CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Bayesian Network Fisher Kernel for Categorical Feature Spaces

An Explorative Evaluation of a Similarity Measure on the Nominal Scale

Master's thesis in Computer Science and Engineering

Victor Ebberstein & Martin Holmberg

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

Bayesian Network Fisher Kernel for Categorical Feature Spaces

An Explorative Evaluation of a Similarity Measure on the Nominal Scale

Victor Ebberstein & Martin Holmberg

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2019

Bayesian Network Fisher Kernel for Categorical Feature Spaces
An Explorative Evaluation of a Similarity Measure on the Nominal Scale
Victor Ebberstein & Martin Holmberg

Supervisor:
Morteza Haghir Chehreghani, Department of Computer Science and Engineering
Advisor:
Tomi Silander, NAVER LABS Europe
Examiner:
Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2019

Bayesian Network Fisher Kernel for Categorical Feature Spaces
An Explorative Evaluation of a Similarity Measure on the Nominal Scale
Victor Ebberstein & Martin Holmberg
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Similarity measures between categorical feature vectors are non-intuitive and difficult to compute, since no definitive way of representing distances between two such vectors exists. The Fisher kernel provides a method for computing similarities by considering an underlying statistical model, which circumvents the problem of computing distances between categorical vectors. A promising probabilistic model is the Bayesian network, which is able to capture local dependencies between variables. In this thesis, the Fisher kernel based on discrete Bayesian networks is explored in a categorical setting. This new similarity measure between categorical vectors is primarily evaluated using the task of clustering. In addition, Bayesian networks are evaluated on the task of imputation in order to address the possibility of incomplete datasets.

By breaking down the structure of the Bayesian network into basic segments, the connection between the network structure and the produced Fisher similarities was investigated. The Fisher kernel was found to have great potential given that a suitable network structure was considered. However, this structure did not necessarily coincide with structures learnt using conventional learning methods for Bayesian networks.

Keywords: Bayesian network, Fisher kernel, kernel, clustering, imputation, categorical, similarity, machine learning.

Acknowledgements

First of all, we would like to thank our supervisor at Chalmers University of Technology, Morteza Haghir Chehregani, without whom this thesis would not have been possible. Secondly, we would also like to thank our advisor, Tomi Silander, who has been involved from the beginning of the project. Their constant support, feedback and advise have been invaluable in terms of progressing the work. We have greatly appreciated the opportunity to work under your supervision.

Lastly, we would like to thank our examiner at Chalmers University of Technology, Devdatt Dubhashi, for his support.

Victor Ebberstein & Martin Holmberg, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1
Introduction

This chapter introduces the concept of similarity measures for categorical variables. It continues by presenting the problem statement, the scope, and ends with an outline of the remainder of the report.

1.1 Background

Machine learning models often rely on both numerical and categorical data. In contrast to numerical variables, categorical variables such as color, brand and nationality offers no intuitive measure of distance. Consequently, there exists no intuitive measure of similarity between vectors consisting of categorical variables.

In order to compute similarity measures, used for tasks such as regression, classification and clustering, many methods are depending on kernels. A kernel is a way of computing pairwise similarities. It works by mapping all entries to an implicit higher dimensional space, after which the similarity between two entries are computed as their inner product.

Kernels in general come with a couple of challenges:

- They are often incompatible with incomplete data entries. Consequently, missing values need to be inferred before computing any similarities.
- They tend to belong to a larger family of kernels, with many different choices of free parameter values. Thus, the parameter values have to be chosen manually, since they influence the similarities.

Using a kernel to compute similarities between categorical vectors, rather than numerical, presents additional challenges:

- The distance between two entries, which is often used in kernel definitions, is not necessarily an accurate or meaningful representation when using categorical variables.
- Correlated variables have a greater impact on the similarity measure since they might be accounted for multiple times.

A kernel called the Fisher kernel provides a way around many of the above mentioned challenges [1]. The kernel is defined using an underlying generative probabilistic model of the data, which in this case was a Hidden Markov Model (HMM). Similarities between data entries are computed by considering how the probability of the entries would be affected by small changes in the underlying model. This kernel has the advantages of neither using any distance norm directly on the data entries, nor being defined using any free parameters. Additionally, the underlying model's ability to capture dependencies in the data can prove useful both in

imputing missing values, but it also influences the kernel to treat correlated and independent variables differently.

The Fisher kernel for discrete Bayesian networks was derived in parallel to this thesis. A Bayesian network is another generative probabilistic model that is useful for capturing local conditional dependencies between variables in a dataset [2]. This thesis consists of an evaluation of said kernel for different datasets consisting of categorical variables.

1.2 Objective

The aim of this project is to address the challenges of defining a similarity measure between vectors of possibly correlated categorical variables by studying the Fisher kernel derived for Bayesian networks. Even though the Fisher kernel is able to address these challenges, it is not necessarily a viable option since the similarities might be uninformative. The main research question is therefore formulated as:

Can the Fisher kernel be used to produce competitive results in tasks related to categorical variables?

Computing the Fisher kernel requires a Bayesian network. Additionally, exact inference in Bayesian networks is an acknowledged method of imputation, i.e., to infer all missing values in a dataset [3]. A natural property to investigate is how this approach compares to other common methods of imputation. The reason being that the most computationally expensive part, i.e., learning the network, already has been done when computing the Fisher kernel. The first sub-research question is therefore formulated as:

Is it beneficial to use imputation with a Bayesian network compared to other traditional approaches?

Finally, one can always derive new kernels from an existing kernel by making small modifications to the definition. The Fisher kernel can either be modified in this way, or by changing the underlying Bayesian network. Hence, the final sub-research question is:

Is there a variant of the Fisher kernel, or the Bayesian network, capable of improving the results?

1.3 Scope

The main limitations to the scope of the project follows naturally from the topic, namely that the approach is restricted to Fisher kernels based on Bayesian networks applied on purely categorical variables. In addition to this, only naive, yet commonly used, methods are used for comparison in the different tasks. These include methods for all of the investigated areas.

Moreover, since Bayesian networks are difficult to train, only relatively small

datasets, consisting of less than 25 variables and 1000 entries, will be used when evaluating its performance. In this way, the structure and parameter values of the Bayesian network can be computed optimally. Another imposed limitation on the datasets is that they should be publicly available. Finally, regardless of the investigated task, supervised metrics will be used whenever the performance of a model is evaluated, even when the class labels are not present at training.

1.4 Outline

The remainder of this thesis is structured as follows. Chapter 2 provides the necessary theory of Bayesian networks, kernels in general and the original Fisher kernel in particular. In Chapter 3, the techniques used to learn Bayesian networks are specified and the investigated kernels are introduced. In particular, the Fisher kernel for Bayesian networks is defined. In Chapter 4, the task of imputation is explored and evaluated for methods based on Bayesian networks. Chapter 5 considers the task of clustering, wherein the Fisher kernel and variants thereof are evaluated. Chapter 6 explores the relation between the underlying Bayesian network structure and the Fisher similarities. Finally, in Chapter 7 the results of the thesis are discussed, conclusions are presented and future work is suggested.

2
Theory

This chapter presents the underlying theory of Bayesian networks and kernels. It begins with an introduction to Bayesian networks, followed by a description on how they are constructed and used for inference. Furthermore, this chapter presents the concept of kernels as well as definitions of relevant kernels.

2.1 Bayesian networks

Bayesian networks, or Bayesian belief networks, are commonly used as a way of representing direct causal influences in a set of random variables [2]. The network can be seen as a probabilistic graphical model that models dependencies between variables. This allows for a compact representation, since only the dependency structure and the local conditional probabilities need to be stored. Even though the representation is compact, the network can still be used to efficiently infer missing values, in the context that the problem in general is NP-hard [4].

Table 2.1: Notation used for the model of a Bayesian network

$B = (G, \theta)$	A Bayesian network model consisting of a DAG, G, and parameters, θ.
$G = (X, E)$	A DAG consisting of a set of nodes, X, and a set of arcs, E.
X	The set of the n random variables in the model ($\|X\| = n$).
r_i	Number of possible values of the variable X_i ($X_i \in \{1, \ldots, r_i\}$).
E	The set of the m arcs in the model ($\|E\| = m$).
π_i	The set of the parent variables of X_i ($\pi_i \subset X$).
$\pi_i(X)$	Value configuration of the parents of X_i.
q_i	Number of possible values of $\pi_i(X)$ ($\pi_i(X) \in \{1, \ldots, q_i\}$).
θ_i	Model parameters for variable X_i ($\theta = \{\theta_1, \ldots, \theta_n\}$).
θ_{ij}	Model parameters for variable X_i, given that $\pi_i(X) = j$ ($\theta_i = \{\theta_{i1}, \ldots, \theta_{iq_i}\}$).
θ_{ijk}	Model parameters corresponding to $P(X_i = k \mid \pi_i(X) = j)$ ($\theta_{ij} = \{\theta_{ij1}, \ldots, \theta_{ijr_i}\}$).
D	A dataset
D_i	Entries of variable X_i in D.
$D_{i,\pi_i=j}$	Entries of variable X_i with parent configuration $\pi_i(X) = j$ in D.
D_{i,π_i}	Entries in D corresponding to the variables in $\{X_i\} \cup \pi_i$.
N_{ij}	Number of occurrences of X_i with parent configuration $\pi_i(X) = j$ in D.
N_{ijk}	Number of occurrences where $X_i = k$ with parent configuration $\pi_i(X) = j$ in D.
$\alpha_{ij}, \alpha_{ijk}$	Pseudo-count hyperparameters.

All of the necessary notation for this section is summarized in Table 2.1 found above, but it will also be introduced when first encountered.

2.1.1 Bayesian network model

A model of a Bayesian network, denoted as $B = (G, \theta)$, attempts to model the joint probability distribution of a set of random variables. The model consists of two parts:

- a directed acyclic graph (DAG), denoted as G, for describing the structure of the network,
- parameters θ describing the conditional probability distributions (CPDs) of each variable.

The DAG, $G = (X, E)$, is also split into two parts: a universe, or a set of random variables, X, and a set of arcs E. The notation π_i is used to specify the parents of X_i according to G, i.e., the set of variables with arcs going directly to X_i in the graph.

Figure 2.1 illustrates a Bayesian network with eight variables. The DAG is demonstrated by the arcs and describes the dependencies among the variables. As an example, the arcs $C \to F$ and $D \to F$ indicate that the value of F is dependent on the values of C and D. Consequently, the parents of F are $\pi_F = \{C, D\}$. However, if a variable is without incoming arcs, its parents are represented using the empty set, e.g., $\pi_A = \emptyset$.

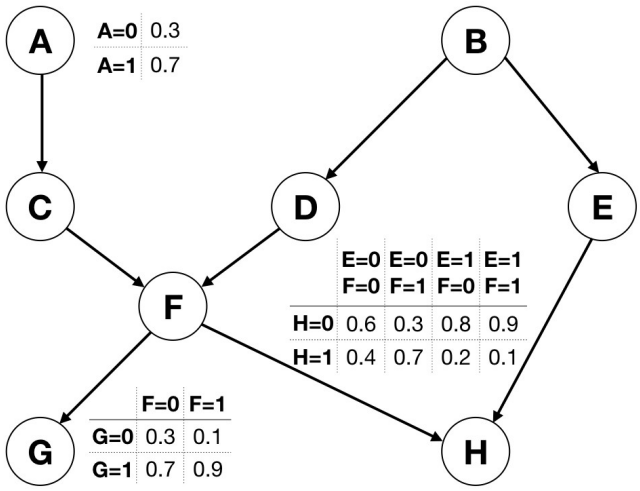

Figure 2.1: An example of a Bayesian network with the complete graph and a subset of the parameters.

The local CPDs are shown in the figure for A, G and H, which corresponds to the parameters θ_A, θ_G and θ_H. As an example, one can read from θ_H that $P(H = 1 \mid E = 0, F = 1) = 0.7$. In general, when the variables in question are categorical, the CPDs are represented using probability tables and the model can be populated with the corresponding parameters.

In order to properly define the parameters of the model, some additional notation is necessary. Consider a set of random categorical variables $X = \{X_1, \ldots, X_n\}$, where each variable X_i can take $r_i \geq 2$ different values. For simplicity, these values will be represented by integers, i.e., $X_i \in \{1, \ldots, r_i\}$. Also consider $\pi_i(X)$ as the value configuration of the variables in π_i. The possible configurations are again enumerated by integers, i.e., $\pi_i(X) \in \{1, \ldots, q_i\}$, where $q_i = \prod_{X_j \in \pi_i} r_j$. With the notation in place, the parameters of the Bayesian network model can be described. The model parameters represent the probability of X_i taking the value k, given that the parent configuration is j, meaning

$$\theta_{ijk} = P(X_i = k \mid \pi_i(X) = j). \tag{2.1}$$

The general model of a Bayesian network presented above is able to capture dependencies among all variables. The model is also able to represent any joint distribution while only utilizing local conditional probabilities.

