the categorical model

For the categorical model, we have already chosen the T-junction experiment in Section 3.1. We will now discuss this experiment in detail in Scilab and show everything that is important about it. The T-junction model in the code below can be divided into two parts. In the first part, on lines //1 to //6, we have the actual right and left turns that represent the generation of the discrete data $y \in \{1, 2\}$, which is described in Appendix C. The second part of the model (lines //7 to //14) focuses on determining the point estimate of the parameter p. To estimate the parameters, we need to define a statistic, and it must be consistent with the prior parameters. So first we define the parameters pE and then we define the strength ka, which we will use to give a prior information to the estimation. Then the statistic has the following form $S = pE * ka$ (line //9) and its construction is based on the theory described in the note in Section 3.1. Next, the statistics in lines //11 and //12 are updated to include the newly measured data. Then an estimate of the parameter pE is constructed in line //13.

The estimate is usually computed either continuously or at the end, in our case continuously.

```
// Model of T-junction + init + estim
// ---------------------------------------
clear, clc, mode(0);

nd=100;                                  // number of data                  //1
p=[.8 .2];                               // parameters of model of y        //2
y=1;                                     // initial condition for y         //3
for t=2:nd                                                                  //4
  y(t)=sum(cumsum(p)<rand(1,1,'u'))+1;
                                         // generation of discrete data     //5
end                                                                         //6

pE=[.5 .5];                              // initial parameters of model     //7
ka=.01;                                  // initial counter statistics      //8
S=pE*ka;                                 // initial summation statistics    //9
for t=2:nd                                                                  //10
  S(y(t))=S(y(t))+1;                     // update of summation statistics  //11
  ka=ka+1;                               // update of counter statistics    //12
  pE(t,:)=S/ka;                          // point estimates                 //13
end                                                                         //14
```

We can change some parameters in the code. First, we show the influence of the initial strength of the information, which is affected by the coefficient ka (initial counter statistics). If we set the value of the coefficient (line //8) to a very small $ka = 0,01$, i.e. we have almost no a prior information, then there are large jumps at the beginning of the estimation, and it can completely miss the mark in more complex cases. Subsequently, we can guess the mean value from the prior information $ka = 10$, where the estimation caught the right direction. On the contrary, we chose a very strong value of the a prior information $ka = 100$ and the estimation did not reach the specified parameters, because the estimation froze and only slowly converged to the desired parameter values. The freezing can be resolved by letting the estimation run, forgetting the statistics, but keeping the estimated initial

parameters and running the original data again. This procedure is repeated until the desired parameter values are reached.

To demonstrate the effect of a prior information, we plotted Figure 6.1 from the Scilab code, showing the generated data and the evolution of the parameter estimation for different initial strengths of information ka. For such a simple demonstration example, the best result was finally obtained for the lowest prior information $ka = 0,01$, where we obtained the parameter estimate $pE = [0,78 \;\; 0,22]$. These values were very close to the initial parameters $p = [0,8 \;\; 0,2]$ set in line //2. Note that we estimate pE as complementary probabilities and the sum is always 1, so the graphs are symmetric. The last figure nicely shows that with strong prior information, the estimation did not actually reach the desired parameters, but only got a little closer to them. The change in the coefficient ka shows that the prior information has a large effect on the parameter estimation.

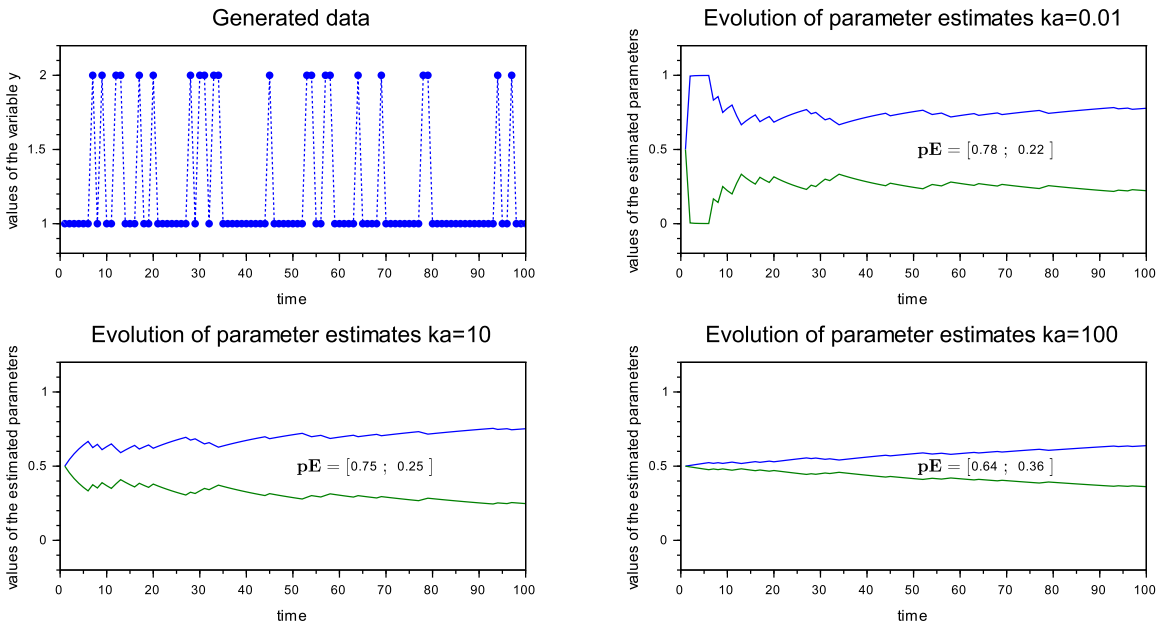

Figure 6.1: The effect of the initial strength of information ka

Next, we show how initialization works using a simple example (for the mixture, it is described in detail in Section 4.1 and Appendix B). The principle of initialization is based on finding and setting the ideal initial parameters of the model pE with which the estimation will start. For the demonstration, 3 options for setting the initial parameters pE (line //7) were chosen. They are shown separately in Figure 6.2. In the Scilab code, the number of data is halved, i.e. $nd = 50$, because we are primarily interested in the beginning of the estimation, and the strength of the prior information is set to $ka = 1$.

First, the ideal option $pE = [0,8\ \ 0,2]$ was chosen, which corresponds to the actual model parameters p that were set in the simulation on line //2 to generate the discrete data y. As a second option, the values of the parameter $pE = [0,5\ \ 0,5]$ were chosen. These values were used in the previous part of the example (influence of prior information). As a last option, the values of $pE = [0,1\ \ 0,9]$ were chosen, which are very far from the set parameters p. The graphs in Figure 6.2 show that the setting of the parameters pE has the greatest effect on the start of the estimation. It is also evident from the graphs that the first variant performs best, when the ideal initial parameters are set and the estimation is stabilized at $pE = [0,78\ \ 0,22]$. The second and third variants have a worse start of estimation and a greater deviation from the true values $p = [0,8\ \ 0,2]$.

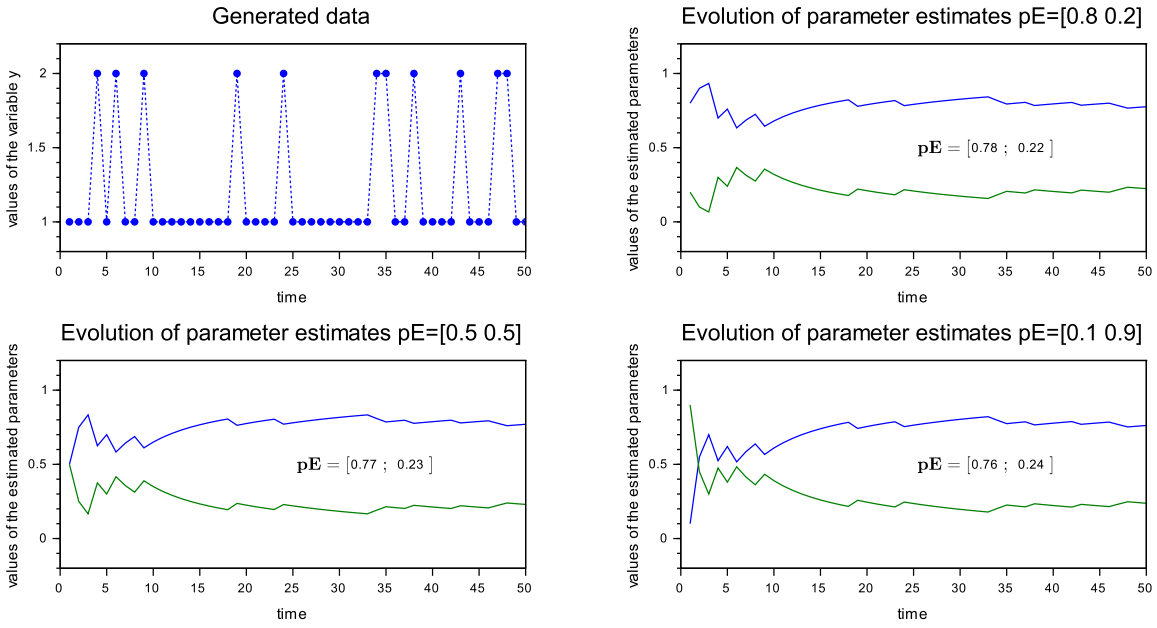

Figure 6.2: The effect of the initial parameters of the model pE

Correctly setting the initial parameters of the model is especially important when estimating mixtures.

As an example of a categorical model with multiple values (namely 5), the example of a 5 arm roundabout was chosen in subsection 3.1.1. We will show and describe this example in more detail using the Scilab code shown below. This is a similar example to the T-junction, but there is a change on the line //2 where we set the model parameters. Here we have not only dealing with two turn options as in the previous example, but now we have 5 of them, so we set 5 values for the parameter p. This allows us to generate the values of the variable $y \in \{1, 2, \cdots, 5\}$. Furthermore, there is an

additional line //7 which creates an auxiliary variable ky and it determines the maximum value of the variable y in case we do not know the simulation and we need to know the number of values to set the initial parameters of the model pE on line //8. The rest of the code remains unchanged.

```
// Model of 5 arm roundabout + init + estim
// ---------------------------------------
clear, clc, mode(0);

nd=100;                                  // number of data                  //1
p=[.2 .1 .3 .2 .2];                      // parameters of model of y        //2
y=1;                                     // initial condition for y         //3
for t=2:nd                                                                  //4
  y(t)=sum(cumsum(p)<rand(1,1,'u'))+1;
                                         // generation of discrete data     //5
end                                                                         //6
ky=max(y);                               // maximum value of variable y     //7

pE=.2*ones(1,ky);                        // initial parameters of model     //8
ka=.01;                                  // initial counter statistics      //9
S=pE*ka;                                 // initial summation statistics    //10
for t=2:nd                                                                  //11
  S(y(t))=S(y(t))+1;                     // update of summation statistics  //12
  ka=ka+1;                               // update of counter statistics    //13
  pE(t,:)=S/ka;                          // point estimates                 //14
end                                                                         //15
```

For clarity, Figure 6.3 shows the generated data y and the evolution of the model parameter estimates with initial model parameters $pE = [0,2\ 0,2\ 0,2\ 0,2\ 0,2]$ and initial counter statistics $ka = 0,01$. It can be seen in the graph that the resulting point estimate values pE slowly converged and stabilized around the true values $p = [0,2\ 0,1\ 0,3\ 0,2\ 0,2]$.

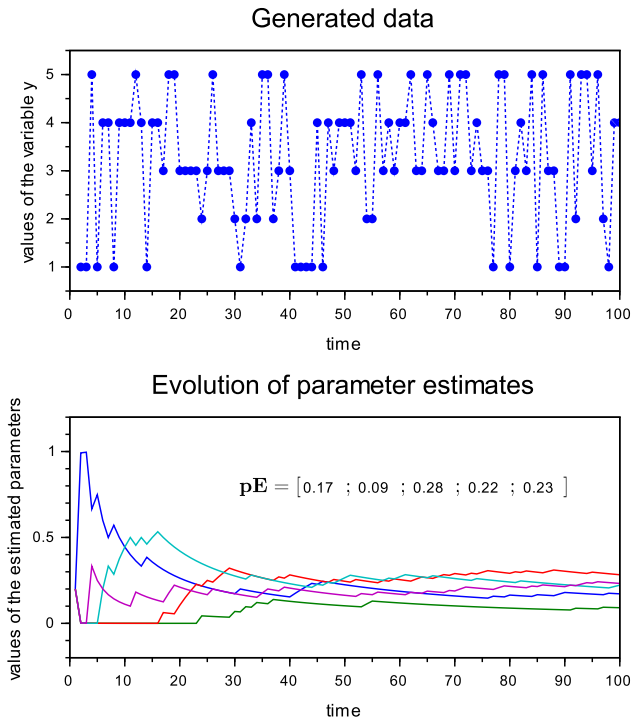

Figure 6.3: Estimation of the categorical model

6.1.2 Estimation of the binomial model

The estimation of the binomial model, which is described in detail in Section 3.2, is based on the same principle as the estimation of the categorical model. This can be seen in the Scilab code below. The first part (lines //1 to //7) is again dedicated to data generation, but here it is the generation of binomial data, where we first use the function on line //4 to determine the probability function of the binomial distribution for the selected parameter $pb = 0,1$. Then, data from the binomial distribution are generated on line //6 according to the principle described in Appendix C. The second part (lines //8 to //15) is devoted to determining the point estimates of the binomial distribution. In this experiment, we have already disregarded any prior knowledge and its strength, hence the value of the initial counter statistic $ka = 0$.

```
// Binomial model + estim
// ---------------------------------------------
clear , clc , mode(0);
```

```
nd=500;                                  // number of data                  //1
pb=.1;                                   // probability of success p        //2
n=3;                                     // number of Bernoulli trials      //3
pc=binomial(pb,n);                       // probability function            //4
for t=1:nd                                                                  //5
  y(t)=sum(cumsum(pc)<rand(1,1,'u'));
                                         // generation of binomial data     //6
end                                                                         //7

S=0;                                     // initial summation statistics    //8
ka=0;                                    // initial counter statistics      //9
for t=1:nd                                                                  //10
  S=S+y(t);                              // update of summation statistics  //11
  ka=ka+1;                               // update of counter statistics    //12
  pE=S/(ka*n);                           // point estimates                 //13
  pt(t)=pE;                              // evolution of point estimate     //14
end                                                                         //15
```

From the Scilab code, we have plotted Figure 6.4, which first shows the generated data from a binomial distribution with parameter $p = 0,1$. The second part of the figure shows the evolution of the parameter pE estimate over time. With the zero knowledge of the prior information, it took a while for the estimate to stabilize close to the true value $p = 0,1$.