2.1.1.1 Markovian factorization

A Bayesian network should be able to represent any joint probability distribution $P(X_1, \ldots, X_n)$. Consider the factorization of the distribution using the chain rule of probability,

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \bigcap_{j=1}^{i-1} X_j), \tag{2.2}$$

for an indexed ordering of the variables in X. This expression can be simplified even further since Bayesian networks satisfy the local Markov property. The local Markov property states that each variable X_i is conditionally independent of its non-descendants given π_i [5]. The joint distribution can therefore be expressed in the more compact form

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \pi_i). \tag{2.3}$$

Thus, Bayesian networks are sufficient to represent any joint probability distribution. Furthermore, looking at the model of a Bayesian network, the information stored precisely relates to the factors in Equation 2.3.

2.1.2 Learning a Bayesian network from data

Apparent from the model representation, the learning of a Bayesian network requires knowledge of both structure and parameters. This implies learning both the numerical values of the CPDs as well as the arcs in the network structure. Hence, learning a Bayesian network from data is similarly split into Parameter- and Structure learning.

2.1.2.1 Parameter learning

Learning the parameters is usually done in one of two ways: Either by computing the expected posterior (EP) estimate or by finding the maximum likelihood (ML) estimate. Both approaches are thoroughly described in Appendix A.1. In short, the two approaches differ in two aspects. Firstly, they use different prior distributions. Secondly, the technique for estimating the parameters given the posterior distribution differs.

In order to specify the parameter values exactly, the following notation is required:

- N_{ij} is the count of occurrences of X_i with parent configuration $\pi_i(X) = j$.
- N_{ijk} is the count of occurrences where $X_i = k$ with parent configuration $\pi_i(X) = j$.
- α_{ij} and α_{ijk} are pseudo-count hyperparameters.

The EP and ML estimate are expressed as

$$\theta_{ijk}^{\texttt{EP}} := \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}, \qquad \theta_{ijk}^{\texttt{ML}} := \frac{N_{ijk}}{N_{ij}}. \tag{2.4}$$

Whenever $N_{ij} = 0$, the ML estimate can not be uniquely determined since the corresponding parameters have no impact on the likelihood. However, it is common to use a uniform distribution over these parameters, i.e., assuming that all categories are equally probable.

Yet another method of selecting the parameters, based on an information theoretic approach, was proposed in [6]. The method is based on the normalized maximum likelihood (NML) distribution, which is family of information theoretic scoring functions for Bayesian networks. Inspired by the Bayesian approach, where the predictive probabilities coincide with the expected parameter values, the proposed method uses the predictive probability of NML to estimate the parameter values. The resulting parameters (cNML) are defined as:

$$\theta_{ijk}^{\texttt{cNML}} := \frac{e(N_{ijk})(N_{ijk} + 1)}{\sum_{k'=1}^{r_i} e(N_{ijk'})(N_{ijk'} + 1)}, \tag{2.5}$$

where $e(n) := (\frac{n+1}{n})^n$ and $e(0) := 1$.

All of the presented methods are applicable when learning the parameters of a Bayesian network. Since the methods optimize non-identical expressions and depend on different assumptions, they do not necessarily provide the same solution. The MAP estimate requires knowledge about a prior distribution of the parameters, whereas the cNML and ML estimates do not. In conclusion, the underlying assumptions and the prior knowledge should be used to guide the choice of method.

2.1.2.2 Structure learning

The many existing methods for learning a Bayesian network structure can be divided into two main categories: score- and constraint-based structure learning. In order to find the best structure, the score-based approach maximizes some scoring function depending on the structure, while the constraint-based approach usually exploits conditional independence statements. However, for the purpose of this thesis, only the score-based approach will be considered.

In general, one can describe a scoring function using two terms. The first term considers the fitness of a given structure to the dataset. The second term is a penalization term, which is used to avoid overfitting the structure to the data. Many scoring functions also omit the penalization term and only uses a measure of fitness.

The choice of scoring function is non-trivial and many choices exist relying on different assumptions. Henceforth, only one scoring function will be presented, which belongs to the Information theoretic scores. Other families of scoring functions, as well as other Information theoretic scores, are presented in Appendix A.2.

Information theoretic scores are based on the idea of compression and in contrast to other scoring functions, they are not dependent on any hyperparameters. In practice the size of an optimal description induced from the network is used as a measure of fitness. This idea corresponds to using the ML estimate of the parameters in Eq. 2.4 to compute the likelihood of the data given a network structure,

$$P_{\mathtt{ML}}(D \mid G) := P(D \mid G, \theta^{\mathtt{ML}}). \tag{2.6}$$

The normalized maximum likelihood (NML) criterion is information theoretic, although it uses a slightly modified fitness measure. This is defined as

$$P_{\mathtt{NML}}(D \mid G) := \frac{P_{\mathtt{ML}}(D \mid G)}{\sum_{D'} P_{\mathtt{ML}}(D' \mid G)}, \tag{2.7}$$

where the sum in the denominator goes over all possible datasets of the same size as D. This expression is obviously intractable as the size of the data increases. Thus, significant adjustments are needed to make it computationally feasible.

It has been proposed to consider NML for the column partitions of the data, i.e., for each variable independently [7]. Using the notation $D_{i,\pi_i=j}$, meaning all data entries of variable X_i where the parent configuration is $\pi_i(X) = j$, the likelihood of a single variable is given as:

$$P^1_{\mathtt{NML}}(D_{i,\pi_i=j} \mid G) := \frac{P_{\mathtt{ML}}(D_{i,\pi_i=j} \mid G)}{\sum_{D'} P_{\mathtt{ML}}(D' \mid G)}, \tag{2.8}$$

where $D' \in \{1, \dots r_i\}^{|D_{i,\pi_i=j}|}$.

The formula is defined for a single variable X_i and a single parent configuration $\pi_i(X) = j$, but it has been generalized to subsets of variables [8]. Let $S \subset X$ represent any subset of variables. In order to consider $P^1_{\mathtt{NML}}(D_S \mid G)$, all $\prod_{X_i \in S} r_i$ value configurations of S are collapsed into a single variable which is able to take $\prod_{X_i \in S} r_i$ unique values. Doing so, $P^1_{\mathtt{NML}}(D_S \mid G)$ can be used to model any subset of variables.

Using the generalized version of $P^1_{\mathtt{NML}}$, a scoring function called the quotient NML (qNML) has been proposed, [8]:

$$s_{\mathtt{qNML}}(G, D) := \sum_{i=1}^{n} \log \frac{P^1_{\mathtt{NML}}(D_{i,\pi_i}; G)}{P^1_{\mathtt{NML}}(D_{\pi_i}; G)}, \tag{2.9}$$

where both D_{i,π_i} and D_{π_i} are the subsets of the dataset corresponding to the variables in $\{X_i\} \cup \pi_i$ and π_i respectively. This score does not use any penalization term.

The qNML-score has a number of pleasant properties. First and foremost it is entirely free from hyperparameters, while still showing good results in finding an optimal structure. Additional properties of the score being discussed in [8] are equivalency, consistency, and regularity.

2.1.3 Inference

Inference in Bayesian networks is the task of updating beliefs based on a set of observed variables, the evidence, and to compute the revised probability distribution over a set of query variables. The general approach would be to compute joint probabilities corresponding to the evidence and marginalize over the remaining non-query variables by summation. However, computing joint probabilities in Bayesian networks has been shown to be an NP-hard problem [4]. Therefore, an approach based on the junction tree algorithm is commonly used, which often is efficient in practice [9]. The details of the algorithm are presented in Appendix A.3.

2.2 Kernel functions

Kernels produce a measure of similarity between data entries by computing their inner product in an implicit higher dimensional space, $\mathcal{V}$. Consider the entries $x, x' \in \mathcal{X}$, where $\mathcal{X}$ is the original feature space. For a mapping $\varphi : \mathcal{X} \to \mathcal{V}$, the kernel value for the entries x and x' is defined as

$$K(x, x') := \langle \varphi(x), \varphi(x') \rangle, \tag{2.10}$$

where $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is the inner product. However, since only the value of the inner product is of interest, a closed-form expression of $\varphi(\cdot)$ is not required. This is often referred to as the *kernel trick*.

2.2.1 Fisher kernel

The Fisher kernel was originally derived as a way of combining generative probability methods with discriminative models [1]. For two categorical vector entries x, x' the Fisher kernel is defined as

$$K(x, x') := s(x; \theta)^T I^{-1}(\theta) s(x'; \theta), \tag{2.11}$$

where $s(\cdot)$ and $I(\cdot)$ are referred to as the Fisher score and Fisher information matrix respectively. Moreover, θ denotes the parameters populating the underlying generative model.

The score is defined as

$$s(x; \theta) := \nabla_\theta \log P(x \mid \theta), \tag{2.12}$$

where $P(x \mid \theta)$ refers to the joint probability of each variable in x given the parameters of the underlying network. The Fisher information matrix is defined as

$$I(\theta) := \mathbf{E}_x[s(x; \theta) s(x; \theta)^T]. \tag{2.13}$$

Consequently, each matrix element is defined as

$$I_{ij}(\theta) := \mathbf{E}_x[s_i(x; \theta) \cdot s_j(x; \theta)] = \mathbf{E}_x \left[\frac{\partial \log P(x \mid \theta)}{\partial \theta_i} \frac{\partial \log P(x \mid \theta)}{\partial \theta_j} \right]. \tag{2.14}$$

Assuming some regularity conditions (interchangeability of derivative and integral) one can formulate an equivalent definition of the matrix elements [10]:

$$I_{ij}(\theta) := -\mathbf{E}_x \left[\frac{\partial s_i(x;\theta)}{\partial \theta_j} \right] = -\mathbf{E}_x \left[\frac{\partial^2 \log P(x \mid \theta)}{\partial \theta_i \partial \theta_j} \right]. \tag{2.15}$$

Both the score and the information matrix are computed using partial derivatives with respect to the parameters of the underlying model. This leads to the intuitive explanation that two vectors x and x' are considered similar if a small change of the parameters θ changes the probability of x and x' in the same way.

2.2.2 Hamming kernel

One existing kernel to compute similarities among data entries is based on the Hamming distance. Thus, throughout this thesis it will be referred to as the Hamming kernel. The Hamming distance was introduced in [11] and can be defined for two entries x and x' as

$$H(x, x') := \sum_{i=1}^{n} 1 - I(x_i, x'_i), \tag{2.16}$$

where $I(x, y)$ is the indicator function evaluating to 1 if $x = y$ and 0 otherwise.

The Hamming kernel is defined as

$$K(x, x') := 1 - \frac{H(x, x')}{n} = \frac{1}{n} \sum_{i=1}^{n} I(x, x'). \tag{2.17}$$

Intuitively it corresponds to the fraction of matching variables between the two entries. Hence, this kernel takes values between 0 and 1, where 0 indicates that not a single variable is matching and 1 corresponds to the entries being identical.

3
Methods

This chapter introduces the general setup of the evaluation. It establishes the techniques used to learn the Bayesian networks as well as the kernels that will be explored. Lastly, an overview of the investigated datasets is presented.

3.1 Evaluation overview

In order to address the research questions, the evaluation is divided into three parts. Firstly, the Fisher kernel's underlying Bayesian network is evaluated on the task of imputation for a variety of circumstances, where it is compared against naive imputation methods. Secondly, the Fisher kernel is evaluated on the task of clustering using two different methods for clustering. The performance is compared to that of naive kernels using the same methods. Thirdly, Bayesian network structure segments are analyzed in order to comprehend their impact on the similarities produced by the Fisher kernel. The remainder of this chapter presents the general method, whereas, specific details are left to each corresponding chapter.

3.2 Bayesian network

The choices when learning a Bayesian network from a dataset are plentiful, with a large variety of methods for both parameter- and structure learning. In addition, since missing values are going to be handled by the Bayesian network, an imputation technique has to be chosen.

Starting with the structure learning, the qNML score will be used to guide the search over possible structures. The algorithm being used to optimize the score can be described as a combination of structured exhaustive search coupled with dynamic programming [12]. It yields the optimal structure with respect to a scoring function and can be run in a feasible time for the purpose of this thesis.

With the structure in place, the parameter values are set using the cNML approach. This is a natural choice due to the values coinciding with the predictive probabilities of NML scoring functions. Using qNML for structure learning and cNML for parameter learning provides a method of learning the Bayesian network entirely free from any hyperparameters.

Lastly, the inference technique of the Bayesian network being used is the junction tree algorithm, which has the favorable property that it collapses to Mode when used with an empty network structure. It produces a probability distribution of each missing value, which can be used for imputation. Due to the limitations imposed

on the datasets, it is computationally feasible even though it is an exact inference technique.

Learning the Bayesian network is also coupled with some practical difficulties. Only complete data entries can be used for structure learning, since it is non-trivial to define a scoring function in the case of missing values. Parameter learning, on the other hand, is not limited to complete entries, which enables it to use the entire dataset.

The representation and learning of the Bayesian networks produced during the experiments rely on two GitHub-repositories. The first one is a repository for exact Bayesian network structure learning [13]. The second is a package for modeling a Bayesian network [14]. Together they provide modules for structure learning, parameter learning, inference and computing Fisher similarities.

3.3 Kernels

The kernels being focused on during all of the experiments are the Hamming kernel and the Fisher kernel. The Hamming kernel is used as a benchmark, since it is straight-forward and less complex than the Fisher kernel. In addition to these two, another kernel inspired by the Fisher kernel is investigated, which throughout this thesis is referred to as the scaled-Hamming kernel. This kernel does not handle dependencies between variables, but still weighs each variable similarly to the Fisher kernel.

3.3.1 Fisher kernel

The Fisher kernel derived for Bayesian networks can be decomposed into a sum of each variable's contribution to the final similarity. For the entries x and x' and an underlying Bayesian network the Fisher similarity can be specified as $K(x, x') = \sum_{i=1}^{n} K_i(x, x')$, where

$$K_i(x, x') := \begin{cases} 0 & \text{if } \pi_i(x) \neq \pi_i(x'), \\ -\frac{1}{P(\pi_i(x))} & \text{if } \pi_i(x) = \pi_i(x') \wedge (x_i \neq x'_i), \\ \frac{1}{P(\pi_i(x))} \cdot \frac{1-\theta_{i\pi_i(x)x_i}}{\theta_{i\pi_i(x)x_i}} & \text{if } \pi_i(x) = \pi_i(x') \wedge (x_i = x'_i). \end{cases} \tag{3.1}$$

3.3.2 Scaled-Hamming kernel

The Scaled-Hamming kernel corresponds to the Fisher Kernel when the Bayesian network is free from arcs. It can also be expressed in a similar fashion as the Hamming kernel, where each variable and value is weighted appropriately.

Formally, we define the Scaled-Hamming kernel as $K(x, x') = \sum_{i=1}^{n} K_i(x, x')$, where

$$K_i(x, x') := \begin{cases} -1 & \text{if } x_i \neq x'_i, \\ \frac{1-P(x_i)}{P(x_i)} & \text{if } x_i = x'_i. \end{cases} \tag{3.2}$$

In order to see that this corresponds to the Fisher kernel of a Bayesian network free from arcs, consider the cases in Eq. 3.1. Without any arcs, all the parent configurations are always considered to be equal, meaning the first case can never occur. Furthermore, $P(\pi_i(x)) = 1$, since there exist no variation of the parent configuration. Finally, $\theta_{i\pi_i(x)x_i} = P(x_i)$, since no conditional dependencies exist with any other variables.

3.4 Datasets

UCI [15] provides a large collection of datasets that are commonly used for evaluation within the field of machine learning. In this thesis, seven categorical datasets are used for evaluation and were chosen from the *UCI* classification datasets. The reason for considering only classification datasets is that labels are required for evaluation using supervised metrics.

The seven datasets provide a variety of challenges while still having a suitable number of variables and entries. Additionally, they consist of discrete data that may or may not be on a nominal scale, however, the data is always treated as being nominal. If the datasets initially contain missing values, those values are treated as an additional category. A short overview of each dataset is presented below, while some of their general properties are summarized in Table 3.1.

Table 3.1: General properties of the datasets.

	Instances	Attributes	Variable categories	Classes
Breast Cancer	286	9	2-11	2
Hayes-Roth	160	4	3-4	3
House Votes-84	435	16	3	2
Lymphography	148	18	2-8	4
Primary Tumor	339	17	2-4	22 (1 empty)
SPECT	267	22	2	2
Tic-Tac-Toe	958	9	3	2

Breast Cancer: This dataset consists of 9 attributes in the domain of breast cancer. The attributes are a combination of nominal and ordinal attributes with varying ranges of categories. Additionally, two of the attributes contains a few missing values. The instances are divided into two different classes of sizes 201 and 85.

Hayes-Roth: The *Hayes-Roth* dataset is divided into three separate classes of 65, 64 and 31 instances. The instances are described using 5 nominal attributes, one of which is ignored since it contains a unique identifier. Furthermore, the values of one of the attributes is completely random and should posses no predicitve information. It is also noteworthy that the dataset is free from missing values. The dataset is originally divided into train and test sets, however, the two sets were merged in order to treat this dataset as any other.

Congressional Voting Records: The *House Votes-84* dataset, which it is also called, consists of congressional voting records from 1984 for congressmen on 16 key votes. The attributes all describe if the individual congressman voted for or against

in a specific voting. The majority of the attributes contain missing values, which are intended as somewhere in between for or against. The dataset is divided into two classes, Democrats and Republicans, of sizes 267 and 168 respectively.

Lymphography: This dataset describes instances in the domain of lymphography using 18 attributes. Some of the attributes are ordinal while the majority is of nominal nature. All of the attributes are free from missing values. The dataset is divided into four classes that are significantly different in size: 2, 81, 61 and 4.

Primary Tumor: Using 17 attributes, this dataset describe 22 classes. The most prevalent classes contains 84 and 39 instances respectively. Additionally, there are five classes ranging between 20-29 instances, four classes between 10-16 instances, four classes between 6-9 instances, 6 classes with 1-2 instances and a single empty class. The empty class is ignored for clustering, reducing the number of clusters to 21. Most of the attributes are binary, one is ordinal and a few are nominal with multiple categories. Furthermore, two of the attributes contain a considerable amount of missing values, whereas another three attributes only contain a single missing value.

SPECT: The *SPECT* dataset describes Single Proton Emission Computed Tomography images in a condensed format, consisting of 22 binary attributes. The instances are arranged into two classes, normal and abnormal of size 55 and 212 respectively. The dataset is completely free from missing values. Originally this dataset is split into train and test sets, however, they are merged as to treat this dataset uniformly with the other datasets.

Tic-Tac-Toe: The *Tic-Tac-Toe* dataset describes all possible final board configurations of the game with the same name. Each of the 9 attributes model the state of a position on the board. Therefore, each variable can take one of three values. Additionally, there are no missing values in the dataset. The instances are grouped into classes based on the outcome of the game. The size of those classes are 626 and 332 respectively.

4

Imputation

This chapter explores the task of imputation. In the beginning of the chapter, the task is described and common methods for addressing it are presented. Additionally, the chapter explains the setup of the experiments as well as the results. It ends with an analysis of the experimental results.

4.1 Task description and approach

Imputation is the task of completing an originally incomplete dataset, or in other words to predict all of the missing values. In real world application no ground truth generally exists for this task. However, this controlled setting allows access to the ground truth for evaluation. There exist a variety of methods to impute missing values in categorical datasets. Although, in addition to inference using a Bayesian network, only Mode and weighted k-Nearest Neighbors are considered.

4.1.1 Mode

When using numerical variables, a naive method of imputation is to use the mean or median value of the variable throughout the dataset. A similar method for categorical variables is to simply use the category that occurs most frequently, i.e., the mode. This method can be slightly refined by using the relative frequency for each category as its probability. Thus, producing a prediction in the form of a probability distribution. Predictions produced this way will assign the highest probability to the mode of the variable and lower probabilities to less frequent categories.

4.1.2 Weighted k-Nearest Neighbors

The key concept of k-Nearest Neighbors classification is that close neighbors of a data entry provide useful information about its class belonging [16]. This concept can be applied to any categorical variable of interest. When trying to impute a missing value of entry x, the first step of the algorithm is to find its k closest neighbors, according to some similarity or distance-measure, e.g., the Hamming distance. The next step is to simply use the distribution of categories among the neighbors as the prediction of the missing values of x.

The simplest way of forming the distributions is to assign equal weights to all neighbors. A more sophisticated approach, which can be useful in certain situations, is to apply different weights to each neighbor. This can be done in several ways,

but the most commonly used method is to weigh the neighbors proportionally to the similarity or distance to x [17]. Another approach is to weigh each category inversely proportional to its relative frequency in the entire dataset [18]. The latter approach gives a higher weight to rare categories, which prevents them from being drowned by common ones.

In short this approach can be seen as using a weighted variant of Mode, limited to a neighborhood of x. One major difference to Mode is that all of the variables affect the outcome of the prediction, since only a neighborhood of x is considered. Mode on the other hand traditionally only considers each variable in isolation.

4.2 Experimental setup

The setup for evaluating Bayesian networks' ability to impute missing values consists of three steps. Firstly, missingness is induced in the complete datasets. This is followed by imputation using the different methods. Lastly, the predictions are evaluated using supervised metrics.

The two naive methods of using Mode and weighted k-Nearest Neighbors (wKNN) to impute missing values are used as benchmarks. For wKNN, k is set to 3 and the weights are inversely proportional to the distribution of the categories. Neither of these methods require complete data entries, which allow them to consider the entire dataset at once. This could be considered biased, since the information used by the different models varies. However, given that the purpose for the evaluation is to impute missing values in an incomplete dataset, each algorithm is allowed as much information as possible.

4.2.1 Inducing missingness

The process of inducing missingness is conducted in two steps. Firstly, the dataset is divided into train- and test sets using a specified train/test-ratio. The split is achieved by uniformly selecting the exact number of data entries that corresponds to the given ratio. The main reason for the train/test-split is to guarantee that the dataset contains complete entries, since the structure learning of Bayesian networks is strictly reliant on them. Secondly, given another ratio of the desired level of missingness, the corresponding number of values in the test set are labeled as missing. Each value in the test set is chosen with equal probability to be labeled as missing. In short, the entire setup can be described as a train/test-split, where values are missing completely at random (MCAR) in the test set.

4.2.2 Augmented models

Bayesian networks are heavily dependent on the available complete data entries when learning the network structure. Therefore, naive imputation methods are used as a preprocessing step before learning the Bayesian network. The Bayesian network is then used to predict the missing values of the original dataset once again. Augmenting the dataset before learning the network might prove useful for datasets with few complete data entries.

4.2.3 Prediction metrics

Since the ground truth of all missing values are known, it is used in order to compute the accuracy and cross-entropy loss for all methods. Below they are defined for a single prediction, whereas in reality they are commonly reported as an average over the total number of predictions. The probability distribution of a prediction is denoted by $\hat{y}$ and the one-hot representation of the ground truth by y.

Accuracy: A prediction is considered accurate if the value predicted to have the highest probability corresponds to the ground truth. Hence, the accuracy is defined as

$$a(\hat{y}, y) := \begin{cases} 1, & \text{if } \operatorname{argmax} \hat{y} = \operatorname{argmax} y, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Cross-entropy loss: The cross-entropy loss is a metric that measures the confidence levels of the predictions. It does so by rewarding highly confident accurate predictions, while punishing confident inaccurate predictions. Formally, the cross-entropy loss is defined as

$$L(\hat{y}, y) := -\sum_i y_i \log \hat{y}_i. \tag{4.2}$$

In the implementation, the cross-entropy loss function is capped such that $L(\hat{y}, y) \leq \log 10^{-5}$.

4.3 Results

The results are divided into three different comparisons: Traditional methods, naive methods and their augmented models, and finally the overall best method. The experiments have been conducted using different ratios of missing values. The settings used when inducing missingness, i.e., the train/test ratio and the ratio of missingness, both ranges from 0.1 to 0.5 with incremental steps of 0.1. This corresponds to 1-25% of missing values in the entire dataset, although restricted to the current test set.

Results are presented in the form of heatmaps over the different settings used to induce missingness. Additionally, macro averages are presented in an attempt to summarize the results for each model and dataset. The macro average corresponds to averaging the accuracy over all settings, giving equal weight to each instance, even though the number of actual predictions varies.

4.3.1 Traditional methods

The accuracy of the traditional methods Bayesian network (BN), wKNN and Mode are compared on the task of imputing missing values in Table 4.1. Each value corresponds to the macro average over all settings for a specific method and dataset. The best performing method of each dataset is highlighted and it is obvious from the

results that the superior method in this context is the Bayesian network. In general it achieves an accuracy a few percentage points above the second best method. These results strengthen the idea that Bayesian networks are well suited for imputing missing values.

Table 4.1: Macro average of accuracy on different datasets for the traditional methods. Each average is taken over 250 runs and consists of 25 settings: Test ratio 0.1-0.5 and ratio of missingness 0.1-0.5.