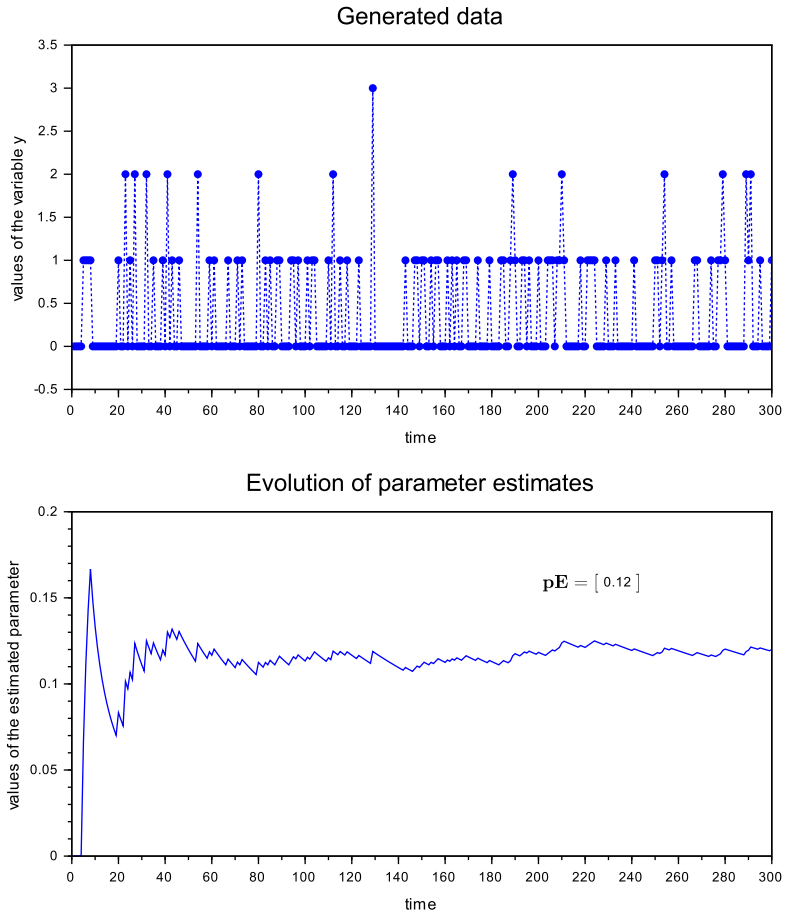

Figure 6.4: Estimation of the binomial model

6.1.3 Estimation of the mixture $f(x_1, x_2|c)$ with binomial x

At the core of the thesis are marginal mixtures, which aim to reduce the high dimensionality of discrete categorical models. The main element contributing to the dimension reduction is the assumption of independence of the explanatory variables x. The presented algorithms are also applied to real data where the independence assumption is not always fully satisfied. Therefore, it is important to know how sensitive the mixture estimation algorithms, performed under the independence assumption, are to violations of this assumption. The following program investigates just this phenomenon. First, we simulate (with the assumption of independence of x) data with independent x. We get the theoretically correct result (which we verify by simulation). We then simulate the data with dependent x in the same way and observe how the point estimates deteriorate. This will determine how much the violation of the independence assumption affects the result.

This example is based on the estimation of a mixture with binomial components, which is described in detail in Chapter 4. The estimation of these components is performed by estimating a standard binomial model with weighted data. For mixture estimation, we observe the same parts as for categorical and binomial model estimation, i.e., simulation including initial parameter settings, initialization, and parameter estimation, but the individual parts are partially different. The differences are mainly in the initialization, where several models have to be initialized. In addition, proximities and weights are also computed in the estimation. There is also a difference in the parameter update, which is weighted and computed for all components. Below is the code from Scilab, where we will show and describe each part in detail. To run the code in the program, you must copy the functions from Appendix D and paste them into the appropriate line.

In the first part (lines //1 to //25), where we set the parameters for the simulation, we also show the impact of violated independence. On the line //2 we set 3 components and then we define the corresponding parameters for them. If we want to simulate data from a binomial distribution, we can compute the probabilities of the individual values for a given parameter p, and then generate the entire distribution as a categorical distribution with binomial parameters. These parameters for the variable x_1 are on the lines //3 to //5. Next, we set the parameters for x_2 to be totally independent (lines //7 to //12), almost independent (lines //13 to //18), and very dependent (lines //19 to //24) with respect to x_1. We create completely independent variables by setting p_2 exactly the same and the change of x_1 values is not reflected. Almost independent variables have p_2 almost the same and their values have little respond to changes in x_1. Conversely, highly dependent variables have p_2 much different, so the values of x_1 have a large effect on x_2. Lines //26 to //31 show the simulation where we generate the pointer values, the variable x_1 and the variable $x_2|x_1$. The variable x_2 is conditional because we want to determine the relationship (dependence or independence) between the variables x_1 and x_2. If we know the parameters of the simulation model, we can easily test the effect of the dependence and independence of the variable x in the binomial distribution by comparing the results of the point estimates of all 3 parameter choices in the simulation.

In the next part of the code (lines //32 to //40) there is the initialization, which is done separately for each component and is crucial for mixtures. If some initial peak of the distribution is far from the data, big problems can occur because the proximity can be zero and the peak of the distribution does not approach the data. Therefore, we need to set the initial peak of the distribution to the real data region by finding the minimum and maximum of the data. Another way to initialize for independent

data is to plot histograms of the data and find clusters (locations with the highest frequency of data) where the initial peaks of the distribution are then placed. However, in the following program, we use the exact simulated parameters (the hardest initialization), because the question is not how well the estimation performs, but how much error is introduced by violating the independence assumptions. If we set the wrong value of the initial parameter, we would not find the effect of dependence, so we need the ideal estimation case for all simulations. Then, the initial probability (lines //36 and //37) and the initial summation statistics (lines //38 and //39) for each component are set using known parameters.

In the last part of the code, on lines //41 to //56, we perform the point parameter estimation, where we first generate the proximity q_1 for variable x_1 and q_2 for variable x_2 in cycle j (lines //43 and //44). The product of these two values $q1 * q2$ on line //45 gives the proximity q for each component under the independence assumption. After normalizing the values of q to the sum of 1, we obtain the weights. These weights are then used to update the counter ka (line //50), to update the summary statistics (lines //51 and //52), and to calculate the point estimates of the parameters (lines //53 and //54).

```
// Estimation of the mixture f(x1,x2|c) with binomial x
// - x1,x2 dependent or independent
// - estimated as for independent x
// ---------------------------------------
clear, clc, mode(0);
// copy the functions from the Appendix D here and run the code

nd=1000;                          // number of data                    //1
pa=[.3 .4 .3];                    // pointer model parameters          //2
C(1).p1=binomial(.1,3);           // parameters of model for x1        //3
C(2).p1=binomial(.3,3);           // parameters of model for x1        //4
C(3).p1=binomial(.8,3);           // parameters of model for x1        //5
select 3              // <== select the dependency of x1 and x2      //6
case 1                // TOTALLY INDEPENDENT                         //7
  C(1).p2=.2*ones(1,4);           // parameters of model for x2        //8
  C(2).p2=.2*ones(1,4);           // parameters of model for x2        //9
```

```
  C(3).p2=.2*ones(1,4);          // parameters of model for x2       //10
  p1=[.1 .2 .5];                 // initial parameters for x1 indep. //11
  p2=[.3 .1 .2];                 // initial parameters for x2 indep. //12
case 2                 // ALMOST INDEPENDENT                         //13
  C(1).p2=[.05 .1 .15 .2];       // parameters of model for x2       //14
  C(2).p2=[.85 .9 .95 .98];      // parameters of model for x2       //15
  C(3).p2=[.4 .45 .5 .55];       // parameters of model for x2       //16
  p1=[.2 .1 .6];                 // initial parameters for x1 indep. //17
  p2=[.7 .1 .5];                 // initial parameters for x2 indep. //18
case 3                 // VERY DEPENDENT                             //19
  C(1).p2=[.1 .3 .6 .9];         // parameters of model for x2       //20
  C(2).p2=[.1 .2 .5 .2];         // parameters of model for x2       //21
  C(3).p2=[.9 .1 .3 .5];         // parameters of model for x2       //22
  p1=[.4 .1 .5];                 // initial parameters for x1 dep.   //23
  p2=[.4 .1 .3];                 // initial parameters for x2 dep.   //24
end                                                                 //25

// SIMULATION
for t=1:nd                                                          //26
  c(t)=sum(cumsum(pa)<randu())+1;          // generation of pointer //27
  x1(t)=sum(cumsum(C(c(t)).p1)<randu())+1; // generation of x1      //28
  pp=binomial(C(c(t)).p2(x1(t)),5);        // parameters of x2|x1   //29
  x2(t)=sum(cumsum(pp)<randu())+1;         // generation of x2      //30
end                                                                 //31

// INITIALIZATION
b=[max(x1) max(x2)];           // maximum of x1 and x2              //32
nc=max(c);                     // number of component               //33
ka=ones(1,nc);                 // initial counter statistics        //34
for j=1:nc                                                          //35
  C(j).pE1=p1(j);              // initial probability of j-th compon. //36
```

```
  C(j).pE2=p2(j);              // initial probability of j-th compon. //37
  C(j).S1=C(j).pE1*ka(j)*(b(1)-1);  // initial summation statistics  //38
  C(j).S2=C(j).pE2*ka(j)*(b(2)-1);  // initial summation statistics  //39
end                                                                  //40

// ESTIMATION
for t=1:nd                                                           //41
  for j=1:nc                                                         //42
    q1=binpdf(x1(t)-1,C(j).pE1,(b(1)-1));  // proximity for x1       //43
    q2=binpdf(x2(t)-1,C(j).pE2,(b(2)-1));  // proximity for x2       //44
    q(j)=q1*q2;                          // proximity of j-th compon. //45
  end                                                                //46
  w=fnorm(q);                            // weights                 //47
  wt(:,t)=w;                             // weights for all records //48
  for j=1:nc                                                         //49
    ka(j)=ka(j)+w(j);                    // update of counter stat. //50
    C(j).S1=C(j).S1+w(j)*(x1(t)-1);      // update of summation stat. //51
    C(j).S2=C(j).S2+w(j)*(x2(t)-1);      // update of summation stat. //52
    C(j).pE1=C(j).S1/(ka(j)*(b(1)-1)); // point estimates             //53
    C(j).pE2=C(j).S2/(ka(j)*(b(2)-1)); // point estimates             //54
  end                                                                //55
end                                                                  //56

// RESULTS
Corrx1x2=cord(x1,x2)                   // discrete correlation        //57
cp=amax(wt,1);                         // argument of maximum weight  //58
h=c2c(c,cp);          // renaming of components in case of rotation   //59
Acc=acc(c,h(cp))      // accuracy of estimated components             //60
```

To verify the relationship (independence versus dependence) between the variables x_1 and x_2, we used the function to determine the correlation coefficient in the code on line //57. The values of this

coefficient for all 3 simulation variants are given in Table 6.1. In the first case (totally independent), the value of the correlation coefficient was close to zero, so there is indeed complete independence between the variables. In the latter case (almost independent), the value of the correlation coefficient is much closer to 0 than to 1, so the variables are almost independent. The last case (very dependent) has a correlation coefficient value higher and close to 1, so the variables are dependent. We then generated the accuracy on line //60, which determines the ratio of correctly estimated components. This step was preceded by determining the actual estimation of the components using the argument of the maximum weights on line //58 and then renaming the components in case of rotation during estimation on line //59. We determined this accuracy again for all 3 variants (totally independent, almost independent and very dependent variables x_1 and x_2) and the results are shown in Table 6.1.

Table 6.1: Correlation coefficient of variables x_1 and x_2 and accuracy of the result

Variables x_1 and x_2	Totally independent	Almost independent	Very dependent
Correlation coefficient [-]	-0,01	0,24	0,61
Accuracy [%]	0,69	0,93	0,59

The accuracy results show that the dependence of the variables has almost no effect on the estimation results, so we can work with this assumption even in the case of data dependence, which we model as independent. This effect is also known from the use of naive Bayes, which has a wide application and the same assumption. Naive Bayes works even when the assumption is not well followed, and this is consistent with our findings.

6.2 Experiments on simulated data

This experiment in Scilab demonstrates the theory described in Chapter 5. Specifically, it involves the estimation of the marginal mixture $f(x_1, x_2|c)$ with binomial variables x_1, x_2.

Algorithm development and testing (for 2 and 2 components)

The algorithm needs to be developed and modified gradually to get everything working as it should. Therefore, only two explanatory variables $x_1 \in \{1, 2, \cdots, 4\}$ and $x_2 \in \{1, 2, \cdots, 6\}$ are used for testing so far, and each of these variables has two components.

The first part of the algorithm deals with pointer estimation, i.e. clustering, based on the previous example of mixture estimation described in Subsection 6.1.3. The investigation verifies (i) the func-

tionality of the algorithm without approximation (easily distinguishable components, purely binomial distribution and really independent variables x) followed by a slight overlap of components (ii) the effect of slightly dependent variables x (iii) the effect of violated binomial distribution.

(i) When the parameters p of the variables x, which determine the location of the modes of the binomial mixture, are set to edge values, the estimation result is excellent. However, if the components are moved closer together by adjusting the values of the parameters p, the point estimation results of the pointer will deteriorate slightly, depending on how much the components have been moved closer together and how much they overlap. Figure 6.5 on the left shows the variable x_1, which represents a mixture with two binomial components at the edges, using the set parameters $p_1 = [0,1 \;\; 0,9]$. This favorable variant has a clustering with $accuracy = 0,97$. Then the value of the parameter $p_1 = [0,3 \;\; 0,7]$ was set so that the components are closer together and therefore overlap more, as shown on the right in Figure 6.5. Although the conditions for estimating the pointer have deteriorated, the *accuracy* is still $0,79$.

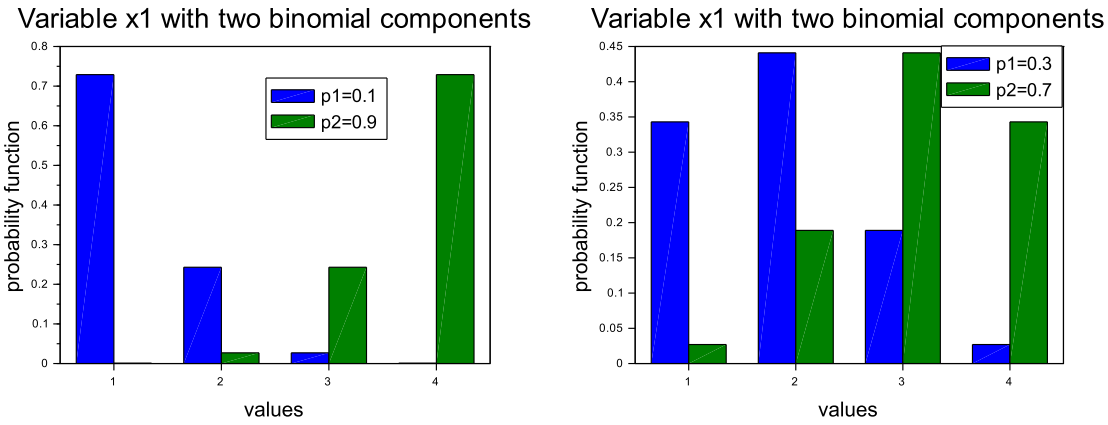

Figure 6.5: Probability functions of the variable x_1 with two binomial components

(ii) The influence of the independence or dependence between the variables x has already been discussed in Subsection 6.1.3. However, it is necessary to examine the impact this has on the clustering accuracy, which in turn affects the classification of the target variable y. The relationship (independence or dependence) between the variables x_1 and x_2 is determined by the parameter p_2 for the variable x_2, which is then influenced by the value of x_1 from the binomial mixture. The result is a binomial model simulated as a categorical model. The independence between the variables is then determined only by this parameter p_2, which must be set to the same values,

e.g. $p_2(1) = [0,99\ \ 0,99\ \ 0,99\ \ 0,99]$ for the first cluster and $p_2(2) = [0,01\ \ 0,01\ \ 0,01\ \ 0,01]$ for the second cluster, so that the change in the values of x_1 does not affect the generation of values for the variable x_2. Under these conditions, the clustering of the variable x_2 is obtained with $accuracy = 1$. Conversely, the dependent variables are set so that p_2 is very different. Thus, the values of x_1 have a large effect on x_2. The experiment uses the values of the parameter $p_2(1) = [0,1\ \ 0,2\ \ 0,3\ \ 0,4]$ for the first cluster and $p_2(2) = [0,9\ \ 0,8\ \ 0,7\ \ 0,6]$ for the second cluster to estimate the pointer of the variable x_2 with $accuracy = 0,87$. The results are very good in both cases, which looks promising for using the algorithm on real data where it is not possible to influence the relationship between the variables x.

(iii) A non-binomial mixture, which can occur in real data, was then tested. This mixture is created using the parameter a, which affects the values of the probability function of each component of the binomial mixture. However, even in the case of a non-binomial mixture (e.g. $a = 1$), it is found that the deterioration of the conditions does not rapidly affect the clustering results, even though we estimate this mixture to be binomial ($a = 0$). Figure 6.6 shows a mixture with two binomial components with $p_1 = [0,1\ \ 0,9]$ that are influenced by the parameter $a = 1$. This slightly transforms the mixture into a non-binomial one, and the result of the clustering of the variable x_1 came out with an $accuracy = 0,77$, even though we use a small number of components for the experiment and also a small number of values in the variables. This indicates that the binomial distribution is very general and is a suitable choice for the algorithm under development.

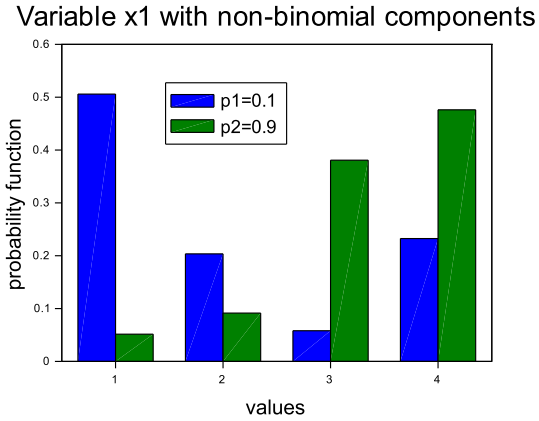

Figure 6.6: Probability function of the variable x_1 with two non-binomial components

The second part of the algorithm deals with local models, where the behavior of these models $f(x|y)$

in each component is tested. In this part, the accuracy of the classification of the target variable y, which describes the relationship of the variable y to x, is examined by changing the parameter py. This describes the parameters of the model of y, from which the probabilities for $y = 0$ and $y = 1$ are subsequently generated, depending on all combinations of values of the explanatory variables x. Therefore, we will first test an almost deterministic case where the parameter values are set to $py = [0,0001\ \ 0,9999]$ and the result $accuracy = 1$ shows that the classification of y is perfect. The deterministic nature of this parameter is then reduced (i.e., moved away from the values 0 and 1) with the values $py = [0,2\ \ 0,8]$, where it begins to depend on the clusters as the perfect $x - y$ coupling deteriorates. In this case, the *accuracy* is $0,81$, which indicates a deterioration of the classification, but still a very good result.

Algorithm development and testing (for 3 and 4 components)

The algorithm was then modified so that the first variable x_1 has 3 components and the second variable x_2 has 4 components, leaving the number of possible values of these variables unchanged. The change in the number of components caused an increase in the number of set parameters, and this induced a strong coupling of the variable x_1 to y. Although the clustering accuracy was severely degraded by overlapping components in the binomial mixture, as can be seen nicely in Figure 6.7, the classification of y remained almost error free, because the binding of the variable x_1 to y is still the same. The clustering is obtained with $accuracy = 0,80$ for x_1, then $accuracy = 0,63$ for x_2 and yet the classification of y using the parameters $py = [0,0001\ \ 0,9999]$ is perfect, i.e. with $accuracy = 1$.

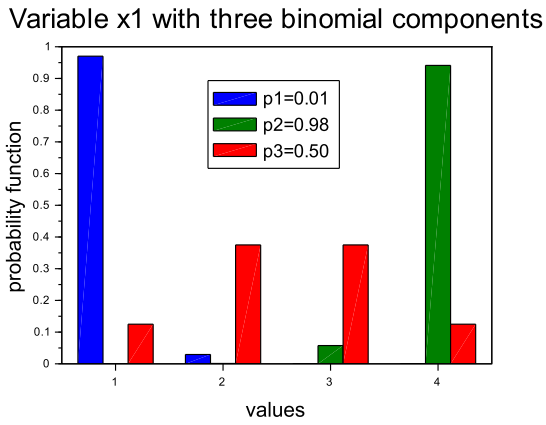

Figure 6.7: Probability function of the variable x_1 with three binomial components

Experiments show that it is still useful to use the algorithm even under less favorable conditions.

Final form of the developed algorithm

The marginal mixtures algorithm was the target of this thesis, and its Scilab code is given below. For clarity, all i cycles are for variables and all j cycles are for components. To run the code, copy and paste the functions from Appendix D into the appropriate line. The individual steps of the algorithm will now be discussed in detail.

The first part of the code sets the parameters (lines //1 to //18) for the subsequent simulation (lines //19 to //33), which generates the required data. The initial parameters are set to the exact values used in the simulation. This is to ensure that the algorithm works correctly. We then change the settings of these parameters and observe how the classification accuracy of the target variable y copes with this change and tune the parameter values.

Lines //1 to //3 specify the number of data used in the algorithm. Part of the simulated data is used for learning, i.e. initialization, estimation and determination of local models. The remaining data are used for testing in the form of the classification itself. Line //4 is used to set the parameter a discussed above, which allows the generated binomial parameters to be distorted to more closely approximate the uniform distribution. Now $a = 0$, which does not affect the generation of the binomial parameters, but increasing this value will change them. Lines //5 and //6 set the parameters of the pointer models that determine the number of components in each variable. The parameter pa_1 is set so that the first variable x_1 has a total of 3 components and the parameter pa_2 produces a total of 4 components in the variable x_2. In the next steps, parameters are set to generate explanatory variables x, which are assumed to be independent according to our assumption. For the previous example in Subsection 6.1.3, which deals with mixture estimation, it is tested that violating this assumption does not have a large effect on the results, so in the following code only one of the options is used, namely an almost independent relationship between the variables x_1 and x_2. This variant of the relationship may be the closest to real data. First, lines //7 to //9 generate categorical parameters from the probability function of the binomial distribution for all 3 components of the first variable x_1, defining the mixture model of the binomial distributions. For example, line //7 sets the value of the binomial parameter $p = 0,1$ to generate a probability function, and the elements of this probability function are taken as probabilities or parameters of the categorical distribution. This procedure is chosen in order to be able to subsequently establish a relationship (almost independence) between the variables x, since the binomial distribution does not have a conditional probability, whereas the categorical distribution does. Lines //10 to //13 set the parameters for the 4 components of the variable x_2 so that for each value

of the variable x_1 these parameters are binomial and always different. The dependency is projected so that for each value of x_1, x_2 has a different binomial distribution, so it is a mixture of binomial distributions controlled by x_1. Finally, lines //14 to //18 set the binomial parameter Py for the target variable y, which consists of 24 different values, because each combination of x_1 (4 values) and x_2 (6 values) is assigned a probability for $y = 0$ and also for $y = 1$.

After setting all the parameters, the simulation can be started in lines //19 to //33, where we first set the number of values in the variables x_1 and x_2 (lines //19 and //20). This is followed by the generation of pointer models for the two variables mentioned (lines //22 and //23), then the generation of values for the variable x_1 on line //24 using the appropriate pointer model. On line //25, the pp parameters for the variable x_2 are created by using both the pointer model of the variable x_2 and the generated value of the variable x_1 itself to create the desired relationship between these variables x (almost independence). Then, using the created parameters pp, the values of the variable x_2 are generated on line //26. Now the generated values of both variables x are encoded in z on line //27, and the values of y are generated on line //28 using this auxiliary variable z and the parameters Py (the pointer model must not be present here, as it would affect both variables equally, and we want to know the effect of the variable x on y). The matrix of variables x is created in line //30, from which the maximum of the values in each of the variables x is determined in line //31. The next step is to determine the number of values in y and the number of variables x (i.e. x_1 and x_2) in lines //32 and //33. A part of the simulated data is then selected for learning on line //34 and stored in the variables x and y, respectively xL and yL.

It is now possible to start the initialization on lines //35 to //45, which is described theoretically in Section 4.1 and used practically in the mixture estimation experiment in Subsection 6.1.3. First of all, a prior virtual data are set in the form of the strength of the initial parameter $k0$ (line //35), and then the initial counter statistic ka is set using this parameter (line //40), from which the statistic is generated (line //43). Lines //36 and //37 set the initial parameters for all components of both variables x. Next, on lines //39 and //40, the number of components in the variable x_i and the aforementioned counter ka for x_i are determined. The initialization is completed by a cycle on lines //41 to //44, where the parameter estimates of the models for the variable x_i and the summation statistic of the variable x_i are determined for each component.

The next part of the code on lines //46 to //65 is devoted to mixture estimation, which is described in Chapter 4 and can be divided into two parts in the code: determining the weights and updating the

statistics (which includes the parameter estimation itself). In the first part, q is defined in line //48, then the proximities are then stored in the cycle for all the components in lines //49 to //51. The following line //52 creates the weights and there is also a solution if there is only one component in the experiment. These weights are remembered in line //53. The second part, which focuses on updating the statistics, is in lines //54 to //59, where the counter is updated in the cycle for each variable, then the statistics are updated, and finally the parameter estimation is performed and then remembered. On line //62, the evolution of the estimated parameters is then saved in a table and finally, on line //64, the point estimate of the pointers is determined using the argument of the maximum of the determined weights.

The following part on lines //66 to //82 deals with the creation of local categorical models, which is described in detail in Section 5.2. On line //68, the variables belonging to the j-th cluster of the i-th variable are collected so that each component in each variable has data belonging to its cluster on line //73. The next cycle on lines //76 to //81 generates the desired local models $f_j\left(x_i|y\right)$, which are created by normalizing the local frequency table $T_j\left(x_i|y\right)$ so that for a given y the sum of x is equal to 1. A part of the construction of local models is the model $f\left(y\right)$ on line //82, which is created by normalizing the frequencies of the values of y to the sum of 1 and is necessary for the next part of the algorithm, i.e. classification.

Now the learning phase is over and the testing phase begins, which is preceded by deleting all existing data (variables x_1, x_2 and y) and loading new data for testing. This last part of the algorithm (lines //85 to //105) is the classification mentioned above, the detailed procedure of which is described in Section 5.3. In the code, the classification can be divided into two parts, as in the estimation, namely the determination of the weights for the prediction (lines //86 to //93) and the prediction itself (lines //94 to //103). The determination of the weights in this phase of the algorithm is the same as in the mixture estimation. The first step is the definition of q, where the proximities are then inserted and, after their normalization, the weights are obtained and stored for later use. The next step is to predict the target variable y by first determining the probability of fy from the model $f\left(y\right)$. Line //96 defines fj, which is then used on line //98 to determine the model $f\left(x_i|y\right)$ using a weighted sum of components. This is followed on line //100 by the construction of the prediction model $f\left(y|x\right)$ using the Naive Bayes principle as the product of $f\left(y\right)$ and $f\left(x_i|y\right)$ over all variables. The prediction of the target variable yp itself is determined on line //102 as the argument of the maximum of the prediction model $f\left(y|x\right)$, i.e. the y with the highest probability is selected. Finally, line //104 renames