	BN	Mode	wKNN
Breast Cancer	**0.584**	0.523	0.476
Hayes-Roth	**0.355**	0.307	0.295
House Votes-84	**0.744**	0.525	0.684
Lymphography	**0.647**	0.603	0.571
Primary Tumor	**0.768**	0.751	0.607
SPECT	**0.790**	0.683	0.703
Tic-Tac-Toe	**0.476**	0.423	0.314

The impact of the settings on the different methods can further be visualized. The accuracy for all variations of test ratio and ratio of missingness are presented in Fig. 4.1 for the dataset *SPECT*. As one can suspect, the Bayesian network tend to perform worse when the number of missing values in the dataset is increased. This is a general behavior for all of the investigated datasets. The trend of Mode appears to be unpredictable, which most likely follows from the value distribution of each variable being heavily dependent on which values are labeled as missing. In the case of wKNN, one can notice that the accuracy depends more on the ratio of missingness than on the test ratio. A partial explanation is that an increasing ratio of missingness reduces both the information content of the neighbors and the search space of possible neighbors.

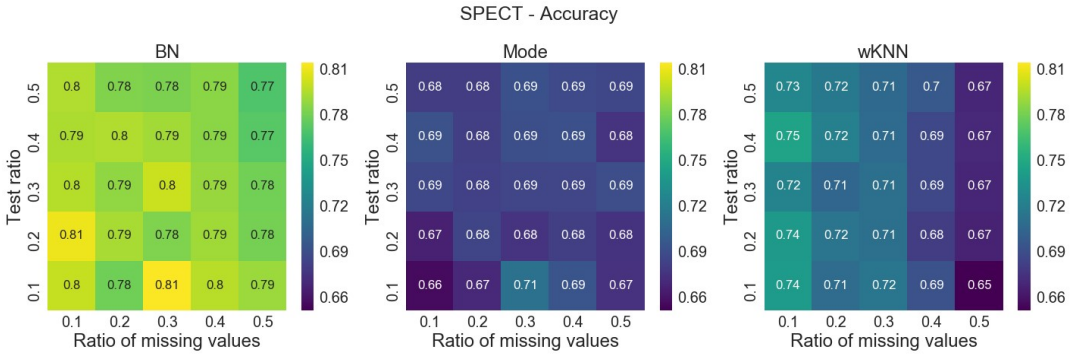

Figure 4.1: Accuracy of imputation for traditional methods, using different settings of missingness. The results are for the dataset *SPECT* where the values are averages over 10 runs for each setting.

Another detail worth noticing in Fig. 4.1 is that the Bayesian network is not only better for some choice of settings, but in fact for every single one. This is the general trend among all of the investigated datasets, although there exist a few exceptions for certain combinations of settings and datasets. This is a good indication that the

averages presented in Table 4.1 are representative for the model in general, and that it is not a product of a single setting drastically changing the average.

4.3.2 Naive methods and augmented models

As reported in the previous section, the Bayesian network is superior to the naive methods for imputing missing values. However, in contrast to the naive methods, the Bayesian network suffers from not being able to fully use incomplete data entries. Thus, two augmented models have been investigated: BN-wKNN and BN-Mode, where wKNN and Mode are used to augment the data.

Using the same example as in the previous section, i.e., the *SPECT* dataset, heatmaps for the naive methods and their augmented models are presented in Fig. 4.2. From this it is apparent that the Bayesian network does not seem to suffer significantly from the possibly incorrect imputed values. The results are improved regardless of the train/test ratio and the ratio of missingness for both of the naive methods. The additional predictive capability of the Bayesian network appears to prevail the uncertainty of the naive imputations.

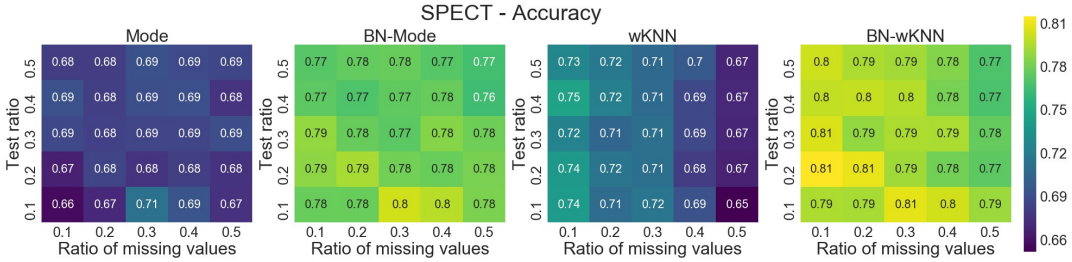

Figure 4.2: Accuracy of imputation for naive methods, and their augmented models, using different settings of missingness. The results are for the dataset *SPECT* where the values are averages over 10 runs for each setting.

Furthermore, by looking at the macro averages presented in Table 4.2, it can be noticed that applying a Bayesian network on top of the naive methods improves the results significantly. The overall best accuracy is obtained by using wKNN to augment the data, although the results for BN-wKNN and BN-Mode yields similar results in most cases.

Table 4.2: Macro average of accuracy on different datasets for the naive methods and their augmented models. Each average is taken over 250 runs and consists of 25 settings: Test ratio 0.1-0.5 and ratio of missingness 0.1-0.5.

	BN-Mode	Mode	BN-wKNN	wKNN
Breast Cancer	0.581	0.523	**0.582**	0.476
Hayes-Roth	0.331	0.307	**0.378**	0.295
House Votes-84	0.737	0.525	**0.749**	0.684
Lymphography	0.640	0.603	**0.641**	0.571
Primary Tumor	**0.771**	0.751	0.763	0.607
SPECT	0.779	0.683	**0.792**	0.703
Tic-Tac-Toe	0.473	0.423	**0.491**	0.314

4.3.3 Overall evaluation

Due to the results in the previous sections, the most interesting case to investigate is how the performance of the augmented models compare to that of the traditional Bayesian network. Initially, the already separately presented macro averages for all methods are collected in Table 4.3. Apparent is that there is no obvious superior approach among the augmented models and the traditional Bayesian network when averaging over all of the settings. Although, BN-wKNN appears to be the best choice with a small margin.

Table 4.3: Macro average of accuracy on different datasets for all methods. Each average is taken over 250 runs and consists of 25 settings: Test ratio 0.1-0.5 and ratio of missingness 0.1-0.5.

	BN	BN-Mode	BN-wKNN	Mode	wKNN
Breast Cancer	**0.584**	0.581	0.582	0.523	0.476
Hayes-Roth	0.355	0.331	**0.378**	0.307	0.295
House Votes-84	0.744	0.737	**0.749**	0.525	0.684
Lymphography	**0.647**	0.640	0.641	0.603	0.571
Primary Tumor	0.768	**0.771**	0.763	0.751	0.607
SPECT	0.790	0.779	**0.792**	0.683	0.703
Tic-Tac-Toe	0.476	0.473	**0.491**	0.423	0.314

In order to better understand the predictions made by the various methods, the macro average of the cross-entropy loss are presented in Table 4.4. The first thing to notice is the poor performance of wKNN with respect to this metric. This can be explained by the fact that the method only distributes its predictions over values of three neighbors, i.e., giving all other possible values a probability of zero. Secondly, one can notice that the Bayesian network and the augmented models perform similarly in terms of cross-entropy loss. However, in contrast to the accuracy metric, the trend is instead that the traditional Bayesian network is the best performing method with a small margin. A possible explanation is that augmenting the data can have a negative impact on the predictive distribution by confirming biases.

Table 4.4: Macro averages of cross-entropy loss on different datasets for all methods. Each average is taken over 250 runs and consists of 25 settings: Test ratio 0.1-0.5 and ratio of missingness 0.1-0.5.

	BN	BN-Mode	BN-wKNN	Mode	wKNN
Breast Cancer	**0.967**	0.975	0.975	1.075	3.337
Hayes-Roth	1.257	1.265	1.312	**1.218**	4.480
House Votes-84	0.605	0.609	**0.601**	0.834	1.772
Lymphography	**0.793**	0.799	0.812	0.839	2.282
Primary Tumor	0.531	**0.530**	0.539	0.570	1.712
SPECT	**0.493**	0.507	0.515	0.604	1.246
Tic-Tac-Toe	**1.008**	1.030	1.010	1.068	5.104

In addition to the results in Table 4.3 and Table 4.4, the corresponding standard deviations are presented in Appendix B.1. In general, there is no significant difference between the methods. Although, Mode has a slightly lower standard deviation than the other methods and wKNN suffers from only considering three neighbors.

This chapter is concluded with a brief reflection on the rationality of the results. The Bayesian network outperforms uninformed guessing in terms of accuracy in all of the datasets. In fact, it does so by quite a large margin in most of the datasets, with the single exception being *Hayes-Roth*. An explanation for this could be that the dataset only contains four variables, one of which is entirely random. Thus, finding correlations between the variables is especially challenging in this dataset.

5
Clustering

In this chapter the task of clustering is explored. It begins with a description of the task and two kernelized algorithms commonly used for clustering. Furthermore, the chapter explains the setup of the experiments as well as the evaluation metrics. It ends with a presentation and analysis of the experimental results.

5.1 Task description and approach

Clustering is the general task of separating a set of objects into groups, such that objects within the same group are more similar to each other than to objects from another group. The groups that objects are being assigned to are referred to as clusters. The task is unsupervised in nature, since no information regarding the label of an entry is provided. In the general formulation, not even the number of clusters are known beforehand.

The task is to divide a given dataset D of size $N \times n$, where each row is a realization of n variables, into an unknown number of clusters K. This should be done in such a way that objects belonging to the same clusters are considered similar in some sense. Formally, denoting a subset of rows in D as C_p, the task is to find the optimal K and $C = \{C_1, \ldots, C_K\}$ satisfying

$$D = \bigcup_{p=1}^{K} C_p, \text{ such that } C_p \cap C_{p'} = \emptyset \ \forall \ p, p'. \tag{5.1}$$

There exists a wide variety of different clustering algorithms trying to solve the clustering task specified above, or variants thereof. In this thesis, two different methods are considered: Correlation clustering and Spectral clustering. Correlation clustering is selected for its ability to handle negative similarity values. Furthermore, Spectral clustering is selected because it is considered a reliable clustering algorithm.

5.1.1 Correlation clustering

Correlation clustering is a graph based clustering method that considers the problem of partitioning a graph into K connected components, i.e., clusters. Thus, the dataset is represented by a fully connected graph, where each entry is a vertex and the pairwise similarities are the edge weights.

The Correlation clustering cost function was originally presented for edges with weights ± 1, but was later generalized to arbitrary values [19]. The cost function aims to minimize the sum of positive inter-cluster edge weights plus the sum of negative

intra-cluster edge weights. Intuitively, this corresponds to penalizing similarities between objects in different clusters, as well as dissimilarities between objects in the same cluster.

Let X_{ij} be the similarity between entry i and j, according to some arbitrary kernel function. The cost function of Correlation clustering can then formally be defined as

$$\text{cost}_{\text{cc}}(C, D) := \frac{1}{2}\sum_{p=1}^{K}\sum_{i,j\in C_p}(|X_{ij}| - X_{ij}) + \frac{1}{2}\sum_{p=1}^{K}\sum_{p'\neq p}\sum_{i\in C_p}\sum_{j\in C_{p'}}(|X_{ij}| + X_{ij}). \quad (5.2)$$

In order to avoid singleton clusters it has been proposed to use shifted similarity values [20]. The proposed *adaptive shift* works by shifting similarities such that the similarity X_{ij} is shifted with respect to the similarities of both i and j. The suggested shift ensures that the sum of the similarities between i and all other objects equals zero, and likewise for j. A local search algorithm for minimizing the cost function has also been reported to produce better results than other approximate optimization methods [20].

5.1.2 Spectral clustering

Spectral clustering is a clustering method originating from spectral graph theory [21]. Rather than directly clustering on the dataset, it creates a spectral embedding of the data entries, which later can be used in any standard clustering algorithm, e.g., K-means.

The spectral embedding are created from the original similarity matrix, X. The first step is to form the Laplacian of this similarity matrix. Secondly, the K eigenvectors corresponding to the K largest eigenvalues are chosen as an embedding of the entries. Finally, after some re-normalization, the embeddings are used to cluster the original data entries into K clusters using a standard clustering algorithm. A full description and analysis of the algorithm is available in [21].

5.2 Experimental setup

Only the easier clustering problem is considered, which means that the number of clusters are given as a part of the problem formulation. This is done in order to focus the evaluation on how the clusters are formed, rather than on the amount of clusters. If the harder problem was considered instead, one would have to do a search over different values of K, which would be both time consuming and of little interest to the evaluation.

The clustering experiments are conducted using both Correlation clustering and Spectral clustering. Both of the algorithms are combined with each kernel among Hamming, scaled-Hamming and Fisher for Bayesian networks.

The implementation of the Spectral clustering algorithm used in the experiments is from *Scikit-learn* [22], together with default parameter values. The algorithm is using K-means to cluster the embeddings, which is randomly reinitialized 10 times. Spectral clustering is incompatible with negative similarity values, meaning that all

similarities has to be shifted. In the presence of negative similarities, this is done by simply subtracting the smallest similarity value.

The Correlation clustering algorithm chosen for the experiments is presented in [20]. The algorithm uses a local search in order to optimize the cost function. The local search is selected to be reinitialized 25 times for each execution of the clustering algorithm. When using this implementation in combination with the Hamming kernel or the scaled-Hamming kernel, the similarities are shifted using the suggested *adaptive-shift*. However, when using Correlation clustering together with the Fisher kernel, no such shift is needed since the similarities produced are already shifted by design.

5.2.1 Clustering metrics

At the time of writing, there is no prevalent best practice for evaluating clustering performance, even less so in a categorical setting. However, the evaluation approach can be either unsupervised or supervised. Supervised evaluation assumes that the true labels of each entry are known, whereas the unsupervised does not. In general, the unsupervised metrics depend on different norms, while the supervised metrics consider pairwise frequencies. The evaluation will be performed in a supervised setting, meaning that only such metrics will be presented here. The supervised metrics being used are *adjusted Rand index*, *adjusted mutual information* and *V-measure*, which are described shortly below. The metrics are not supervised by definition, since they are defined for two arbitrary clusterings. However, by assigning the true labels as one of the clusterings the metrics effectively become supervised. Formal definitions for the metrics can be found in Appendix C.1.

Adjusted Rand index [23]: Intuitively, one can think of the *adjusted Rand index* as a score that measures the agreement between two clusterings. The agreement is measured by counting the number of times that the two clusterings agree that both entries either belong to the same cluster, or that they belong to different ones. The score lies in the range $[-1.0, 1.0]$, where 1.0 represents a perfect match and random clusterings have a value close to zero.

Adjusted mutual information [24]: The mutual information metric measures how the information of one of the clusterings influence the certainty of the other. It attempts to quantify how much information is available about one clustering given the other. Once again, a perfect match yields a score of 1.0 and a random assignment to clusters has an expected value close to zero.

V-measure [25]: The *V-measure* is a metric that returns the harmonic mean of the completeness and homogeneity of a clustering solution. Both completeness and homogeneity refers to desirable properties of a clustering solution. Completeness is that all members of a single class also belong to the same cluster, whereas homogeneity is that each cluster only contains members from a single class. The score lies in the range $[0.0, 1.0]$ and, consistent with the previous metrics, a score of 1.0 corresponds to a perfect match.

5.3 Results

The average *adjusted Rand index* over 10 runs is presented in Table 5.1 for each algorithm-kernel combination. From the results it can be noticed that using the Fisher kernel appears to be a poor choice, regardless of which clustering technique is used. This is at least the case for the evaluated datasets when comparing to the true labeling of the entries. The same trend is prevalent when evaluating the performance using *adjusted mutual information* or *V-measure*, which can be seen in Appendix C.2.

Table 5.1: *Adjusted Rand index* averaged over 10 runs for different algorithm-kernel combinations. Abbreviations: Correlation clustering (CC), Spectral clustering (SC), Fisher kernel (F), Hamming kernel (H), scaled-Hamming kernel (SH).

	CC-F	CC-H	CC-SH	SC-F	SC-H	SC-SH
Breast Cancer	0,122	-0,003	0,147	0,006	0,004	**0,162**
Hayes-Roth	**0,082**	-0,007	**0,082**	-0,009	-0,011	-0,009
House Votes-84	0,077	0,557	0,557	0,006	**0,564**	**0,564**
Lymphography	0,105	0,188	0,242	-0,018	0,258	**0,259**
Primary Tumor	0,035	0,081	**0,123**	0,076	0,083	0,076
SPECT	0,004	0,055	**0,057**	0,048	-0,061	-0,029
Tic-Tac-Toe	0,005	0,000	**0,020**	0,007	-0,001	-0,001

The Fisher kernel could perform poorly due to a number of factors:

- The dataset is not suitable for clustering in general, i.e., the labels are needed to find the desired structure in the dataset.
- There might exist a more natural clustering of the data than the one suggested by the labels.
- The labels are organized relying on similarities among dependent variables.

The most likely conclusion is that the datasets *Hayes-Roth*, *Tic-Tac-Toe*, and *SPECT* belongs to the first or second category, since none of the methods appears to find reasonable clusterings according to the true label for these cases. The more interesting datasets are the remaining ones, where a clear difference can be noticed between the investigated kernels. It appears that the best clustering in general is achieved using either the scaled-Hamming or the Hamming kernel. In common for these kernels is that they do not consider correlations between variables.

Shifting the focus towards the scaled-Hamming kernel, it is worth noticing that the actual weighing of each variable independently appears to work well. In fact, looking at Table 5.1, the weighing actually improves the results in most cases. Specifically, in combination with Correlation clustering the scaled-Hamming kernel either performs best, or reaches results similar to the best performing model. In sharp contrast, the performance drops significantly when arcs and dependencies are used with the Fisher kernel.

Even though no method produces suitable clusters for *Hayes-Roth*, it is still interesting since it is the only dataset where the Fisher kernel is competitive. However, when inspecting the network structure being used by the Fisher kernel for the

dataset, it is empty. Thus, the Fisher kernel has been reduced to scaled-Hamming and the only difference in performance is due to random initialisation in the clustering algorithms.

At the other end of the spectrum is the *House Votes-84* dataset, for which all methods except Fisher yields good results. The specific network structure used by the Fisher kernel for *House Votes-84* contains 26 arcs, and every variable has at least either an incoming or an outgoing arc. This complex structure proved useful during imputation, however, when used for producing similarities it seems to impede the performance.

A complex structure also has the property of producing more extreme similarity values for the Fisher kernel, whereas the other kernels are non-dependent on any underlying structure. The Hamming kernel only produces similarities in the range $[0, 1]$. With the scaled-Hamming kernel, the contribution of each variable is upper bounded by the inverse of individual variable distributions and lower bounded by -1. With the Fisher kernel, on the other hand, a complex network structure allows the contribution of a single variable to be both upper and lower bounded by the inverse of joint distributions of many variables. As a consequence, the similarity values of the Fisher kernel is distributed over a wider range of values. Looking at the *House Votes-84* dataset, one can see how the learnt structure influences the range of similarity values between entries. The scaled-Hamming kernel ranges from -16.0 to 396.6, whereas the Fisher kernel ranges from -75.3 to 759.9. The wider range most likely influences the clustering performance of the kernel-algorithm combinations.

In general, one can argue that the Fisher similarities are highly dependent on the actual structure of the Bayesian network. Thus, the structure used for imputation is not necessarily a good choice on the task of clustering. However, it is still possible that the same structure can be used to distinguish objects on the easier task of classification, when the similarities can be related to the actual true class labeling. In the light of this insight, a natural question is what network structure is suitable for the Fisher kernel in different tasks. Even more interesting is how the produced similarities are affected by the structure.

6

Bayesian Network Structure Segments

This chapter focuses on network structure segments and their influence on the Fisher similarities. The similarities are analyzed from a theoretical as well as a practical point of view. The cases that are being investigated are single-arc networks, two-arc networks and fully connected networks.

6.1 Theoretical exploration

The clustering results indicated that arcs in the Bayesian network structure prevented the Fisher kernel from finding the true clusters. This follows from the fact that the scaled-Hamming kernel produced competitive results, whereas the Fisher kernel did not. As a consequence of this, the same similarities were also briefly evaluated on the task of classification without any success, which is reported in Appendix D. Hence, since no definitive use of the existing structure has been found, the basic structural building blocks of the network are analysed theoretically. The case being investigated first is a network structure consisting of a single arc. Next, a similar investigation is carried out for the cases involving two arcs. Finally, since the empty network is considered, the other edge case of a fully connected structure is explored.

6.1.1 Structure using a single arc

Consider two variables in a dataset, u and v, which can take on an unspecified number of values. Furthermore, consider two arbitrary entries. The first entry takes the values $u = u_1$ and $v = v_1$, with $P(u = u_1) = \alpha$ and $P(v = v_1) = \beta$ being their respective probability. The second entry takes the values u_2 and v_2. Consequently, when computing the collective contribution of the two variables to the similarity score, there are four different cases to consider:

$$\begin{aligned} &\text{a) } u_1 = u_2 \wedge v_1 = v_2 \qquad &&\text{b) } u_1 = u_2 \wedge v_1 \neq v_2 \\ &\text{c) } u_1 \neq u_2 \wedge v_1 = v_2 \qquad &&\text{d) } u_1 \neq u_2 \wedge v_1 \neq v_2 \end{aligned}$$

First, consider the four cases for an empty network, i.e., with no arcs at all, which corresponds to the scaled-Hamming kernel. The joint contribution of the two

variables to the similarity score in the four cases are:

$$\text{a) } \frac{1}{\alpha}+\frac{1}{\beta}-2 \qquad \text{b) } \frac{1}{\alpha}-2 \qquad \text{c) } \frac{1}{\beta}-2 \qquad \text{d) } -2.$$

As can be seen from these cases, the general pattern is that each matching variable contributes with the inverse of its probability. Additionally, the contribution of each variable is offset with -1, regardless of whether it is matching or not. A property worth noticing is that a fully anticipated match, e.g., $\alpha = 1$, will yield a similarity contribution of 0 for that variable. This means that fully anticipated matches will only cancel out the offset, thus, not contributing at all to the similarity score.

After introducing an arc $u \to v$ with the conditional probability $P(v = v_1 \mid u = u_1) = \varphi$, the combined contribution of u and v are:

$$\text{a) } \frac{1}{\alpha\varphi}-1 \qquad \text{b) } -1 \qquad \text{c) } -1 \qquad \text{d) } -1.$$

In this case it is only when both variables match that yields a positive contribution, while all other combinations yield a score of -1. Thus, adding a single arc between two variables can be seen as a way of neglecting a variable on its own and only considering it in combination with another one.

In fact, one can choose to look at both of the variables as one collapsed variable, where a match corresponds to both of the original variables matching. When matching, this collapsed variable yields a similarity that depends on the probability of the configuration, rather than each variable individually. On the other hand, when mismatching, it only counts as a single mismatching variable.

6.1.2 Structure using two arcs

Considering a network with two arcs, the arcs can either have a variable in common or not. If they do not have a variable in common, it is simply two separate instances of the already explored case of a single arc. In this case one can follow the procedure in the previous section. The interesting case is when the arcs are connected in some way. All three structures where this could happen are explored, but first the necessary notation is introduced.

Consider three variables u, v and w, which can take an unspecified number of values. Additionally, consider two entries. The first entry is given by $u = u_1, v = v_1, w = w_1$, where the probabilities are given as $P(u = u_1) = \alpha$, $P(v = v_1) = \beta$ and $P(w = w_1) = \gamma$. The second entry is given by $u = u_2, v = v_2, w = w_2$. This gives a total of eight different cases to consider:

$$\begin{aligned}
&\text{a) } u_1 = u_2 \wedge v_1 = v_2 \wedge w_1 = w_2 && \text{b) } u_1 = u_2 \wedge v_1 = v_2 \wedge w_1 \neq w_2 \\
&\text{c) } u_1 = u_2 \wedge v_1 \neq v_2 \wedge w_1 = w_2 && \text{d) } u_1 \neq u_2 \wedge v_1 = v_2 \wedge w_1 = w_2 \\
&\text{e) } u_1 = u_2 \wedge v_1 \neq v_2 \wedge w_1 \neq w_2 && \text{f) } u_1 \neq u_2 \wedge v_1 = v_2 \wedge w_1 \neq w_2 \\
&\text{g) } u_1 \neq u_2 \wedge v_1 \neq v_2 \wedge w_1 = w_2 && \text{h) } u_1 \neq u_2 \wedge v_1 \neq v_2 \wedge w_1 \neq w_2
\end{aligned}$$

The collective contribution to the similarity score of an empty Bayesian network is given by:

a) $\frac{1}{\alpha}+\frac{1}{\beta}+\frac{1}{\gamma}-3$ b) $\frac{1}{\alpha}+\frac{1}{\beta}-3$ c) $\frac{1}{\alpha}+\frac{1}{\gamma}-3$ d) $\frac{1}{\beta}+\frac{1}{\gamma}-3$

e) $\frac{1}{\alpha}-3$ f) $\frac{1}{\beta}-3$ g) $\frac{1}{\gamma}-3$ h) -3

This is what one would expect. Each match contributes individually according to its probability and a mismatch simply means that this is not counted. The remainder of this section is dedicated to the three different structures where two arcs are connected: The V-structure, the A-structure and the Chain structure.

6.1.2.1 V-structure

The first structure that is considered is the V-structure, which is displayed in Fig. 6.1. The structure consists of two variables u and v that has a child w in common.

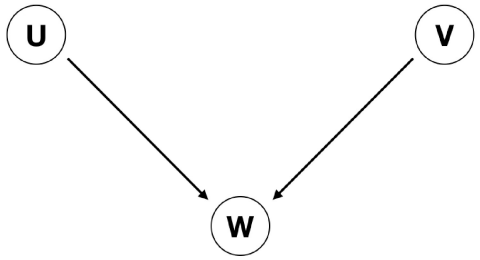

Figure 6.1: Illustration of the V-structure, consisting of the arcs $u \to w$ and $v \to w$.