the predicted target variables yp in the case of rotation in prediction, and line //105 determines the accuracy of this prediction by comparing the predicted and actual values of y, which is also known as the classification y.

```
// Estimation of the marginal mixture f(x1,x2|c) with binomial x
// - different components
// - x1,x2 more or less dependent
// - estimated as for independent x
// - learning and testing data
// - simBin - corrupted binomial (for "a" large we get categorical)
// ---------------------------------------
clear, clc, mode(0);
// copy the functions from the Appendix D here and run the code

nd=800;                          // number of data                    //1
nL=500;                          // number of learning data           //2
nT=nd-nL;                        // number of testing data            //3
a=.0;       // corruption of binomial distibution (see the function) //4

pa1=[.3 .4 .3];                  // pointer model parameters for x1   //5
pa2=[.2 .3 .3 .2];               // pointer model parameters for x2   //6

// categorical parameters for the 3 components of the binomial x1
C(1).p1=simBin(.01,3,a);                                             //7
C(2).p1=simBin(.98,3,a);                                             //8
C(3).p1=simBin(.5,3,a);                                              //9

// parameters for the 4 components of x2 - almost independent
C(1).p2=[.05 .1 .15 .2];                                             //10
C(2).p2=[.85 .9 .95 .98];                                            //11
C(3).p2=[.45 .5 .55 .6];                                             //12
C(4).p2=[.75 .77 .8 .82];                                            //13
```

```
py=[.0001*ones(1,12) .9999*ones(1,12)]; // parameters of y          //14
for k=1:24                                                          //15
  p=binomial(py(k),1);      // binomial parameters y for all x      //16
  Py(:,k)=p';               // parameters y for all combinations of x //17
end                                                                 //18

// SIMULATION
nx1=length(C(1).p1);                          // number of values x1  //19
nx2=6;                                        // number of values x2  //20
for t=1:nd                                                          //21
  c(1,t)=sum(cumsum(pa1)<randu())+1;          // pointer model for x1 //22
  c(2,t)=sum(cumsum(pa2)<randu())+1;          // pointer model for x2 //23
  x1All(t)=sum(cumsum(C(c(1,t)).p1)<randu())+1; // generation of x1   //24
  pp=simBin(C(c(2,t)).p2(x1All(t)),nx2-1,a);   // parameters for x2   //25
  x2All(t)=sum(cumsum(pp)<randu())+1;          // generation of x2    //26
  z(t)=xt2col([x1All(t) x2All(t)],[nx1 nx2]); // encoding x1 and x2   //27
  yAll(t)=sum(cumsum(Py(:,z(t)))<randu())+1;   // generation of y     //28
end                                                                 //29
xAll=[x1All x2All];              // matrix of variables x           //30
b=max(xAll,'r');                 // maximum of values in each x     //31
ny=max(yAll);                    // maximum of values in y          //32
nv=length(b);                    // number of variables x (x1,x2)   //33

// selection of data for learning
x=xAll(1:nL,:); y=yAll(1:nL); xL=x; yL=y;                           //34

// INITIALIZATION
k0=5;                            // strength of initial parameters  //35
X(1).pI=[.1 .5 .9];              // parameters for 3 components of x1 //36
X(2).pI=[.1 .3 .4 .9];           // parameters for 4 components of x2 //37
```

```
for i=1:nv                                                          //38
  nc(i)=length(X(i).pI);       // number of components in variable xi //39
  X(i).ka=k0*ones(1,nc(i));    // counter for xi                     //40
  for j=1:nc(i)                                                     //41
    X(i).c(j).pE=X(i).pI(j);  // parameters estimates of variable xi //42
    X(i).c(j).S=X(i).c(j).pE*X(i).ka(j)*(b(i)-1);
                              // summation statistics of variable xi //43
  end                                                               //44
end                                                                 //45

// ESTIMATION
for t=1:nL                                                          //46
  for i=1:nv                          // cycle for weights and updates //47
    // -- WEIGHTS
    q=zeros(1,nc(i));                                 // definition of q //48
    for j=1:nc(i)                                                   //49
      q(1,j)=binpdf(x(t,i)-1,X(i).c(j).pE,(b(i)-1));  // proximities //50
    end                                                             //51
    if length(q)==1, X(i).w=1; else X(i).w=fnorm(q); end
          // creation of weights and solution for only one component //52
    X(i).wt(:,t)=X(i).w';                          // remember the weights //53
    // -- ESTIMATION
    for j=1:nc(i)                                                   //54
      X(i).ka(j)=X(i).ka(j)+X(i).w(j);                // counter update //55
      X(i).c(j).S=X(i).c(j).S+X(i).w(j)*(x(t,i)-1);  // stat. update //56
      X(i).c(j).pE=X(i).c(j).S/(X(i).ka(j)*(b(i)-1)); // estimation //57
      X(i).c(j).pt(t)=X(i).c(j).pE; // remember - evolution of param.//58
    end                                                             //59
  end                                                               //60
end                                                                 //61
for i=1:nv, for j=1:nc(i), P(i,j)=X(i).c(j).pE; end, end
```

```
                    // save the evolution of parameters into a table //62
for i=1:nv                                                          //63
  cp(i,:)=amax(X(i).wt,1);      // point estimates of pointers      //64
end                                                                 //65

// LOCAL MODELS
for i=1:nv                                                          //66
  for j=1:nc(i)                                                     //67
    X(i).c(j).dt=[]; // definition of variables for data in clusters //68
  end                                                               //69
end                                                                 //70
for i=1:nv                                                          //71
  for t=1:nL                                                        //72
    X(i).c(cp(i,t)).dt=[X(i).c(cp(i,t)).dt; [y(t) x(t,i)]];
                              // creation of data in clusters       //73
  end                                                               //74
end                                                                 //75
for i=1:nv                                                          //76
  for j=1:nc(i)                                                     //77
    T=table(X(i).c(j).dt(:,1),X(i).c(j).dt(:,2),1:ny,1:b(i));
                              // local tables Tj(xi|y)              //78
    X(i).c(j).fy=fnorm(T,1);  // local models fj(xi|y)              //79
  end                                                               //80
end                                                                 //81
fY=fnorm(vals2(y));           // model f(y)                         //82

clear x1 x2 y                 // deletion of variables x1, x2 and y //83
// selection of data for testing
x=xAll(nL+1:nd,:); y=yAll(nL+1:nd);                                 //84

// CLASSIFICATION
```

```
for t=1:nT                                                              //85
  // --- WEIGHTS for prediction (same as for estimation)
  for i=1:nv                                                            //86
    q=zeros(1,nc(i));          // definition of q                       //87
    for j=1:nc(i)                                                       //88
      q(1,j)=binpdf(x(t,i)-1,X(i).c(j).pE,(b(i)-1)); // proximities      //89
    end                                                                 //90
    X(i).w=fnorm(q);           // weights                               //91
    X(i).Wt(:,t)=X(i).w';      // remember the weights                  //92
  end                                                                   //93
  // --- PREDICTION
  fy=fY;                       // probability of fy                     //94
  for i=1:nv                                                            //95
    fj=0;                      // definition of fj                      //96
    for j=1:nc(i)                                                       //97
      fj=fj+X(i).w(j)*X(i).c(j).fy(:,x(t,i));
                               // weighted sum of components            //98
    end                                                                 //99
    fy=fy.*fj;                 // product over variables                //100
  end                                                                   //101
  yp(t)=amax(fy);     // argument of the maximum f(y|x) = prediction     //102
end                                                                     //103
u=c2c(y,yp);          // renaming of variables yp in case of rotation    //104
Accuracy=acc(y,u(yp)) // accuracy of prediction y = classification y    //105
```

Results of the final algorithm

First of all, the parameters are set so that the algorithm works perfectly and the result of the code gives the *accuracy* = 1, i.e. 100%. This tested that the algorithm is fully functional and now we need to try other parameter settings and observe the deterioration of the results, because the real data will never be as nice as this simulated data. So the parameters py for the generation of the target variable y on line //14 were changed to use the new less deterministic parameters $py = [0,3\ \ 0,7]$ instead of the

original parameters $py = [0,0001\ \ 0,9999]$. The result, which is around $accuracy = 0,70$, shows that the accuracy has deteriorated and the reason is the not so accurate generation of the target variable y in the simulation (which was the goal).

In the following Section 6.3, the tested marginal mixtures algorithm is used for experiments on real data.

6.3 Experiments on real data

In this phase, the marginal mixtures experiment is performed on three types of real data. These are accident data, car data and medical data. All these data are discreet and it meets the requirements of multimodality.

6.3.1 Data for experiments

The real data used for the experiments and the results of their classification using marginal mixtures are described below.

Accident data

Accident data for the Czech Republic are obtained from Czech Police records. The data come from the cooperation of the Faculty of Transportation Sciences with the Prague City Hall on the analysis of traffic accidents in Prague. The data set consists of several parts (accidents, vehicles, consequences, pedestrians and GPS), and the records of accidents involving pedestrians from 2019 are used for this experiment.

The data set used has a total of 3219 records and these data are determined by the following attributes (explanatory variables):

- pedestrian category (5 values),
- pedestrian status (7 values),
- pedestrian behavior (7 values),
- situation at the accident site (6 values),
- gender of the pedestrian (4 values),
- provision of first aid (6 values).

The individual values of the described attributes are given in Appendix E.

The consequences for the life and health of pedestrians in the event of an accident is the target variable, which has 4 values, namely death, serious injury, minor injury and no injury.

To determine the results, i.e. the accuracy of prediction of the target variable, the code from section 6.2 is used. Since we are now using real data and not simulated data, it is necessary to remove the first part of this code (parameter setting and data simulation). Next, we need to modify the initialization parameters, which are based on the histograms of the loaded explanatory variables x. The complete code is shown in Appendix F, where the first part is replaced by loading the real accident data involving pedestrians, and then the initialization parameters mentioned above are modified according to the histograms of the variables x. The rest of the code remains the same and corresponds to the marginal mixtures method. Of the 3219 records, 1500 are used for the learning part and the rest, 1719 records, are used for testing. The accuracy of the prediction using marginal mixtures is 84,52% for the accident data.

Car data

Car Evaluation Data are publicly available on the website [48]. This data set relates to the condition of used cars and is suitable for this experiment due to its multivariate discrete data. The total number of data samples is 1728 and the attributes (explanatory variables) are:

- buying (4 values) - buying price (vhigh, high, med, low),
- maint (4 values) - price of the maintenance (vhigh, high, med, low),
- doors (4 values) - number of doors (2, 3, 4, 5 or more),
- persons (3 values) - capacity in terms of persons to carry (2, 4, more),
- lug_boot (3 values) - the size of luggage boot (small, med, big),
- safety (3 values) - estimated safety of the car (low, med, high).

For the car data, we chose car acceptability as the target variable. This variable has 4 values - unacceptable, acceptable, good and very good.

The algorithm of the marginal mixtures with car data determined the classification with an accuracy of 86,09%.

Medical data

In addition to transportation, it is possible to test the algorithm on real medical data. The Breast Cancer Data set used is publicly available on the website [48]. This data set contains 286 instances and the attributes (explanatory variables) are:

- age (9 values) - 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99,
- menopause (3 values) - lt40, ge40, premeno,
- tumor-size (12 values) - 0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59,
- inv-nodes (13 values) - 0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39,
- node-caps (2 values) - yes, no,
- deg-malig (3 values) - 1, 2, 3,
- breast (2 values) - left, right,
- breast-quad (5 values) - left-up, left-low, right-up, right-low, central,
- irradiat (2 values) - yes, no.

The target variable for the breast cancer data is a class with two values - no recurrence events and recurrence events.

The data set contains missing values, so 274 data from the sample are used in the experiment. The accuracy of the classification using marginal mixtures with medical data is 75,67%.

Results of experiments with marginal mixtures

Table 6.2 summarizes the classification accuracy results for real data using the marginal mixtures method.

Table 6.2: Classification accuracy using marginal mixtures method

Data	Accident	Car	Medical
Accuracy [%]	84,52	86,09	75,67

6.3.2 Results of experiments with other methods

Data mining searches for useful information in large data sets generated by advanced data collection technologies in transportation, medicine, and all sciences. Data mining methods consist of traditional data analysis methods and algorithms for processing extensive data files. These methods are used to obtain information through data analysis as well as classification [49]. However, they cannot always be applied to all data sets. The goal is to explore large databases and find new patents that would otherwise not be discovered. To compare the classification results of the marginal mixtures method, six well known data mining methods were selected:

- k-nearest neighbor,
- decision tree,
- neural networks,
- logistic regression,
- naive Bayes,
- fuzzy rules.

All of these methods are described below. The data mining system KNIME (www.knime.com) is used to determine the classification results, and Figure 6.8 shows how the program looks with accident data.

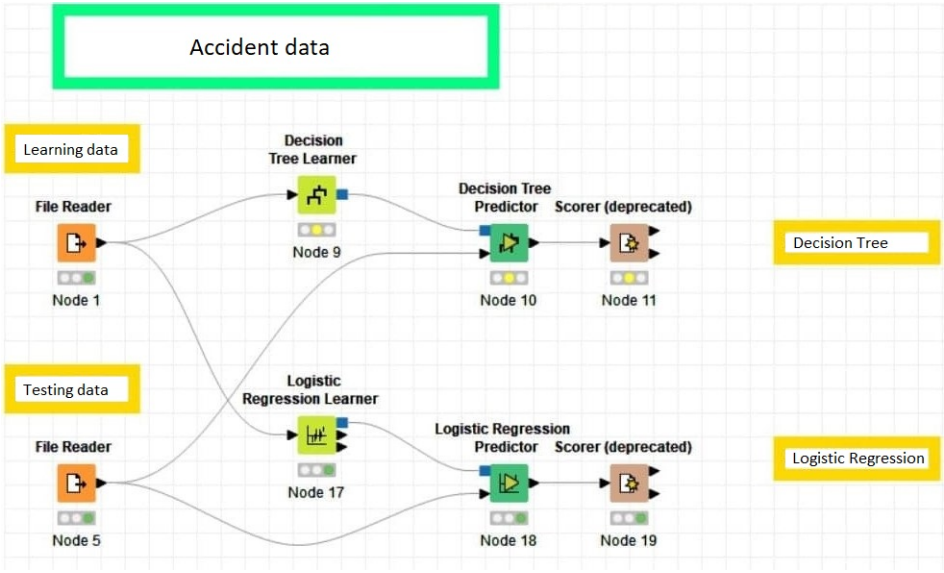

Figure 6.8: Illustration of the KNIME program with accident data

The KNIME system is one of the most renowned in its field, so its results are considered optimal. Our goal is to at least to approach them, as our algorithm is built in an approximate way, avoiding the theoretically correct way that leads to an extremely high dimension of the solution.

K-nearest neighbor

The k-nearest neighbor algorithm is commonly used for classification, estimation and classification. This method is based on instance learning and it uses a training data set. Thus, a new unclassified record can be classified by comparing it to the most similar records in the training data set. The condition is that the training set must contain only data that will not be used for classification [50]. Table 6.3 shows the classification accuracy results using the k-nearest neighbor method for all 3 types of real data, using the same conditions as for the marginal mixtures (e.g., 200 accident data for testing and the rest for learning).

Table 6.3: Classification accuracy using k-nearest neighbor method

Data	Accident	Car	Medical
Accuracy [%]	84,41	94,30	70,27

Decision tree

Decision tree is an attractive classification method. It involves the construction of a decision tree, a set of decision nodes connected by branches. These branches go down from the root node to the leaf nodes. The initial root node is located at the top of the decision tree diagram. Attributes are tested in decision nodes, and each possible outcome creates a new branch. Each branch then goes either to a final leaf node or to another decision node [50]. The accuracy of predicting transportation and medical data using the decision tree method is shown in Table 6.4.

Table 6.4: Classification accuracy using decision tree method

Data	Accident	Car	Medical
Accuracy [%]	84,87	97,81	72,97

Neural networks

The neural network method is inspired by the complex learning systems of the animal brain, which consist of interconnected sets of neurons. The structure of a single neuron can be relatively simple, but a dense network of interconnected neurons can create complex recognition or classification tasks.

Neural networks attempt to mimic the basic level of nonlinear learning that occurs in networks in nature.

The inputs are grouped from the data set and then combined using the combination function. One of the most commonly used functions is summation, which is an input to a function to create an output response. This response is then sent to other neurons [50]. Table 6.5 shows the classification accuracy results of the neural network for all real data.

Table 6.5: Classification accuracy using neural networks method

Data	Accident	Car	Medical
Accuracy [%]	84,53	77,63	71,62

Logistic regression

Logistic regression method describes the relationship between one or more explanatory variables and a response variable. The resulting variable is discrete and often has only two possible values. The basis is an appropriate model, which is also necessary for other methods. In contrast to the linear regression, which is used for the modeling of continuous data, the output of the logistic regression is discrete. To deal with this„ the logistic regression model uses the logit function, which shrinks the general continuous variable to a probabilistic interval $(0, 1)$. Then, instead of the output, the probability that the output is equal to one is modeled.

This method considers a sample of independent pairs of variables (x_i, y_i), $i \in \{1, 2, \cdots, n\}$, where x_i is an independent variable of the i-th attribute and y_i is a binary outcome variable. This outcome variable is coded as 0 and 1, indicating the presence or absence of the characteristic. To apply the logistics model to a data set, it is necessary to estimate unknown parameters. These parameters are estimated using the maximum likelihood method [51]. Table 6.6 shows the accuracy of the classification of the real data when the logistic regression method is used.

Table 6.6: Classification accuracy using logistic regression method

Data	Accident	Car	Medical
Accuracy [%]	84,18	78,07	70,27

Naive Bayes

Conventional statistics considers parameters as unknown but fixed numbers. The Bayesian approach

to statistics uses parameters as random variables, and these parameters are estimated from measured data according to the well known Bayes formula described in Section 2.2 [52].

The naive Bayes method estimates a conditional probability based on attributes that are conditionally independent. Assuming conditional independence, it is not necessary to determine the probability for each combination of $x = [x_1, x_2, \cdots, x_n]$, where n is the number of attributes. However, it is necessary to estimate the conditional probability of each x_t relative to y. This method is very practical because it does not require a large training set [49]. The classification accuracy results using the naive Bayes method for real data are shown in Table 6.7.

Table 6.7: Classification accuracy using naive Bayes method

Data	Accident	Car	Medical
Accuracy [%]	81,68	75,88	72,97

Fuzzy rules

Fuzzy logic, and in particular fuzzy rule technology, is a very effective method for solving complex systems. In addition to fuzzy rules, fuzzy modeling and fuzzy control are often part of the output. In fuzzy rules, fuzzy inference is used to determine the result from the specified input information. The input variable is assigned to the known information, then the result for the output variable is calculated using the consequent rules [53]. The resulting classification accuracy for real data using fuzzy rules is shown in Table 6.8.

Table 6.8: Classification accuracy using Fuzzy rules method

Data	Accident	Car	Medical
Accuracy [%]	82,66	94,04	71,64

6.3.3 Comparison of results

This chapter compares the results of data classification using all the methods presented in this thesis with the newly developed marginal mixtures method. The goal of marginal mixtures is to show that sufficient results can also be obtained using the approximation method (reducing the dimension of the model). For the sake of completeness, naive Bayes, the other approximation method, was also considered. The difference between the marginal mixtures approach and naive Bayes is that the marginal mixtures uses local modeling, which is described in Section 5.2. The summary of this

comparison in terms of accuracy results is shown in Table 6.9, and the best results for each data set are always marked in bold.

Table 6.9: Comparison of the classification accuracy

Method	Accident data	Car data	Medical data
K-nearest neighbor	84,41	94,30	70,27
Decision tree	**84,87**	**97,81**	72,97
Neural networks	84,53	77,63	71,62
Logistic regression	84,18	78,07	70,27
Naive Bayes	81,68	75,88	72,97
Fuzzy rules	82,66	94,04	71,64
Marginal mixtures	84,52	86,09	**75,67**

The first column of the table shows the classification accuracy results for accident data involving pedestrians. The best result for the accident data set was achieved by the decision tree method with a value of 84,87%. It was followed by neural networks with the result of 84,53%. The proposed method of marginal mixtures reached the third position with an accuracy of 84,52%, which is very close to the methods that have better results. On the other hand, the proposed method of marginal mixtures works with a smaller model dimension.

The second column of Table 6.9 shows the accuracy of the classification for car data. The best classification is obtained by the decision tree method with a value of 97,81%. Although the results of the decision tree method are very good, the complexity of the model can be a problem because all combinations of explanatory variables are examined. The marginal mixtures method uses only the reduced model and ranks fourth with an acceptable difference. It is interesting to note that neural networks (second best for accident data) rank only sixth out of seven for car data, which may be due to the insufficient representative amount of training data.

The methods were also applied to the medical data set. The results are shown in the last column of Table 6.9. The best classification method for this data set is the newly developed marginal mixtures method with a classification accuracy of 75,67%.

Moreover, for all three data sets, the marginal mixtures method achieves a better result than the similarly approximated naive Bayes method, which was the main objective of this newly developed method.

Chapter 7

Conclusion

The main objective of this research was to propose a model for data classification with dimension reduction based on clustering and local model construction. In the introduction of the thesis, two main goals were set, namely, to develop a method for modeling and predicting large discrete data sets using a low-dimensional model, and to compare this method with existing methods for prediction.

The first goal is achieved through the development of the marginal mixtures method, whose algorithm is not only described in detail in Chapter 5, but also verified experimentally first on simulated data (Section 6.2), and then on real data (Subsection 6.3.1). The main contribution of this method is the actual reduction of the model dimension while maintaining sufficient accuracy, which was achieved by modeling the explanatory variables as independent mixtures with binomial components (with the possibility of easy initialization from a prior data) and then constructing local categorical models for individual clusters. This is done by finding clusters and then building local models on top of them, which have a better chance of describing even complex situations. The subsequent prediction is then constructed by a weighted combination of predictions from the local models.

The second goal, i.e., to compare the marginal mixtures method with other methods, specifically the six selected prediction methods, is met in Subsection 6.3.3. This subsection provides an overview of the prediction accuracy results of the target variable for three different real discrete transportation and medical data sets. The results show that the marginal mixtures method is competitive with other methods as its prediction accuracy results are very good. Based on the evaluation results, it can be concluded that a different method is appropriate for each specific data set.

The advantage of our method is that it keeps the dimension of the model small and gets closer

to reality by using binomial mixtures. For example, a discrete categorical model with 10 variables of 7 values each has a dimension of 7^{10}, and our method reduces the size of this dimension to only $7 * 10$. Other methods (mainly neural networks and decision trees) do not consider the size of the dimension, and this can cause problems for really extensive discrete data sets. The first issue is related to overparameterization of the model, which in the worst case can lead to a failure to achieve the result. Another problem is time consumption. Due to the smaller dimension of the model, one of the benefits of marginal mixtures is increased speed. However, on real data sets, the results are almost the same for all methods compared, but the discrete model (which uses marginal mixtures) is much simpler and therefore faster.

Another great advantage of the presented method of data analysis is its easy initialization, which should bring the estimated model closer to reality. When there are more explanatory variables in the data set (which is always the case when dealing with questionnaires), we are working in a multidimensional space that is difficult to inspect. However, this thesis treats individual variables separately, and the locations of increased data density can be found simply from the histograms of the variables, so we avoid working in high-dimensional spaces.

Although marginal mixtures are a useful tool for data analysis, there are several issues that need to be addressed in future research. Discrete data are typical in their diversity and the frequency of the variable values (e.g., one value is more frequent and another one is much less frequent) and this leads to inaccuracies in the classification. Therefore, the goal is to find an appropriate approach to managing the data, since the predicted value will always be the value with the highest frequency.

In conclusion, the doctoral thesis fulfilled its purpose because the method was developed according to the requirements. It has been shown that the marginal mixture method is suitable in practice for classifying questionnaire data from different fields with sufficient results, and all the objectives of the thesis can be considered achieved.

Bibliography

[1] Verleysen, M. and François, D. (2005). The curse of dimensionality in data mining and time series prediction. In Computational Intelligence and Bioinspired Systems: 8th International Work-Conference on Artificial Neural Networks, IWANN 2005, Vilanova i la Geltrú, Barcelona, Spain. Springer Berlin Heidelberg, 8, pp. 758-770.

[2] Abdel-Aty, M. A., Hassan, H. M., Ahmed, M. and Al-Ghamdi, A. S. (2012). Real-time prediction of visibility related crashes. Transportation research part C: emerging technologies, 24, pp. 288-298.

[3] Jovanis, P. P. and Chang, H. L. (1989). Disaggregate model of highway accident occurrence using survival theory. Accident Analysis & Prevention, 21(5), pp. 445-458.

[4] Mannering, F. L. and Bhat, C. R. (2014). Analytic methods in accident research: Methodological frontier and future directions. Analytic methods in accident research, 1, pp. 1-22.

[5] Mannering, F. L., Shankar, V., and Bhat, C. R. (2016). Unobserved heterogeneity and the statistical analysis of highway accident data. Analytic Methods in Accident Research, 11, pp. 1-16.

[6] Pecherková, P. and Nagy, I. (2017). Analysis of discrete data from traffic accidents. In: 2017 Smart City Symposium Prague (SCSP). IEEE, pp. 1-4.

[7] Kaplan, R. M. and Saccuzzo, D. P. (2017). Psychological testing: Principles, applications, and issues. Cengage Learning. ISBN: 978-1337098137.

[8] Heymans, M. W. and Eekhout, I. (2019). Applied missing data analysis with SPSS and (R) Studio. Heymans and Eekhout: Amsterdam. Available online: https://bookdown. org/mwheymans/book-mi/.

[9] Alwin, D. F. (2007). Margins of error: A study of reliability in survey measurement. John Wiley & Sons. ISBN: 978-0-470-08148-8.

[10] Saris, W. E. and Gallhofer, I. N. (2014). Design, evaluation, and analysis of questionnaires for survey research. 2nd Edition, John Wiley & Sons. ISBN: 978-1118634615.

[11] Shanthi, S. and Ramani, R. G. (2012). Feature Relevance Analysis and Classification of Road Traffic Accident Data through Data Mining Techniques. In: Proceedings of the World Congress on Engineering and Computer Science, vol. 1, pp. 24-26.

[12] Peng, Y., Li, C., Wang, K., Gao, Z. and Yu, R. (2020). Examining imbalanced classification algorithms in predicting real-time traffic crash risk. Accident Analysis & Prevention, 144: 105610.

[13] Jozová, Š., Uglickich, E., Nagy, I. and Likhonina, R. (2022). Modeling of discrete questionnaire data with dimension reduction. Neural Network World, 32(1), pp. 15-41.

[14] Perrakis, K., Karlis, D., Cools, M. and Janssens, D. (2015). Bayesian inference for transportation origin-destination matrices: the Poisson-inverse Gaussian and other Poisson mixtures. Journal of the Royal Statistical Society: Series A (Statistics in Society), 178(1), pp. 271–296.

[15] Wiper, M., Insua, D. and Ruggeri, F. (2001). Mixtures of Gamma Distributions with Applications. Journal of Computational and Graphical Statistics, 10(3), pp. 440–454.

[16] Jozová, Š. and Nagy, I. (2020). Modelování a odhad měřených veličin. Automa. 26(1), pp. 26-31.

[17] Jozová, Š., and Nagy, I. (2020, June). Estimation of Discrete Data using Binomial Mixture. In 2020 Smart City Symposium Prague (SCSP). IEEE, pp. 1-5.

[18] Kárný, M., Böhm, J., Guy, T. V., Jirsa, L., Nagy, I., Nedoma, P., and Tesař, L. (2006). Optimized Bayesian Dynamic Advising: Theory and Algorithms. Springer-Verlag, London. ISBN: 978-1852339289.

[19] Nagy, I., Suzdaleva, E., Kárný, M., and Mlynářová, T. (2011). Bayesian Estimation of Dynamic Finite Mixtures. International Journal of Adaptive Control and Signal Processing, 25(9), pp. 765-787.

[20] Suzdaleva, E. and Nagy, I. (2018). An online estimation of driving style using data-dependent pointer model. Transportation Research Part C: emerging technologies, 86, pp. 23–36.

[21] Suzdaleva, E., Nagy, I. and Mlynářová, T. (2016). Expert-based initialization of recursive mixture estimation. In: 8th IEEE International Conference on Intelligent Systems, 2016, September 4-6, Sofia, Bulgaria, pp. 308-315.

[22] Nagy, I. and Suzdaleva, E. (2017). Algorithms and Programs of Dynamic Mixture Estimation. Unified Approach to Different Types of Components. SpringerBriefs in Statistics. Springer International Publishing. ISBN: 978-3-319-64670-1.

[23] Safri, Y. F., Arifudin, R. and Muslim, M. A. (2018). K-nearest neighbor and naive Bayes classifier algorithm in determining the classification of healthy card Indonesia giving to the poor. Sci. J. Informatics, 5(1), pp. 9-18.

[24] Sharma, H. and Kumar, S. (2016). A survey on decision tree algorithms of classification in data mining. International Journal of Science and Research (IJSR), 5(4), pp. 2094-2097.

[25] Zhang, G. P. (2000). Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 30(4), pp. 451-462.

[26] Komarek, P. (2004). Logistic regression for data mining and high-dimensional classification. Technical report, PhD Thesis, Carnegie Mellon University.

[27] Ishibuchi, H., Nozaki, K. and Tanaka, H. (1992). Distributed representation of fuzzy rules and its application to pattern classification. Fuzzy sets and systems, 52(1), pp. 21-32.

[28] Gupta, M. R. and Chen, Y. (2011). Theory and use of the EM algorithm. Foundations and Trends® in Signal Processing, 4(3), pp. 223-296.

[29] Böhning, D. (2003). The EM algorithm with gradient function update for discrete mixtures with known (fixed) number of components. Statistics and Computing, 13, pp. 257-265.

[30] Karlis, D. (2005). EM algorithm for mixed Poisson and other discrete distributions. ASTIN Bulletin: The Journal of the IAA, 35(1), pp. 3-24.

[31] Train, K. E. (2008). EM algorithms for nonparametric estimation of mixing distributions. Journal of Choice Modelling, 1(1), pp. 40-69.

[32] Ishwaran, H., James, L. F. and Sun, J. (2001). Bayesian Model selection in finite mixtures by marginal density decompositions. Journal of the American Statistical Association, 96(456), pp. 1316–1332.

[33] Jozová, Š., Uglickich, E. and Nagy, I. (2021). Bayesian Mixture Estimation without Tears. In: 18th International Conference on Informatics in Control, Automation and Robotics (ICINCO), pp. 641-648.

[34] Simmons, G. F. (1995). Calculus With Analytic Geometry. 2nd Edition, McGraw-Hill Education. ISBN: 978-0070576421.

[35] Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, pp. 4-15.

[36] Johnson, N. L., Kemp, A. W. and Kotz, S. (2005). Univariate discrete distributions (Vol. 444). 3rd Edition, John Wiley & Sons. ISBN: 978-0-471-27246-5.

[37] Nagy, I. (2003). Základy bayesovského odhadování a řízení. CTU, Prague. ISBN: 80-01-02748-1.

[38] Bolck, A., Croon, M. and Hagenaars, J. (2004). Estimating latent structure models with categorical variables: One-step versus three-step estimators. Political analysis, 12(1), pp. 3-27.

[39] Krishnamoorthy, K. (2006). Handbook of Statistical Distributions with Applications. 1st Edition, Chapman and Hall/CRC, New York. ISBN: 1-58488-635-8.

[40] Chew, V. (1971). Point estimation of the parameter of the binomial distribution. The American Statistician, 25(5), pp. 47-50.

[41] Kárný, M., Kadlec, J., Sutanto, E. L., Rojíček, J., Valečková, M. and Warwick, K. (1998). Quasi-Bayes estimation applied to normal mixture. In Preprints of the 3rd European IEEE Workshop on Computer-Intensive Methods in Control and Data Processing, Praha, Vol. 98, No. 3, pp. 77-82.

[42] Nagy, I., Suzdaleva, E. and Mlynářová, T. (2016). Mixture-based clustering non-gaussian data with fixed bounds. In 2016 IEEE 8th International Conference on Intelligent Systems (IS), IEEE, pp. 265-271.