In order to compute the similarities that this structure yields some notation is necessary. Using $P(u = u_1, v = v_1) = \delta$ and $P(w = w_1 \mid u = u_1, v = v_1) = \varphi$, the similarity score in the eight different cases is computed as:

a) $\frac{1}{\alpha}+\frac{1}{\beta}-2+\frac{1}{\delta\varphi}-\frac{1}{\delta}$ b) $\frac{1}{\alpha}+\frac{1}{\beta}-2-\frac{1}{\delta}$ c) $\frac{1}{\alpha}-2$ d) $\frac{1}{\beta}-2$

e) $\frac{1}{\alpha}-2$ f) $\frac{1}{\beta}-2$ g) -2 h) -2

It can be noticed that this no longer corresponds to collapsing any of the variables. Thus, it is far more difficult to provide an intuitive explanation of how the similarity score behaves in the different cases.

Matching both u and v includes the negative term $-1/\delta$, regardless if w is matched or not. However, this term is compensated for by adding the term $1/\delta\varphi$ whenever w is matched. In the case when w is not matched though, this could potentially result in a large negative similarity Another interesting property worth noticing is that whenever one of u and v are mismatched, the contribution of w is negated. This leads to a number of cases that are equal according to the similarity score: c) equals e), d) equals f), and g) equals h). Other than this, not much can be stated about the similarities of the V-structure.

6.1.2.2 A-structure

The second structure that is considered is the A-structure, which is shown in Fig. 6.2. This structure corresponds to the case where two variables have a parent in common. The parent is denoted as u and the children as v and w.

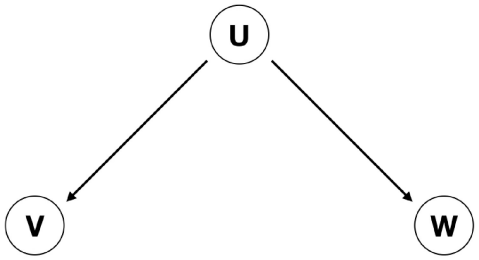

Figure 6.2: Illustration of the A-structure, consisting of the arcs $u \to v$ and $u \to w$.

Using the probabilities $P(v = v_1 \mid u = u_1) = \phi$ and $P(w = w_1 \mid u = u_1) = \theta$, the joint contribution to the similarity score is computed as:

a)	$\frac{1}{\alpha\phi} + \frac{1}{\alpha\theta} - \frac{1}{\alpha} - 1$	b)	$\frac{1}{\alpha\phi} - \frac{1}{\alpha} - 1$	c)	$\frac{1}{\alpha\theta} - \frac{1}{\alpha} - 1$	d)	-1
e)	$-\frac{1}{\alpha} - 1$	f)	-1	g)	-1	h)	-1

This case is somewhat more intuitive than the V-structure, but still difficult to fully grasp. The negative term $-1/\alpha$ appears whenever u matches, but it is compensated for by either $1/\alpha\phi$ or $1/\alpha\theta$ in all cases but one. The case e), where only u matches, is the only exception. This implies that very large negative similarities might appear in this specific case. In fact, a higher similarity score is always obtained if no variables are matched than if only u matches. Apart from this, the A-structure behaves in a similar way as if treating the arcs $u \to v$ and $u \to w$ as two separate single arcs.

6.1.2.3 Chain structure

The chain structure is the last structure that is considered. It is shown in Fig. 6.3 and consists of a chain of arcs. The specific choice of order here in comparison to the previous structures might appear confusing, but will become apparent once the similarities are computed.

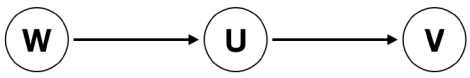

Figure 6.3: Illustration of the chain structure, consisting of the arcs $w \to u$ and $u \to v$.

The probabilities $P(v = v_1 \mid u = u_1) = \phi$ and $P(u = u_1 \mid w = w_1) = \psi$ are used to compute the similarity score:

a) $\frac{1}{\alpha\phi} + \frac{1}{\gamma\psi} - \frac{1}{\alpha} - 1$ b) $\frac{1}{\alpha\phi} - \frac{1}{\alpha} - 1$ c) $\frac{1}{\gamma\psi} - \frac{1}{\alpha} - 1$ d) -1

e) $-\frac{1}{\alpha} - 1$ f) -1 g) -1 h) -1

These expressions are similar to the ones obtained in the A-structure, with the only difference being terms including $\gamma\psi$ instead of $\alpha\theta$. This is no coincidence. Using that $\gamma\psi = P(w = w_1)P(u = u_1 \mid w = w_1) = P(u = u_1,\ w = w_1)$ and $\alpha\theta = P(u = u_1)P(w = w_1 \mid u = u_1) = P(u = u_1,\ w = w_1)$, it can be shown that these two structures actually are equivalent. Thus, any chain structure can be rewritten in terms of an A-structure using a different ordering, and vice versa.

6.1.3 Fully connected DAG structure

A fully connected structure is when the structure contains the maximal amount of arcs, while still retaining the DAG property. One of the simplest such structures has already been explored in the case of one arc and two variables, since no other arc can be added to that structure without violating the DAG property.

In the case of a single arc, it was discussed that the collective contribution to the similarity score corresponded to that of a single collapsed variable. It turns out that this is a general behavior for fully connected DAG structures, regardless of the number of variables.

Claim: A Bayesian network structure that is a fully connected DAG yields a Fisher similarity equal to that of a single collapsed variable.

The proof to this claim is found in Appendix E.

An empty network considers each variable on their own, while a fully connected DAG treats all variables as a single collapsed variable. This presents two well behaved edge cases, indicating that other possible structures represents something in between. The most obvious case is whenever a subset of variables are fully connected among themselves. Such a fully connected segment is collapsed and will be treated as a single variable, although the similarities considering all other variables remain unchanged.

6.2 Experimental setup

In order to investigate the possible benefits, or disadvantages, of including arcs in the network, network structures consisting of one and two arcs are evaluated on the task of clustering. The network structures being evaluated for one arc is found using exhaustive search over all possible arcs, starting with an empty Bayesian network. The arc yielding the best result is the one being used for comparison. The structures of two arcs are found by continuing the process by adding the second arc in a greedy manner. It is worth mentioning that the process of finding the

structures is supervised and, as such, provides no aid in finding the structures in an unsupervised setting. Thus, only an investigation regarding the potential of such structures is presented, whereas, an unsupervised method of finding them is not.

The setup when evaluating clustering is in general the same as the one presented in Section 5.2. The difference is that only Correlation clustering and the *adjusted Rand index* is considered when evaluating this new network. In addition, correlations between variables and the class label is used to understand how the added arc influences the similarities. Two metrics for evaluation are considered: *Uncertainty coefficient* (*Theil's U*) [26] and *Cramér's V* [27]. Both metrics produce results in the interval $[0.0, 1.0]$ for two entries, where higher values indicate a higher correlation. One major difference is that *Cramér's V* is symmetric, whereas *Theil's U* is not. These metrics are only considered for networks with a single arc, since there is no intuitive way of collapsing the other structures into a single variable.

6.3 Results

This section is divided into two parts. The first part considers the case of a single arc and the second part extends the evaluation to two arcs.

6.3.1 Structure using a single arc

The *adjusted Rand index* is presented in Table 6.1 for the case when the network consist of a single arc. The scores are compared against the best scores that were found in Section 5.3. Note that an added arc, if chosen correctly, appears to always produce better results than any previous approaches. This is not surprising, since the possibility to add an arc and collapse two variables should only benefit the model. In some datasets, e.g., *Lymphography*, the difference is quite large, whereas in others, e.g., *SPECT*, the difference is more subtle.

Table 6.1: *Adjusted Rand index* for best structure using one arc. It is compared to the best result found in Table 5.1. Abbreviations: Correlation clustering (CC), Spectral clustering (SC), Fisher kernel (F), Hamming kernel (H), scaled-Hamming kernel (SH).

	One arc	Best score previously
Breast Cancer	**0.173**	0.162 (SC-SH)
Hayes-Roth	**0.093**	0.082 (CC-F/SH)
House Votes-84	**0.585**	0.564 (SC-H/SH)
Lymphography	**0.316**	0.259 (SC-SH)
Primary Tumor	**0.164**	0.123 (CC-SH)
SPECT	**0.066**	0.057 (CC-SH)
Tic-Tac-Toe	**0.069**	0.020 (CC-SH)

The results in Table 6.1 raises the question of how the added arc improves the score. As suggested by the theoretical exploration, adding a single arc is simply a way of collapsing two variables into one. Thus, it could be the case that the combination of the two variables convey more information about the class label than

the individual variables. The correlation to the class label according to *Theil's U* is presented in Table 6.2. The direction of the metric presented is the one conditioned on the variables, thus, revealing the amount of information the variables contain about the class belonging. The columns v_1 and v_2 correspond to the individual variables and $v_1 \cap v_2$ to the collapsed variable. Apparent from the results is that the collapsed variable reveals more information about the class belonging than each of the individual variables.

Table 6.2: *Theil's U* for the class label conditioned on each variable individually, as well as the corresponding collapsed variable.

	v_1	v_2	$v_1 \cap v_2$
Breast Cancer	0,002	0,088	**0,116**
Hayes-Roth	0,181	0,178	**0,431**
House Votes-84	0,439	0,205	**0,450**
Lymphography	0,028	0,327	**0,406**
Primary Tumor	0,080	0,152	**0,209**
SPECT	0,039	0,033	**0,062**
Tic-Tac-Toe	0,015	0,015	**0,031**

In addition to *Theil's U*, correlations between the variables and the class label are compared using *Cramer's V*, which is presented in Table 6.3. The trend appears to be the same, i.e., that the correlation between the class label and the collapsed variable is higher than for any of the individual variables. Neither *Lymphography* nor *Primary Tumor* follows this trend though, which most likely is due to the fact that the metric is symmetrical. It appears that in these cases the class label reveals much information about the individual variables, which gives a high correlation. However, this is not a direction of any interest, since in practice the class label would not be present when the dataset is clustered.

Table 6.3: *Cramer's V* between the class label and each variable individually, as well as the corresponding collapsed variable.

	v_1	v_2	$v_1 \cap v_2$
Breast Cancer	0,000	0,323	**0,336**
Hayes-Roth	0,459	0,455	**0,621**
House Votes-84	0,710	0,509	0,704
Lymphography	0,166	0,490	**0,589**
Primary Tumor	0,686	0,449	0,396
SPECT	0,177	0,153	**0,220**
Tic-Tac-Toe	0,130	0,130	**0,177**

6.3.2 Structure using two arcs

When yet another arc is added to the network structure, all possible varieties of structure segments are found. The best network structure for *Breast Cancer*, *SPECT* and *Tic-Tac-Toe* contain two separate single arcs. On the other hand, for *Hayes-Roth* and *House Votes-84*, the best networks contain A-structures (or Chain structures). Lastly, the best networks for *Lymphography* and *Primary Tumor* contain V-structures.

The results of clustering when using network structures with two arcs are presented in Table 6.4. It is apparent that adding the second arc helps in most of the cases, with the exceptions of *Hayes-Roth* and *Lymphography*. For *Hayes-Roth* the score actually decreases, which could indicate that the best structure for this rather small dataset (4 variables) consists of a single arc.

Table 6.4: *Adjusted Rand index* for the best structure using two arcs compared to that of one arc.

	Two arcs	One arc
Breast Cancer	**0.204**	0.173
Hayes-Roth	0.086	**0.093**
House Votes-84	**0.606**	0.585
Lymphography	**0.316**	**0.316**
Primary Tumor	**0.201**	0.164
SPECT	**0.071**	0.066
Tic-Tac-Toe	**0.084**	0.069

It is difficult to determine any practical differences between the structures using two arcs. It appears that even the non-intuitive V/A-structures can prove useful for the Fisher similarities in certain cases if the arcs are chosen correctly. Lastly, even though the score improved for almost all datasets, the number of arcs for the best structure could vary drastically between each dataset. As an example, *Hayes-Roth* appears to have reached that limit already and *Lymphography* is inclined to do the same in the next iteration. It should also be mentioned that when choosing the arcs greedily in this manner, it is likely that the optimal number of arcs differs from the true optimal structure.

7
Discussion and Conclusions

This chapter contains a general discussion of the results as well as conclusions. The research questions are addressed and possible future work is proposed.

7.1 Discussion

The first thing worth addressing is the narrow restrictions on the number of variables and entries of the datasets, which is a limitation to the study overall and closely related to the scope of the thesis. These restrictions are motivated by using exact inference methods as well as an approach that finds an optimal structure, as opposed to an adequate structure. While extended studies on larger datasets would be interesting, they would likely have to utilize structure learning algorithms more suitable for such conditions and perhaps even consider approximate inference methods.

Within the limits of the scope, however, we have shown that Bayesian networks is a solid choice for imputation in categorical datasets, which is in line with previous research. It clearly outperforms the naive approaches for each dataset and for the vast majority of settings of missing values. One could argue that the total range of missing values is too narrow and does not cover the full spectrum. Specifically, that it prevents any general conclusion to be drawn regarding the operational range for inference in Bayesian networks. However, the evaluated range of $1 - 25\%$ of missing values probably covers the majority of realistic scenarios. The choice to guarantee a fraction of complete entries to be used for structure learning might not be anchored in realistic scenarios in the same way. Although, as hypothesized, one could probably circumvent this problem to some degree by utilizing augmented models.

The main question regarding the imputation results in general is whether they are applicable in the clustering case or not. Later parts of the thesis suggests that a change of structure is required in order to enhance the performance of the Fisher kernel. The performance of such a structure, on the task of imputation, is undocumented and would be a natural topic for future work.

In contrast to the imputation task, the Bayesian network structure that optimizes the qNML-score seems to be a poor choice for the clustering task. Specifically, the Fisher kernel did not yield competitive results when based on this structure. It became apparent that a structure suitable for imputation, which best describes dependencies in the dataset, not necessarily coincides with a good structure for the purpose of clustering. Other scoring functions, such as Bayesian Dirichlet scores, were not investigated in this thesis, but are expected to produce similar results since they also guide the structure to best describe the data. Similarly, only supervised

metrics were considered, although it is again unlikely that unsupervised metrics will be able to show drastically different results. Surprisingly, the weighing of each individual variable proved to be useful, as the scaled-Hamming kernel was able to consistently compete with the best performing model.

Out of the two clustering algorithms that were used, only Correlation clustering appeared to produce any meaningful results in combination with the Fisher kernel. Nevertheless, these results were not competitive in the broader picture. Spectral clustering combined with the Fisher kernel consistently produced poor clusters, which were always evaluated to be close to random assignments. This is most likely a result of Spectral clustering only considering positive similarities, whereas the Fisher similarities can be both positive and negative. Hence, the Fisher similarities need to be preprocessed in this case. It is possible that a more sophisticated approach than simply shifting the similarity values could produce better results.

The fact that the scaled-Hamming kernel appeared to be reliable initiated the process of considering sparse network structures. The resulting investigation of Bayesian network structure segments displayed a number of interesting properties of the Fisher kernel. It was discovered that the addition of arcs can improve the clustering performance of the empty Bayesian network, given that the correct arcs are chosen. Notably, the Fisher kernel using such structures outperformed every other kernel in the study. However, no method of finding these arcs in an unsupervised setting was discovered. This could perhaps be done by considering a semi-supervised setting, where the labels of a subset of entries are used to guide the structural choices. Another approach could be to investigate correlation metrics between variables, which possibly can be used to guide the choice of arcs. Due to the ambiguity as to what a true clustering is, pursuing this in an entirely unsupervised setting is likely to be difficult, although this needs to be further investigated.

Another interesting property that was discovered is that both structural edge cases, i.e., an empty and a fully connected network structure, behaved in a predictable and intuitive way. The empty network treats all variables independently, whereas the fully connected structure can be treated as a single collapsed variable. This suggests that a structure in between these edge cases corresponds to a combination of collapsed, or partially collapsed, subsets of variables, even though this has not been proven. In the case of a single arc, the results indicated that additional information regarding the label could be provided by collapsing the variables. A closer inspection of other small structure segments can possibly provide similar insights, which can be used to understand when certain structures are useful. A better understanding of the properties of smaller structure segments would be helpful in developing a method for finding an optimal structure. This might allow a method to learn the structure by adding smaller structure segments, rather than single arcs, which could make the method more efficient.

7.2 Conclusions

The main purpose of this thesis was to investigate if, or how, the Fisher kernel can be used to produce competitive results in tasks related to categorical variables. Even though clustering using the Fisher kernel initially proved unsuccessful, it was later

demonstrated that it can be competitive under correct circumstances. The most significant discovery is that there exist network structures suitable for the purpose of clustering, although, they do not necessarily coincide with conventionally learnt structures. A correct choice of network structure, however, enables the Fisher kernel to improve the clustering results in comparison to other naive kernels. This was demonstrated by investigating, both theoretically and practically, the impact that various small network segments has on the Fisher kernel.

Additionally, it was discovered that the Fisher kernel of an empty network structure performs both reliably and consistently for clustering. This kernel is referred to as the scaled-Hamming kernel and can be used without the need of a Bayesian network, while still showing promising results. Thus, in contrast to the general Fisher kernel for Bayesian networks, the scaled-Hamming kernel is readily available for clustering without further modifications.

Another aspect of the thesis was related to the suspected advantages of using a Bayesian network for imputation. In line with existing research, inference in Bayesian networks proved to be a suitable method for imputation, given that the structure was learnt using conventional methods. This was the best performing method overall, surpassing all other naive methods of imputation. However, since this network does not necessarily coincide with the optimal network for clustering, the practical advantages could be limited.

7.2.1 Future work

The possible directions for future work are many. However, only the most prominent ones will be presented here. In the discussion section a few additional topics of future work have been introduced.

In order to make the Fisher kernel accessible, one could pursue the topic of developing a method of finding suitable arcs for the Bayesian network. This would preferably be pursued in either a semi-supervised or an unsupervised setting. If successfully done, it opens up the opportunity to use the Fisher kernel in practical scenarios, where the true labels do not exist. In addition, it is not clear how such a Bayesian network would perform when imputing missing values. This aspect also needs to be evaluated, in order to establish if the Fisher kernel truly can benefit from imputation using its underlying Bayesian network.

Another interesting topic to pursue further is the relation between arcs and collapsed variables with regards to the Fisher similarities. In this thesis, both edge cases have been presented, but it is still not clear how the similarities behave for arbitrary networks. It might be the case that arbitrary networks corresponds to partially collapsed variables, which if successfully shown can be used to provide additional insights regarding the Fisher kernels behavior for different structures.

Bibliography

[1] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493, 1999.

[2] Judea Pearl. Probabilistic reasoning in intelligent systems: Networks of plausible reasoning, 1988.

[3] Marco Di Zio, Mauro Scanu, Lucia Coppola, Orietta Luzi, and Alessandra Ponti. Bayesian networks for imputation. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 167(2):309–322, 2004.

[4] Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405, 1990.

[5] Richard E Neapolitan et al. *Learning bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[6] Tomi Silander, Teemu Roos, and Petri Myllymäki. Locally minimax optimal predictive modeling with bayesian networks. In *Artificial Intelligence and Statistics*, pages 504–511, 2009.

[7] Tomi Silander, Teemu Roos, Petri Kontkanen, and Petri Myllymäki. Factorized normalized maximum likelihood criterion for learning bayesian network structures. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models*, pages 257–264, 2008.

[8] Tomi Silander, Janne Leppä-aho, Elias Jääsaari, and Teemu Roos. Quotient normalized maximum likelihood criterion for learning bayesian network structures. In *International Conference on Artificial Intelligence and Statistics*, pages 948–957, 2018.

[9] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

[10] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[11] Richard W Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950.

[12] Tomi Silander and Petri Myllymaki. A simple approach for finding the globally optimal bayesian network structure. *arXiv preprint arXiv:1206.6875*, 2012.

[13] Tomi Silander. An exact bayesian network structure learning software based on dynamic programming. `https://github.com/tomisilander/bene`, 2018.

[14] Tomi Silander. Bayesian network learning package. `https://github.com/tomisilander/bn`, 2019.

[15] Dheeru Dua and Casey Graff. UCI machine learning repository, 2019.

[16] Thomas M Cover, Peter E Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[17] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):325–327, 1976.

[18] Songbo Tan. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667–671, 2005.

[19] Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

[20] Morteza Haghir Chehreghani. Clustering by shift. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 793–798. IEEE, 2017.

[21] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[24] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

[25] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[26] Henri Theil. On the estimation of relationships involving qualitative variables. *American Journal of Sociology*, 76(1):103–154, 1970.

[27] Harald Cramér. *Mathematical methods of statistics (PMS-9)*, volume 9. Princeton university press, 2016.

[28] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

[29] Wray Buntine. Theory refinement on bayesian networks. In *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann Publishers Inc., 1991.

[30] Tomi Silander, Petri Kontkanen, and Petri Myllymaki. On sensitivity of the map bayesian network structure to the equivalent sample size parameter. *arXiv preprint arXiv:1206.5293*, 2012.

[31] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[32] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pages 199–213. Springer, 1998.

[33] Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is np-hard. *Journal of the ACM (JACM)*, 55(2):11, 2008.

[34] Bill Rosgen and Lorna Stewart. Complexity results on graphs with few cliques. *Discrete Mathematics and Theoretical Computer Science*, 9(1):127–135, 2007.

[35] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

A
Bayesian Networks

This chapter provides an in depth description of learning a Bayesian network from data, including parameter- and structure learning. It also contains a description of inference in Bayesian networks using the junction tree algorithm.

A.1 Parameter learning

Learning the parameters is usually done in one of two ways. Either by computing the expected posterior (EP) estimate or by finding the maximum likelihood (ML) estimate.

Given a structure, G, and a dataset, D, the probability of a set of parameters θ can be expressed as

$$P(\theta \mid D, G) \propto P(D \mid G, \theta) P(\theta \mid G), \tag{A.1}$$

where the proportionality follows from Bayes' formula. The EP estimate is given by finding the θ that corresponds to the expected value of this quantity. This can be expressed formally as

$$\mathbf{E}_\theta \left[P(D \mid G, \theta) P(\theta \mid G) \right], \tag{A.2}$$

where $P(\theta \mid G)$ is a prior distribution over the parameters.

Assuming that $P(D \mid G, \theta)$ is multinomially distributed and that the prior $P(\theta \mid G)$ is Dirichlet distributed, then $P(\theta \mid D, G)$ is also Dirichlet distributed. This is the case because the Dirichlet distribution is the conjugate prior to the multinomial distribution. Under these assumptions, one can specify the parameter values for the EP estimate exactly, however, additional notation regarding the dataset is required:

- N_{ij} is the count of occurrences of X_i with parent configuration $\pi_i(X) = j$.
- N_{ijk} is the count of occurrences where $X_i = k$ with parent configuration $\pi_i(X) = j$.
- α_{ij} and α_{ijk} are the pseudo-count hyperparameters of the prior Dirichlet distribution.

Using this notation, the parameters can be specified exactly for the EP estimate:

$$\theta_{ijk}^{\texttt{EP}} := \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}. \tag{A.3}$$

Conversely, in the case where no information of the prior is given, one usually ignores it, i.e., set it to a uniform distribution. Solving the simplified maximization

problem,

$$\sup_{\theta} P(D \mid G, \theta), \tag{A.4}$$

yields the ML estimate of the parameters. Assuming that $P(D \mid G, \theta)$ is multinomially distributed, the ML estimate is expressed as

$$\theta_{ijk}^{\texttt{ML}} := \frac{N_{ijk}}{N_{ij}}. \tag{A.5}$$

Whenever $N_{ij} = 0$, the ML estimate can not be uniquely determined since the corresponding parameters have no impact on the maximization in Eq. A.4. However, in this situation, it is common to use a uniform distribution over these parameters, i.e., assuming that all categories are equally probable.

A.2 Structure learning

Using the score-based approach requires a sophisticated way of defining the score of a structure. In general, one can describe a scoring function using two terms. The first term considers the fitness of a given structure for the dataset. The second term is a penalization term, which is used to avoid overfitting the structure to the data. Many scoring functions omit the penalization term and only use a measure of fitness. The first term can be described in several different ways, however, a common starting point is to consider either the likelihood or the marginalized likelihood of a dataset. Although, in general, one wants the fitness measure to correspond to a measure of how well the data can be described using a given structure. In addition, it is customary to define a scoring function in terms of the logarithm of this fitness measure.

Let $s(G, D)$ represent an arbitrary scoring function. In the following sections, a few scoring functions relying on different assumptions are introduced. However, they can all be expressed uniformly in terms of a fitness measure and a penalty term. If the penalty term is included, it is denoted as $\Delta(G, D)$ and is subtracted from the fitness of the structure.