[43] Nagy, I., Suzdaleva, E. and Pecherková, P. (2016). Comparison of Various Definitions of Proximity in Mixture Estimation. In: 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO), pp. 527-534.

[44] Suzdaleva, E. and Nagy, I. (2020). Practical Initialization of Recursive Mixture-Based Clustering for Non-negative Data. In: Gusikhin, O., Madani, K. (eds) 14th International Conference Informatics in Control, Automation and Robotics (ICINCO 2017), pp. 679-698. Lecture Notes in Electrical Engineering, vol 495. Springer, Cham.

[45] Suzdaleva, E., Nagy, I., Pecherková, P. and Likhonina, R. (2017). Initialization of Recursive Mixture-based Clustering with Uniform Components. In: 14th International Conference Informatics in Control, Automation and Robotics (ICINCO), pp. 449-458.

[46] Křivda, V., Richtář, M. and Olivková, I. (2007). 2. Silniční doprava. VSB – Technical university of Ostrava. ISBN: 978-80-248-1521-3.

[47] Ředitelství silnic a dálnic ČR (2009) [online]. Sčítání dopravy, stupně provozu a detekce kolon. [Cit. 13.1.2021]. Available from: https://portal.dopravniinfo.cz/.

[48] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[49] Tan, P. N., Steinbach, M. and Kumar, V. (2006). Introduction to Data Mining. Pearson Education. ISBN: 978-0-321-42052-7.

[50] Larose, D. T. (2005). Discovering Knowledge in Data: An Introduction to Data Mining. John Wiley & Sons, Inc., New York. ISBN: 0-471-66657-2.

[51] Hosmer, D. W. and Lemeshow, S. (2000). Applied logistic regression. 2nd Edition, John Wiley & Sons, Inc., New York. ISBN: 0-471-35632-8.

[52] Larose, D. T. (2006) Data Mining Methods and Models. John Wiley & Sons, Inc., New York. ISBN: 978-0-471-66656-1.

[53] Huaguang, Z. and Derong, L. (2006). Fuzzy Modeling and Fuzzy Control. Birkhäuser, Boston. ISBN: 978-0-8176-4539-7.

List of Figures

List of Tables

Appendices

Appendix A

Full derivation of mixture estimation

A mixture model consists of n_c components

$$f_j\left(x|p\right),\, j \in \left\{1, 2, \cdots, n_c\right\},$$

where j denotes the component and p is a vector of parameters (for a binomial component it is a vector of probabilities p_j), and the corresponding categorical pointer model

$$f\left(c = j|\alpha\right) = \alpha_j\,,$$

where c is the pointer variable and α is the probabilistic vector of the parameters.

To derive an algorithm for the estimation of the parameters p and α, we express the probability of all unknown objects conditional on all data $x\left(t\right)$ measured up to the current time point t and factorize it using the chain rule

$$f\left(c_t = j, p, \alpha|x\left(t\right)\right) \underbrace{\propto}_{Bayes\ rule} f\left(x_t, c_t = j, p, \alpha|x\left(t-1\right)\right) =$$

$$= f\left(x_t|c_t = j, p_j\right) f\left(c_t = j|\alpha\right) f\left(p|x\left(t-1\right)\right)\right) f\left(\alpha|x\left(t-1\right)\right), \tag{A.1}$$

where we used the following assumed independencies:

- the component models do not depend on the parameters α, and the pointer model is independent on p,
- the parameters p of the components do not depend on the switching parameter α,
- the components are independent.

Equation (A.1) can be divided into several parts, which will be discussed now. The first part of the equation can be rewritten in the form of a component of the mixture model

$$f\left(x_t|c_t=j,p_j\right)=f_j\left(x_t|p_j\right).$$

The second one is the pointer model

$$f\left(c_t=j|\alpha\right)=\alpha_j$$

and the rest are prior distributions of parameters p and α

$$f\left(p|x\left(t-1\right)\right)),\ \ f\left(\alpha|x\left(t-1\right)\right).$$

A typical application of Bayes rule is shown in Equation (A.1). However, the main problem lies in the variable c_t - its elimination leads to the sum form of a Bayes rule and it causes the growing complexity when updating the statistics. Therefore, the classification of the measured data x_t with respect to the individual components must be performed at the beginning. In other words, we need to construct the probability function $f\left(c_t=j|x\left(t\right)\right),\ \forall j$. This can be done with the formula (A.2)

$$\begin{aligned}
f\left(c_t=j|x\left(t\right),p,\alpha\right)&\propto\int_0^1\int_0^1 f\left(c_t=j,p,\alpha|x\left(t\right)\right)dp\,d\alpha=\\
&=\int_0^1\int_0^1 f\left(x_t|c_t=j,p_j\right)f\left(c_t=j|\alpha\right)f\left(p|x\left(t-1\right)\right))f\left(\alpha|x\left(t-1\right)\right)dp\,d\alpha=\\
&=\int_0^1\underbrace{f\left(x_t|c_t=j,p_j\right)f\left(p|x\left(t-1\right)\right))}_{\to f(p|x(t))}dp\times\int_0^1\underbrace{f\left(c_t=j|\alpha\right)f\left(\alpha|x\left(t-1\right)\right)}_{\to f(\alpha|x(t))}d\alpha\,,
\end{aligned}\tag{A.2}$$

where the first term in the product is the prediction from the j-th component and the second one is the prediction from the pointer model.

Using the most recent point estimates of the parameters $\hat{p}_{t-1}$ and $\hat{\alpha}_{t-1}$, we can write the weights

$$w_j = f\left(\hat{x}|\hat{p}_{j;t-1}\right)\hat{\alpha}_{j;t-1}\,.$$

The weights are the product of the component model proximity and the pointer model proximity with the actual point estimates of the parameters and the measured value of $x_t = \hat{x}$ inserted.

Using the Bayes rule and the product form of the models

$$f_{c_t}\left(x_t|p_{c_t}\right) = \prod_j f_j\left(x_t|j\right)^{\delta(c_t,j)} \text{ and } f\left(c_t|\alpha_{c_t}\right) = \prod_j f\left(j|\alpha_j\right)^{\delta(c_t,j)}\,,$$

where $\delta(\cdot)$ is the Kronecker function, $\delta(a,b) = \begin{cases} 1 & \text{for } a = b \\ 0 & \text{elsewhere} \end{cases}$ and we can derive the following algorithm for the estimation:

Algorithm

Initial part

- Construct the prior statistics S and κ (for binomial components) and ν for the pointer model and construct the prior point parameter estimates.

Recursive part (for $t = 1, 2, \cdots, T$)

- Measure the current data x_t.
- Evaluate the weights $w_{t;j}$ for $j \in \{1, 2, \cdots, n_c\}$

$$w_{j;t} = f\left(c_t = j|x\left(t\right), \hat{p}_{t-1}, \hat{\alpha}_{t-1}\right)$$

  using the formula (A.2) with the current parameter estimates $\hat{p}_{t-1}$ and $\hat{\alpha}_{t-1}$.

- Perform a weighted update of the statistics

$$S_{j;t} = S_{j;t-1} + w_{j;t}x_t\,,$$

$$\kappa_{j;t} = \kappa_{j;t-1} + w_{j;t}\,,$$

$$\nu_{j;t} = \nu_{j;t-1} + w_{j;t}\,,$$

where the data are added to a component only with the ratio corresponding to the probability that it belongs to a particular component.

- Construct the point estimate of the parameters

$$\hat{p}_{j;t} = \frac{S_{j;t}}{\kappa_{j;t}} \text{ and } \hat{\alpha}_t = \aleph\left(\nu_{j;t}\right),$$

where $\aleph(\cdot)$ means normalization to the sum equal to one.

Appendix B

Theoretical description of each initialization point

1. Data area

Suppose we have 3 variables x_1, x_2 and x_3 arranged in a data matrix x with three columns $'c'$ and as many rows as the number of measurements. Then

$$mi = \min(x,' c'), \quad ma = \max(x,' c')$$

gives the 3-element vectors of the minimum and maximum values of the variables. We can place the coordinates of the initial components in this space with the command

```
for i=1:nc
      thI(:,i)=(mi+ma)/2+.2*(ma-mi).*rand(3,1,'n')
end
```

where $(mi + ma)/2$ is the center of the region, $(ma - mi)$ is the width of the region, and nc is the number of components.

2. Initial parameter estimates

Great emphasis is placed on the initial location of the parameters (cluster centers for static components). The centers should definitely lie in the region where the data occur, and ideally, the individual centers should lie near the peaks of the data, i.e., at the locations where the density maxima of the expected clusters (working modes of the system) occur.

If we do not have prior data, we follow the previous point (we scale the data and decide the centers around the origin).

If prior data are available (and it should be, because the process has been running somehow - even just a trial - and usually all we need to do is put some effort into it and the data will be found), we definitely want to use it. First of all, we determine the region where the data occur (see previous point) and then we look for density peaks - either in histograms of individual variables or in pairs of variables. The histograms are clear. We show the procedure for pairs:

We have 3 variables $x1$, $x2$ and $x3$. We plot xy-graphs for the pairs $x1 - x2$ and $x2 - x3$.

`plot(x1,x2,':')` and `plot(x2,x3,'.')`

On the x-axis of the first plot we find the coordinates corresponding to the centers of visible clusters and on the y-axis the corresponding y-coordinates. There may be multiple y-coordinates to one x-coordinate - then we record all of them, with the x-coordinates repeated.

The second figure shows all the y-coordinates from the first figure on its x-axis. To these, we assign the y-coordinates from the second figure and add them as a third number to the existing coordinates. We can continue in this way for more coordinates. We use the resulting coordinates as the component centers, i.e. the prior parameters of the static components.

The procedure is illustrated in the Figure B.1:

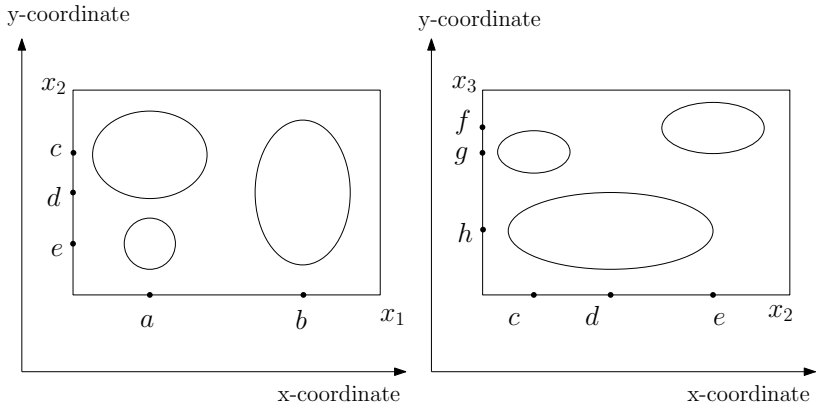

Figure B.1: Cluster centers for static components

The centers from the first figure will be:

$$C1 = [a, c], C2 = [a, e], C3 = [b, d].$$

From the second figure, add

$$C1 = [a, c, g], C2 = [a, e, f], C3 = [b, d, h].$$

These are not necessarily the true centers of the multivariate components, but at least we know that something is going on here and the initial centers somehow belong to the density vertices. The fine-tuning should happen in the actual estimation.

3. Holding of prior centers

This is a very important method, used more or less in every estimation.

At the beginning of the estimation, information about the parameters can only be derived from a small amount of data. If the parameters were left completely free, they would "rush" meaninglessly after each measured data vector and could easily stray somewhere from which there would be no return. That is why we need to start with statistics that already have some information in them - either from the data or from an expert.

We will demonstrate the situation for normal distribution of components. In other cases, the situation is similar.

The evolution of the statistics to estimate the regression coefficients is done as follows

$$V_t = V_{t-1} + \Psi\Psi' \,,$$

where V is the information matrix, $\Psi = [y_t,\, 1]'$ is the extended regression vector with the new data. This shows that the matrix V grows gradually as data are loaded into it.

It is important to note that if the matrix V is zero at the beginning, immediately the first data will change it a lot.

The point estimates of the regression coefficients follow the formula

$$\hat{\theta} = V_{\psi}^{-1} V_{y\psi} \,,$$

where $V_y, V_{y\psi}$ and V_ψ are the submatrices of V divided by the regression vector (in general, $\Psi = [y_t,\, \psi_t]'$; here, for static components, $\psi_t = 1$). If we multiply the matrix V by c, then

$$\hat{\theta} = (cV_\psi)^{-1} (cV_{y\psi}) = V_{\psi}^{-1} V_{y\psi} \,,$$

then the estimates will not change - only (for large c) cV will be larger, and hence more robust to changes due to newly coming data.

So the conclusion is quite simple: The large information matrix is used to hold the initial component centers.

Note: *If we have some initial parameters in mind $\hat{\theta}_0$ and we want to construct an information matrix for them, we proceed as follows*

$$V = \begin{bmatrix} 1 & \hat{\theta}_0' \\ \hat{\theta}_0 & 1 \end{bmatrix}.$$

Then the first guess

$$\hat{\theta} = V_{\psi}^{-1} V_{y\psi} = \hat{\theta}_0.$$

4. Fixed noise covariance

The noise covariance determines the shape of the clusters. If we are primarily concerned with finding the cluster centers, we can keep the covariances small and fixed (not estimating them). We specify them as a unit matrix multiplied by a tenth to a hundredth of the range (radius) of the expected data region.

If we care about the shape of the clusters, e.g. in classification when we divide the data into individual components, it is recommended to turn on their estimation only during the estimation process (e.g. in the middle), when the centers will be essentially found. However, there is still a danger that the covariances will run away or that one component will overlap the others.

5. Repeated estimation on the same data

The component centers start at their initial centers and gradually travel to the density peaks. Each data record is shifted a little according to which component it belongs to (according to the weights w). If the data sample is not fully sufficient, it can and does happen that the estimation ends before the centers have arrived to their proper places. Then it is reasonable to continue the estimation with the same data again, but to start not from the initial centers, but from those that have arrived so far.

There is one problem, however. At the end of the estimation, the information matrices are large (we say that the estimation is tight) and the centers would have moved either no more or very little. Therefore, between individual runs of the estimation, the statistics need to be suitably oblivious (divided by a number approximately equal to the length of the previous estimation).

This can be done repeatedly. In doing so, it is a good idea to follow the evolution of the centers, for example in a graph, and continue until the center estimates move.

6. Artificial regression vectors

A good way to convert often abstract expert knowledge into data that are suitable for initialization is the creation of artificial data vectors. Each data vector consists of a regression vector and its corresponding output value. We will illustrate the situation with the following example:

We observe the length of a queue in just one leg of a controlled intersection that collects traffic from

a certain area. There are 5 critical points in this area, which may be at different traffic levels depending on the situation. Thus, the regression vector will contain 5 variables (traffic levels in the area) and the output is the length of the queue at the intersection. The expert can convert his knowledge into an output of the most important combinations of loads for each location in the area and assign (according to his belief) the corresponding value of the length of the queue at the intersection.

Note: *If prior data are available, it is of course possible to use them and select some important data vectors that are critical for the situation and carry a lot of information. The selection can again be according to the expert's recommendation.*

The constructed data vectors are then treated normally as measured data vectors in the initialization.

7. Expert classification

This method follows the previous procedure, but instead of assigning an output value to a regression vector, the selected data are expertly assigned to the class (component) to which it belongs.

Again, there are several ways to perform this pre-classification.

1. Expertly create the entire data vector and classify it into a class.
2. Take some prior measured data vector and expertly assign a class to it.
3. Use some superior tools (let a human observe the situation or rent some expensive measuring instrument) to measure not only the data records but also the corresponding classification classes for prior measurement.

We use what we get for initialization, where we perform learning with the teacher (i.e., knowing the correct classification).

Appendix C

Generation of discrete data in Scilab

Let X is a random variable with distribution given by distribution function F. Then it holds

$$F(X) = U,$$

where U is uniform on the interval $(0, 1)$.

So, mapping uniform distribution on interval $(0, 1)$ gives values of random variable with distribution function $F(\cdot)$.

It holds for a continuous random variable. For the discrete one, the distribution function is given as a cumulative sum ($cumsum()$ of the probability function, which is a vector of probabilities of its individual values

x	1	2	3	$\cdots$	n
$f(x)$	p_1	p_2	p_3	$\cdots$	p_n

The probability function $f(x)$ and the distribution function $F(x)$ for $n = 3$ are shown in Figure C.1.

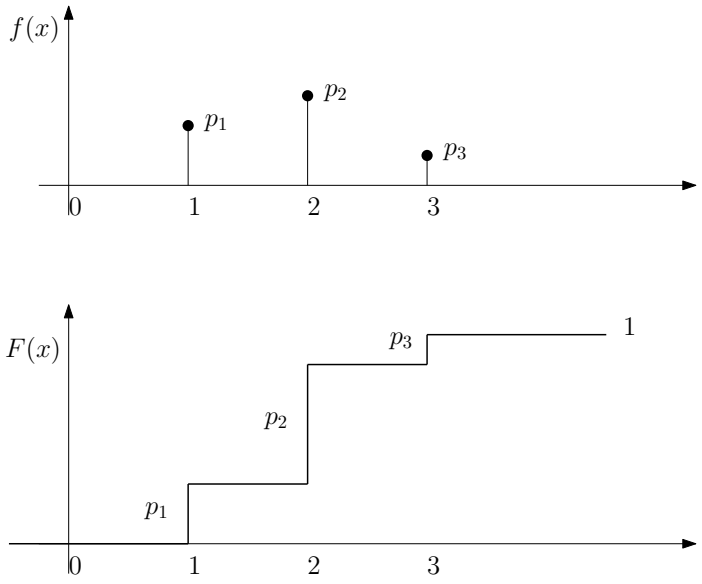

Figure C.1: The probability function $f(x)$ and the distribution function $F(x)$

Now, we generate values on the vertical axis uniformly from $(0,1)$. Discrete distribution function is a piecewise continuous function that has a finite number of jumps (intervals) in its values. The jumps at a specific value is equal to the probability of this value. The probabilities that we are going to hit a specific vertical interval, say i-th is p_i. Let us generate $U = u_0$ so that u_0 lies in the interval with the probability p_2, then $(F(x) < U) = [1,0,0]$ (where 1 denotes "true" and 0 "false"). Then

$$\sum (F(x) < u_0) = \sum [1,0,0] = 1\,.$$

Then $\sum (F(x) < u_0) + 1 = 2$ and we generate the value 2. As the probability that u_0 is from that interval is p_2, the probability of generating 2 is p_2. This corresponds to the probability function whose values we generate.

This example can be easily generated to the interval with probability p_i. Then

$$\sum (F(x) < u_0) + 1 = i$$

with probability p_i.

The whole function for generating discrete data in Scilab is as follows

$$y(t) = sum(cumsum(p) < rand(1,1,'u')) + 1$$

and we use it in the *for* loop. The procedure for generating discrete data $y \in \{1, 2, 3\}$ is described in the steps below:

1. create the cumulative sum of the probability function, if $p_1 = 0,31$, $p_2 = 0,52$ and $p_3 = 0,17$, then the cumulative sum gives the value

$$P_1 = 0,31,\ P_2 = 0,83,\ P_3 = 1\,,$$

2. use the rand function to generate a random variable in the interval $(0, 1)$. E.g.

$$u_0 = 0,45\,,$$

3. compare all values of the cumulative sum with the variable u_0 as follows $cumsum\,(p) < u_0$ with 1 denotes "true" and 0 "false"

$$0,31 < 0,45\ \ true \to 1\,,$$

$$0,83 < 0,45\ \ false \to 0\,,$$

$$0,1 < 0,45\ \ false \to 0\,,$$

4. determine the sum of the values 1 ("true") and 0 ("false")

$$\sum [1, 0, 0] = 1\,,$$

5. now we get the output value $y \in \{0, 1, 2\}$, but we want discrete data $y \in \{1, 2, 3\}$, so add 1 in this way $\sum (cumsum\,(p) < u_0) + 1$ and the result is

$$y = 1 + 1 = 2\,.$$

The result shows that if a random variable from the interval p_2 is generated, the output will be $y = 2$. Consequently, $y = 1$ is obtained for the interval p_1 and $y = 3$ for the interval p_3. This is shown in Figure C.2.

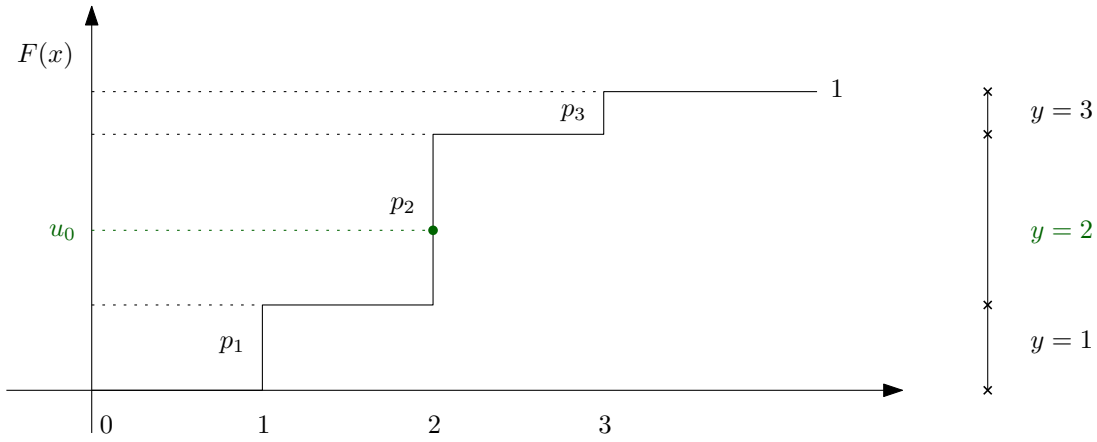

Figure C.2: Generation of discrete data

Appendix D

Functions for executing Scilab codes

For the sake of clarity of the codes in the thesis, both mixture estimation algorithms are given without functions. These functions are necessary for the operation of the code after execution in Scilab, but they are not essential for the core of the thesis. Therefore, both sets of functions are listed below, and once copied to the appropriate place in the code, the algorithms are fully functional. The first set of functions belongs to the algorithm for estimating the mixture $f(x_1, x_2|c)$ with binomial x, presented in Subsection 6.1.3. The second set of functions belongs to the experiment on simulated data in Section 6.2, which describes the estimation of the marginal mixture $f(x_1, x_2|c)$ with binomial x and also uses many functions that are not part of Scilab.

Set of functions for the estimation of the mixture $f(x_1, x_2|c)$ with binomial x

---

```
function y=randu(m,n)
  // uniform distribution (0,1)
  if argn(2)<1, m=1; n=1; end
  if argn(2)<2, n=1; end
  y=grand(m,n,'unf',0,1);
endfunction

function [pr,Lp]=binpdf(k,p,n)
  // probability of binomial pf for the value k
```

```
  m=length(k);
  num=factLn(n);
  den=factLn(k)+factLn(n-k);
  Lp=num-den+k*log(p)+(n-k)*log(1-p);
  pr=exp(Lp);
endfunction

function Lf=factLn(n)
  // logarithm of factorial
  m=length(n)
  for i=1:m
    Lf(i)=sum(log(1:n(i)));
  end
endfunction

function fn=fnorm(f,i)
  // fn=fnorm(f,i)   normalization of probabilistic table
  // fn  normalized table
  // f   table
  // i   direction i=1 norm colunms, i=2 norm rows
  // Rem: f can have zero rows or columns
  if argn(2)==1,
    [m,n]=size(f);
    if n>m, f=f'; end
    sf=sum(f);
    if sf==0
      fn=ones(f)/length(f);
    else
      fn=f/sf;
    end
  else
```

```
    [m n]=size(f);
    if i==1
      f1=sum(f,1);
      s=find(f1==0);
      f1(s)=m;
      f(:,s)=ones(m,length(s));
      fn=f./(ones(m,1)*f1);
    else
      f2=sum(f,2);
      s=find(f2==0);
      f2(s)=n;
      f(s,:)=ones(length(s),n);
      fn=f./(f2*ones(1,n));
    end
  end
endfunction

function [a,m]=amax(x,tx)
  // arg of max
  [n1,n2]=size(x);
  if (n1==1) | (n2==1)
    [m,a]=max(x);
  else
    if tx==1, tx='r'; end
    if tx==2, tx='c'; end
    [m,a]=max(x,tx);
    if tx=='r'       // if uniform denote it by %nan
      for i=1:size(x,2)
        if variance(x(:,i))==0
          a(i)=0
        end
```

```
      end
    else
      for i=1:size(x,1)
        if variance(x(i,:))==0
          a(i)=0
        end
      end
    end
end
endfunction


function [q,T]=c2c(ct,Ect)
  // cc=c2c(ct,Ect)   permutation of pointer values for plot
  // ct       simulated pointer
  // Ect      estimated pointer
  // q        order vector for Ect
  // T        transf. matrix
  // USSAGE:        ct=q(Ect)
  // set:  [q T]=c2c(ct,Ect);       simul and estim pointer
  // plot(1:nd,ct,1:nd,q(Ect))      plotting
  n=min([length(ct),length(Ect)]);
  if max(ct(1:n))~=max(Ect(1:n))
    disp 'WARNING from c2c.sci: Different numbers of components'
  end
  nc=max(ct);
  T=zeros(nc,nc);
  for t=1:n
    T(ct(t),Ect(t))=T(ct(t),Ect(t))+1;  // transformation matrix
  end
  for i=1:nc
    [xxx,q(i)]=max(T(:,i));              // order vector
```

```
  end
endfunction

function [ac,z]=acc(y,yp,n)
  // accuracy
  if argn(2)<3, n=0; end
  e=abs(y(:)-yp(:))<=n;
  ac=sum(e)/length(y);
  z.all=length(y);
  z.wrong=sum(y(:)~=yp(:));
  z.good=sum(y(:)==yp(:));
endfunction

function c=cord(x1,x2)
  // Spearman correlation coefficient
  c=correl(ranks(x1),ranks(x2));
endfunction

function j=ranks(a)
  // ranks of a discrete random variable
  // j  ranks, i.e. for a=[3 5 2 2] we get j=[3 4 1.5 1.5]
  //    for equal values in a, we set average rank
  // a  vector of discrete (and maybe also real) values
  [u,i]=gsort(a,'g','i');
  [xx,j]=gsort(i,'g','i');
  av=u(1);
  for i=2:length(a)
    if abs(u(i-1)-u(i))>1e-8
      av=[av u(i)];
    end
  end
```

```
  for i=1:length(av)
    ia=find(abs(a-av(i))<1e-8)
    if length(ia)>1
      k=j(ia);
      j(ia)=mean(k);
    end
  end
endfunction
```

Set of functions for the estimation of the marginal mixture $f(x_1, x_2|c)$ with binomial x

---

```
function f=simBin(p,N,a)
  // binomial distribution corrupted by noise
  if argn(2)<3, a=0; end
  fp=binomial(p,N)+a*rand(1,N+1,'u');
  f=fp/sum(fp);
endfunction

function y=randu(m,n)
  // uniform distribution (0,1)
  if argn(2)<1, m=1; n=1; end
  if argn(2)<2, n=1; end
  y=grand(m,n,'unf',0,1);
endfunction

function i=xt2col(x,b)
  // i=xt2col(x,b)  i is the column number of a model table with
  //                the regression vector xt with the base b;
  //                elements of x(i) are 1,2,...,nb(i)
  // it is based on the relation
```

```
  //          i=b(n-1)b(n-2)...b(1)(x(n)-1)+...+b(1)(x(2)-1)+x(1)
  n=length(x);
  b=b(:)';
  bb=b(2:n);
  bb=bb(:)';
  b=[bb 1];
  i=0;
  for j=1:n
    i=(i+x(j)-1)*b(j);
  end
  i=i+1;
endfunction

function [pr,Lp]=binpdf(k,p,n)
  // probability of binomial pf for the value k
  m=length(k);
  num=factLn(n);
  den=factLn(k)+factLn(n-k);
  Lp=num-den+k*log(p)+(n-k)*log(1-p);
  pr=exp(Lp);
endfunction

function Lf=factLn(n)
  // logarithm of factorial
  m=length(n)
  for i=1:m
    Lf(i)=sum(log(1:n(i)));
  end
endfunction

function fn=fnorm(f,i)
```

```
// fn=fnorm(f,i)    normalization of probabilistic table
// fn   normalized table
// f    table
// i    direction i=1 norm colunms, i=2 norm rows
// Rem: f can have zero rows or columns
if argn(2)==1,
  [m,n]=size(f);
  if n>m, f=f'; end
  sf=sum(f);
  if sf==0
    fn=ones(f)/length(f);
  else
    fn=f/sf;
  end
else
  [m n]=size(f);
  if i==1
    f1=sum(f,1);
    s=find(f1==0);
    f1(s)=m;
    f(:,s)=ones(m,length(s));
    fn=f./(ones(m,1)*f1);
  else
    f2=sum(f,2);
    s=find(f2==0);
    f2(s)=n;
    f(s,:)=ones(length(s),n);
    fn=f./(f2*ones(1,n));
  end
end
endfunction
```

```
function [a,m]=amax(x,tx)
  // arg of max
  [n1,n2]=size(x);
  if (n1==1) | (n2==1)
    [m,a]=max(x);
  else
    if tx==1, tx='r'; end
    if tx==2, tx='c'; end
    [m,a]=max(x,tx);
    if tx=='r'       // if uniform denote it by %nan
      for i=1:size(x,2)
        if variance(x(:,i))==0
          a(i)=0
        end
      end
    else
      for i=1:size(x,1)
        if variance(x(i,:))==0
          a(i)=0
        end
      end
    end
end
endfunction

function [T,kx,ky]=table(x,y,kx,ky)
  // T=table(x,y)    contingency table T(nx,ny)
  // x,y          data
  // kx,ky        values of x,y
  xv=vals(x);
```

```
  yv=vals(y);
  if argn(2)<3
    kx=xv(1,:);                  // if not given, read from data
    ky=yv(1,:);
  end
  ix=min(kx):max(kx);            // values by one
  iy=min(ky):max(ky);
  dx=min(xv(1,:))-min(kx);       // shift of data 1,2,3
  dy=min(yv(1,:))-min(ky);       // .. form given values e.g. 0,1,2
  x=x-min(x)+1;                  // data
  y=y-min(y)+1;                  // .. startingwith 1
  mx = length(ix);               // dimensions of the
  my = length(iy);               // .. final data
  T = zeros(mx, my);
  for t = 1:length(x)
    T(x(t)+dx,y(t)+dy) = T(x(t)+dx,y(t)+dy) + 1;
  end
endfunction


function [h,f]=vals(a)
  // [h f]=vals(a)  find different values of a and their frequencies
  // h        values and frequencies  [vals;abs_freq]
  // f        relative frequencies
  a=a(:)';
  b=gsort(a,'g','i');
  [v,m]=unique(b);
  dm=diff(m);
  n1=length(b)+1;
  n=[dm n1-m($)];
  f=n/sum(n);
  h=[v(:)';n];
```

```
  if sum(n)~=max(size(a))
    disp('Error: in vals.sci')
    return
  end
endfunction

function [n,f]=vals2(a)
  // [h f]=vals(a)  find different values of a and their frequencies
  //                Show only the frequencies
  // h        values and frequencies  [vals;abs_freq]
  // f        relative frequencies
  a=a(:)';
  b=gsort(a,'g','i');
  [v,m]=unique(b);
  dm=diff(m);
  n1=length(b)+1;
  n=[dm n1-m($)];
  f=n/sum(n);
  if sum(n)~=max(size(a))
    disp('Error: in vals.sci')
    return
  end
endfunction

function [q,T]=c2c(ct,Ect)
  // cc=c2c(ct,Ect)   permutation of pointer values for plot
  // ct       simulated pointer
  // Ect      estimated pointer
  // q        order vector for Ect
  // T        transf. matrix
  // USSAGE:        ct=q(Ect)
```

```
  // set:  [q T]=c2c(ct,Ect);        simul and estim pointer
  // plot(1:nd,ct,1:nd,q(Ect))       plotting
  n=min([length(ct),length(Ect)]);
  if max(ct(1:n))~=max(Ect(1:n))
    disp 'WARNING from c2c.sci: Different numbers of components'
  end
  nc=max(ct);
  T=zeros(nc,nc);
  for t=1:n
    T(ct(t),Ect(t))=T(ct(t),Ect(t))+1;  // transformation matrix
  end
  for i=1:nc
    [xxx,q(i)]=max(T(:,i));             // order vector
  end
endfunction

function [ac,z]=acc(y,yp,n)
  // accuracy
  if argn(2)<3, n=0; end
  e=abs(y(:)-yp(:))<=n;
  ac=sum(e)/length(y);
  z.all=length(y);
  z.wrong=sum(y(:)~=yp(:));
  z.good=sum(y(:)==yp(:));
endfunction
```

Appendix E

Description of accident data values

Accidents involving pedestrians have the following attributes:

- pedestrian category (5 values),
  - male,
  - female,
  - child (up to 15 years),
  - group of children,
  - other group,
- pedestrian status (7 values),
  - good (no adverse circumstances identified),
  - inattention, distraction,
  - under the influence of drugs, narcotics or alcohol,
  - physical disability (illness, nausea, reduced mobility, disability, etc.),
  - attempted suicide, suicide,
  - other unlisted condition,
  - not identified,
- pedestrian behavior (7 values),

  - correct, adequate,
  - poor judgement of distance and vehicle speed,
  - suddenly entering the carriageway from a pavement, verge, boarding or dividing island,
  - confused, rushed, indecisive behavior, sudden change of direction,
  - hitting a vehicle from the side,
  - children playing on the road,
  - none of the above,

- situation at the accident site (6 values),
  - pedestrian entering at the FREE or STOP signal,
  - crossing outside or near a crossing,
  - crossing at a marked crossing,
  - crossing immediately in front of or behind a stopped or parked vehicle,
  - walking or standing on the pavement, walking on the right or wrong side of the road,
  - other situations,

- gender of the pedestrian (4 values),
  - male,
  - female,
  - boy (up to 15 years),
  - girl (up to 15 years),

- provision of first aid (6 values),
  - there was no need for,
  - by the occupants of the vehicles involved in the accident,
  - by another person,
  - by the air ambulance,
  - by an ambulance,
  - not provided, but had to be provided.

Appendix F

Experiment with real data in Scilab

The code in Scilab for real accident data involving pedestrians is shown below.

```
// Marginal mixtures on real data
// - accident data involving pedestrians
// ---------------------------------------
clear, clc, mode(0);

xAll=csvRead('xAllR.csv',';');     // loading data - variables x       //1
yAll=csvRead('yAllR.csv',';');     // loading data - variable y        //2
[nd,nv]=size(xAll);                // number of data and variables x   //3
nL=1500;                           // number of learning data          //4
nT=nd-nL;                          // number of testing data           //5

b=max(xAll,'r');                   // maximum of values in each x      //6
ny=max(yAll);                      // maximum of values in y           //7
nv=length(b);                      // number of variables x (x1-x6)    //8

// selection of data for learning
x=xAll(1:nL,:); y=yAll(1:nL); xL=x; yL=y;                              //9

// INITIALIZATION
```

```
k0=5;                                  // strength of initial parameters   //10
X(1).pI=[.15 .9];                      // parameters for variable x1       //11
X(2).pI=[.2];                          // parameter for variable x2        //12
X(3).pI=[.2 .5];                       // parameters for variable x3       //13
X(4).pI=[.1 .6 .9];                    // parameters for variable x4       //14
X(5).pI=[.2];                          // parameter for variable x5        //15
X(6).pI=[.1 .4 .8];                    // parameters for variable x6       //16
for i=1:nv                                                                 //17
  nc(i)=length(X(i).pI);          // number of components in variable xi   //18
  X(i).ka=k0*ones(1,nc(i));       // counter for xi                        //19
  for j=1:nc(i)                                                            //20
    X(i).c(j).pE=X(i).pI(j);      // parameters estimates of variable xi   //21
    X(i).c(j).S=X(i).c(j).pE*X(i).ka(j)*(b(i)-1);
                                  // summation statistics of variable xi   //22
  end                                                                      //23
end                                                                        //24

// ESTIMATION
for t=1:nL                                                                 //25
  for i=1:nv                           // cycle for weights and updates    //26
    // -- WEIGHTS
    q=zeros(1,nc(i));                                  // definition of q   //27
    for j=1:nc(i)                                                          //28
      q(1,j)=binpdf(x(t,i)-1,X(i).c(j).pE,(b(i)-1));  // proximities       //29
    end                                                                    //30
    if length(q)==1, X(i).w=1; else X(i).w=fnorm(q); end
          // creation of weights and solution for only one component       //31
    X(i).wt(:,t)=X(i).w';                          // remember the weights //32
    // -- ESTIMATION
    for j=1:nc(i)                                                          //33
      X(i).ka(j)=X(i).ka(j)+X(i).w(j);                 // counter update   //34
```

```
      X(i).c(j).S=X(i).c(j).S+X(i).w(j)*(x(t,i)-1);  // stat. update  //35
      X(i).c(j).pE=X(i).c(j).S/(X(i).ka(j)*(b(i)-1)); // estimation  //36
      X(i).c(j).pt(t)=X(i).c(j).pE; // remember - evolution of param. //37
    end                                                               //38
  end                                                                 //39
end                                                                   //40
for i=1:nv, for j=1:nc(i), P(i,j)=X(i).c(j).pE; end, end
                    // save the evolution of parameters into a table  //41
for i=1:nv                                                            //42
  cp(i,:)=amax(X(i).wt,1);      // point estimates of pointers        //43
end                                                                   //44

// LOCAL MODELS
for i=1:nv                                                            //45
  for j=1:nc(i)                                                       //46
    X(i).c(j).dt=[]; // definition of variables for data in clusters  //47
  end                                                                 //48
end                                                                   //49
for i=1:nv                                                            //50
  for t=1:nL                                                          //51
    X(i).c(cp(i,t)).dt=[X(i).c(cp(i,t)).dt; [y(t) x(t,i)]];
                              // creation of data in clusters         //52
  end                                                                 //53
end                                                                   //54
for i=1:nv                                                            //55
  for j=1:nc(i)                                                       //56
    T=table(X(i).c(j).dt(:,1),X(i).c(j).dt(:,2),1:ny,1:b(i));
                              // local tables Tj(xi|y)                //57
    X(i).c(j).fy=fnorm(T,1);    // local models fj(xi|y)              //58
  end                                                                 //59
end                                                                   //60
```

```
fY=fnorm(vals2(y));              // model f(y)                              //61

clear x1 x2 y                    // deletion of variables x1, x2 and y      //62
// selection of data for testing
x=xAll(nL+1:nd,:); y=yAll(nL+1:nd);                                         //63

// CLASSIFICATION
for t=1:nT                                                                  //64
  // --- WEIGHTS for prediction (same as for estimation)
  for i=1:nv                                                                //65
    q=zeros(1,nc(i));            // definition of q                         //66
    for j=1:nc(i)                                                           //67
      q(1,j)=binpdf(x(t,i)-1,X(i).c(j).pE,(b(i)-1)); // proximities          //68
    end                                                                     //69
    X(i).w=fnorm(q);             // weights                                 //70
    X(i).Wt(:,t)=X(i).w';        // remember the weights                    //71
  end                                                                       //72
  // --- PREDICTION
  fy=fY';                        // probability of fy                       //73
  for i=1:nv                                                                //74
    fj=0;                        // definition of fj                        //75
    for j=1:nc(i)                                                           //76
      fj=fj+X(i).w(j)*X(i).c(j).fy(:,x(t,i));
                                 // weighted sum of components              //77
    end                                                                     //78
    fy=fy.*fj;                   // product over variables                  //79
  end                                                                       //80
  yp(t)=amax(fy);    // argument of the maximum f(y|x) = prediction         //81
end                                                                         //82
u=c2c(y,yp);         // renaming of variables yp in case of rotation        //83
Acc_y=acc(y,u(yp))   // accuracy of prediction y = classification y         //84
```

The same code is used for experiments on car data and medical data. The only changes are in the loading of the data (lines //1 and //2) and in the setting of the parameters in the initialization (lines //11 to //16). These parameters are set according to the histograms of all variables x selected for the experiment.

Appendix G

Publications

1. Jozová, Š., Uglickich, E., Nagy, I. and Likhonina, R. (2022). Modeling of discrete questionnaire data with dimension reduction. Neural Network World, 32(1), pp. 15-41.

2. Jozová, Š., Matowicki, M., Přibyl, O., Zachová, M., Opasanon, S. and Ziolkowski, R. (2021). On the Analysis of Discrete Data – Finding Dependencies in Small Sample Sizes. Neural Network World, 31(5), pp. 311-328.

3. Jozová, Š., Tobiška, J. and Nagy, I. (2021). On-line Recognition of Critical Driving Situations. Neural Network World, 31(3), pp. 227-238.

4. Jozová, Š., Uglickich, E. and Nagy, I. (2021). Bayesian Mixture Estimation without Tears. In: 18th International Conference on Informatics in Control, Automation and Robotics (ICINCO), pp. 641-648. ISBN: 978-989-758-522-7.

5. Jozová, Š. and Nagy, I. (2021). Use of Linear Regression to Discrete Data. In 2021 Smart City Symposium Prague (SCSP), pp. 1-6, New York: IEEE Press. ISBN: 978-1-6654-1524-8.

6. Jozová, Š. and Nagy, I. (2020). Estimation of discrete data using binomial mixture. In 2020 Smart City Symposium Prague (SCSP), pp. 1-5, New York: IEEE Press. ISBN: 978-1-7281-6821-0.

7. Jozová, Š. and Nagy, I. (2020). Modelování a odhad měřených veličin. Automa. 26(1), pp. 26-31. ISSN: 1210-9592.

8. Toman, P., Svoboda, J., Bouchner, P., Jozová, Š., Heřmanová, J., Mashko, A. and Válek, J. (2020). Motostudent Electric 2019/20 business project. [Technical Report] Praha: CTU FTS. Department of Vehicle Technology.

9. Svoboda, J., Heřmanová, J., Toman, P., Jozová, Š., Bouchner, P., First, J., Plomer, J., Mík, J., Skarolek, P., Ira, L., Růžička, M., Válek, J. and Rozhdestvenskiy, D. (2018). Motostudent Electric 2017/2018 project. [Technical Report] Praha: CTU FTS. Department of Vehicle Technology.

10. Toman, P., Svoboda, J., Heřmanová, J., Jozová, Š., Orlický, A., First, J., Bouchner, P., Plomer, J., Mík, J., Růžička, M., Paprčka, O., Válek, J. and Rozhdestvenskiy, D. (2018). Motostudent Petrol 2017/2018 project. [Technical Report] Praha: CTU FTS. Department of Vehicle Technology.

11. Toman, P., Mík, J., First, J., Svoboda, J., Bouchner, P., Orlický, A., Plomer, J., Rozhdestvenskiy, D., Heřmanová, J., Jozová, Š., Skarolek, P., Ira, L., Válek, J., Růžička, M. and Paprčka, O. (2018). Motostudent – Závodní elektrický motocykl 2018. [Functional Sample].

12. First, J., Toman, P., Bouchner, P., Plomer, J., Svoboda, J., Mík, J., Orlický, A., Válek, J., Heřmanová, J., Jozová, Š., Růžička, M. and Rozhdestvenskiy, D. (2018). Motostudent – Závodní benzínový motocykl 2018. [Functional Sample].

Publication references and citations

Citation of publications by Šárka Jozová (Author Identifier: 57218708304) according to the Scopus database (h-index: 1):

- Jozová, Š., Tobiška, J. and Nagy, I. (2021). On-line Recognition of Critical Driving Situations. Neural Network World, 31(3), pp. 227-238.

    1. Lehet, D. and Novotný, J. (2022). Assessing the feasibility of using eye-tracking technology for assessment of external HMI. In 2022 Smart City Symposium Prague (SCSP), pp. 1-6, New York: IEEE Press. ISBN: 978-166547923-3.

- Jozová, Š., Uglickich, E. and Nagy, I. (2021). Bayesian Mixture Estimation without Tears. In: 18th International Conference on Informatics in Control, Automation and Robotics (ICINCO), pp. 641-648. ISBN: 978-989-758-522-7.

1. Uglickich, E., Nagy, I. and Petrouš, M. (2021). Prediction of Multimodal Poisson Variable using Discretization of Gaussian Data. In: 18th International Conference on Informatics in Control, Automation and Robotics (ICINCO), pp. 600-608. ISBN 978-989-758-522-7.
2. Jozová, Š., Uglickich, E., Nagy, I. and Likhonina, R. (2022). Modeling of discrete questionnaire data with dimension reduction. Neural Network World, 32(1), pp. 15-41.