A.2.1 Bayesian Dirichlet scores

A score that is often encountered in practice is the Bayesian Dirichlet equivalent uniform (BDeu) score, which is a special case of the Bayesian Dirichlet (BD) score. As the name of the score suggests, it employs a Bayesian approach and considers the marginal likelihood, where the parameters have been marginalized out:

$$P(D \mid G) = \int P(D \mid G, \theta) P(\theta \mid G) \, \mathrm{d}\theta. \tag{A.6}$$

Assume that the data is multinomially distributed and that θ is Dirichlet distributed. Under the additional assumptions of *parameter-* and *independence modularity* it has been shown that [28]:

$$P_{\texttt{BD}}(D \mid G) := \prod_{i=1}^{n} \prod_{j=1}^{q_i} \left(\frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \times \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})} \right), \tag{A.7}$$

where $\Gamma(\cdot)$ is the gamma function. The Bayesian Dirichlet (BD) score does not use a penalization term and is defined as:

$$s_{\texttt{BD}}(G, D) := \log P_{\texttt{BD}}(D \mid G). \tag{A.8}$$

Adding the two additional assumptions of *likelihood equivalence* and *structure possibility*, the hyperparameters in Eq. A.7 can be specified as

$$\alpha_{ijk} = \alpha \times P(X_i = k, \pi_i(X) = j \mid G), \tag{A.9}$$

which is the only difference when defining the BDe score [28]. Finally, it was proposed to let

$$P(X_i = k, \pi_i(X) = j \mid G) = \frac{1}{r_i q_i}, \tag{A.10}$$

in order to give the same score to equivalent structures [29]. This score is referred to as the BDeu score and it only depends on a single hyperparameter, α, commonly referred to as the *equivalent sample size*.

Although BDeu is a commonly used score in practice, the corresponding optimal structure is sensitive to the choice of its single hyperparameter. By letting α range from small to large values, it has been shown that the optimal network can contain any number of arcs [30].

A.2.2 Information theoretic scores

Two well known scores, belonging to the information theoretic scores, are the Bayesian Information Criterion (BIC) [31] and the Akaike Information Criterion (AIC) [32]. In order to describe the fitness of a structure, both scores use the ML estimate of the parameters in Eq. A.5 to compute the likelihood of the data given a network structure,

$$P_{\texttt{ML}}(D \mid G) := P(D \mid G, \theta^{\texttt{ML}}). \tag{A.11}$$

The difference between the two scores is the penalization terms:

$$\Delta_{\texttt{AIC}}(G, D) := \sum_{i=1}^{n}(r_i - 1)q_i, \quad \Delta_{\texttt{BIC}}(G, D) := \frac{1}{2}\log(N)\sum_{i=1}^{n}(r_i - 1)q_i. \tag{A.12}$$

The AIC penalization term is simply the number of parameters in θ for the network, while the BIC term includes the fact that each parameter uses $\log(N)/2$ bits of memory in order to be stored. The scores are expressed in their entirety as:

$$s_{\texttt{AIC}}(G, D) := \log P_{\texttt{ML}}(D \mid G) - \Delta_{\texttt{AIC}}(G, D), \tag{A.13}$$

$$s_{\texttt{BIC}}(G, D) := \log P_{\texttt{ML}}(D \mid G) - \Delta_{\texttt{BIC}}(G, D). \tag{A.14}$$

A.3 Inference

An inference approach, which is often efficient in practice, is based on the junction tree algorithm [9]. The core idea of the junction tree algorithm is to compute joint probabilities from a representation based on clique marginals, rather than using the conditional probabilities of the Bayesian network directly. A clique is defined in undirected graphs as a subset of nodes, such that all of the nodes are neighbors to each other.

The first step of the algorithm is to moralize the graph, which is specific for Bayesian networks since they utilize directed graphs to describe the structure. Moralization is done by adding edges between nodes with children in common, as well as forgetting the direction of all edges.

The next step is to triangulate the graph such that every cycle of length 4 or longer has a short-cut, or a chord. Several solutions exist to this problem. However, it would be preferable to find the triangulation that minimizes the maximal clique state size, since this is crucial for the time complexity of the later parts of the algorithm [9]. To find such a solution is an instance of the minimum-weight triangulation problem, which is known to be NP-hard [33].

As an example, by conducting these two steps on the graph presented in Figure 2.1, one obtains the undirected graph in Figure A.1. The dashed blue lines correspond to the added edges from moralization and the dotted red line corresponds to one solution of the resulting triangulation problem.

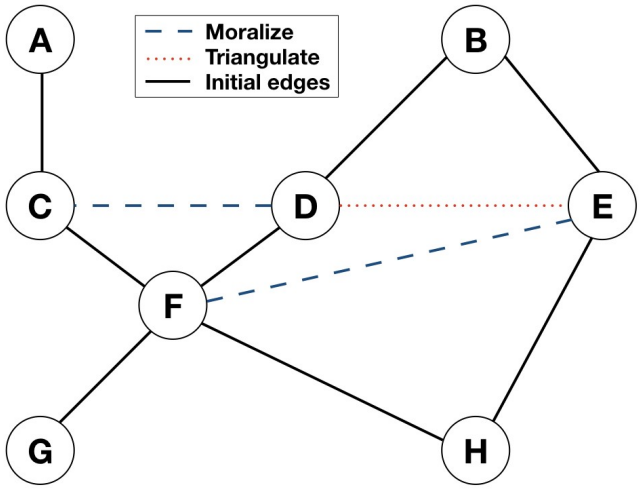

Figure A.1: The example graph after moralization (dashed blue lines) and loss of directionality, followed by triangulation (dotted red line).

In order to construct a junction tree from the triangulated graph, G, its corresponding clique graph must first be formed. The clique graph, $C(G)$, is formed by grouping together nodes of G that form maximal cliques. Each maximal clique in G corresponds to a node in $C(G)$ and an edge in the clique graph $C(G)$ corresponds to two intersecting maximal cliques of G. In other words, the clique graph $C(G)$ is

the intersection graph of the maximal cliques of G. The clique graph of the example is illustrated by the graph in Figure A.2. In the general case, it is NP-complete to find all the maximal cliques in a graph. However, at this stage of the algorithm the graph is triangulated, which guarantees that it can be done in polynomial time [34].

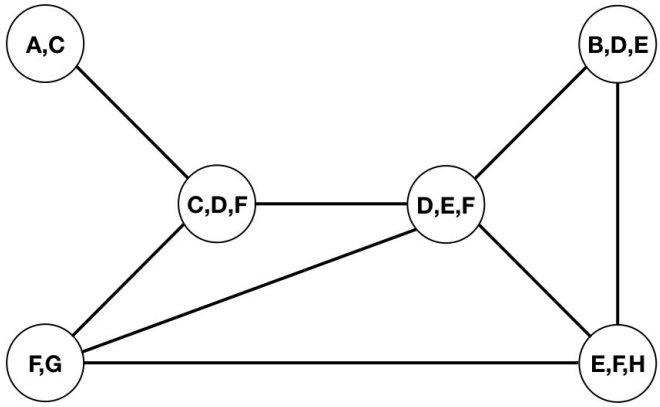

Figure A.2: Continued example showing the clique graph of the triangulated moral graph.

A junction tree is a tree of the clique graph with the *running intersection property*, which demands that the intersection of any two cliques is contained in every clique on the path connecting the two cliques. Triangulated graphs are guaranteed to have a junction tree that corresponds to the maximum spanning tree of the clique graph, assuming that the edges are weighted by the size of the clique intersections. The junction tree of the example is illustrated in Figure A.3, where the intersection of neighboring cliques are presented on the edges of the junction tree. As an example, the intersection $\{F, G\} \cap \{E, F, H\} = \{F\}$ appears at every edge on the path between $\{F, G\}$ and $\{E, F, H\}$, which is in accordance with the *running intersection property*.

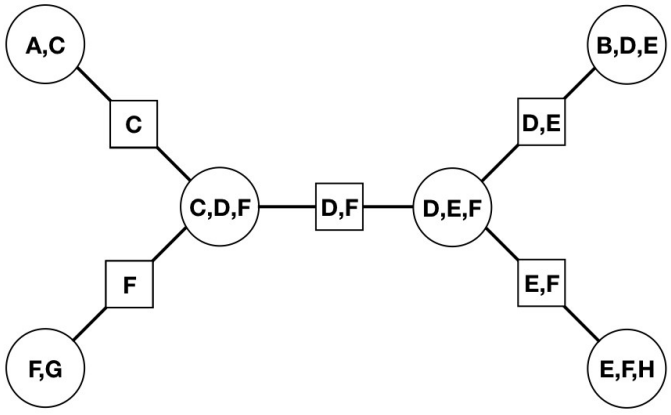

Figure A.3: Example of a junction tree with clique intersections, or separator sets, presented on the edges.

With the junction tree in place, the joint probability distribution can be expressed in terms of the clique marginals. Let $C_1, \ldots, C_k$ be the k nodes in the junction tree ordered in the Breadth First Search order, starting from an arbitrary root node C_1. Additionally, let the separator sets $S_i = C_i \cap C_p$ be the intersection of clique C_i and its parent C_p, and let the residual sets be $R_i = C_i \setminus S_i$. Due to the running intersection property, the joint probability distribution can be expressed as

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{k} P(R_i \mid S_i). \tag{A.15}$$

Furthermore, the running intersection property ensures that the clique marginals can be computed efficiently using the conditional probabilities [9]. With the junction tree in place and the clique marginals initiated, the effects of any evidence can be propagated with message passing, using the standard belief propagation algorithms on trees [2].

B

Imputation

This chapter contains the standard deviations of the results from Section 4.

B.1 Standard deviations

Table B.1: Standard deviation of the accuracy for all investigated datasets and methods. It is computed over 250 runs consisting of 25 settings: Test ratio 0.1-0.5 and ratio of missingness 0.1-0.5.

	BN	BN-Mode	BN-wKNN	Mode	wKNN
Breast Cancer	0.045	0.046	0.045	0.039	0.046
Hayes-Roth	0.086	0.076	0.084	0.074	0.078
House Votes-84	0.025	0.025	0.023	0.026	0.032
Lymphography	0.041	0.039	0.039	0.036	0.046
Primary Tumor	0.023	0.024	0.025	0.025	0.039
SPECT	0.027	0.029	0.029	0.036	0.038
Tic-Tac-Toe	0.029	0.035	0.039	0.019	0.063

Table B.2: Standard deviation of the cross-entropy loss for all investigated datasets and methods. It is computed over 250 runs consisting of 25 settings: Test ratio 0.1-0.5 and ratio of missingness 0.1-0.5.

	BN	BN-Mode	BN-wKNN	Mode	wKNN
Breast Cancer	0.077	0.079	0.079	0.067	0.413
Hayes-Roth	0.096	0.093	0.122	0.077	0.930
House Votes-84	0.050	0.050	0.051	0.032	0.237
Lymphography	0.085	0.079	0.086	0.060	0.360
Primary Tumor	0.041	0.043	0.044	0.038	0.241
SPECT	0.054	0.060	0.063	0.030	0.219
Tic-Tac-Toe	0.036	0.037	0.044	0.010	0.967

C
Clustering

This chapter contains an in depth description of the clustering metrics and additional experimental results from Section 5.

C.1 Clustering metrics

Before presenting three different clustering metrics, some notation has to be introduced. Let $U := \{U_1, \ldots, U_R\}$ and $V := \{V_1, \ldots, V_S\}$ be two different clusterings, or non-overlapping partitions, of a dataset with N entries.

Adjusted Rand index [23]: Consider an arbitrary pair of entries in a dataset, which are denoted as d_1 and d_2. There exists $\binom{N}{2}$ such pairs, which are used to compute the two following statistics:

- a is the number of pairs such that the two elements are in the same set in V and in the same set in U, i.e., $d_1, d_2 \in U_i$ and $d_1, d_2 \in V_j$ for some i, j.
- b is the number of pairs such that the two elements are in different sets in both V and U, i.e., $d_1 \in U_i, d_2 \notin U_i$ and $d_1 \in V_j, d_2 \notin V_j$ for some i, j.

The *Rand index* can now be expressed as

$$RI(U,V) := \frac{a+b}{\binom{N}{2}}, \tag{C.1}$$

and the *adjusted Rand index* is defined as

$$ARI(U,V) := \frac{RI(U,V) - \mathbf{E}[RI(U,V)]}{\max(RI(U,V)) - \mathbf{E}[RI(U,V)]}. \tag{C.2}$$

This corresponds to normalizing the *Rand index* and adjusting it for chance.

Adjusted mutual information [24]: The *mutual information* metric considers how the information of one of the clusterings influences the certainty of the other. It is defined as:

$$MI(U,V) := \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|} \tag{C.3}$$

Adjusting it for chance yields the *adjusted mutual information*:

$$AMI(U,V) := \frac{MI(U,V) - \mathbf{E}[MI(U,V)]}{\max\{H(U), H(V)\} - \mathbf{E}[MI(U,V)]}, \tag{C.4}$$

where $H(\cdot)$ is the entropy of a clustering. The entropy of an arbitrary clustering C is defined as

$$H(C) := -\sum_{i=1}^{|C|} \frac{|C_j|}{N} \log \frac{|C_j|}{N}. \tag{C.5}$$

V-measure [25]: The *V-measure* is a metric that returns the harmonic mean of the completeness and the homogeneity of a clustering solution. Usually it is defined between the true class labels and a clustering, but it can also be expressed using two arbitrary clusterings. The homogeneity and completeness are defined as the conditional entropies

$$h(U,V) := 1 - \frac{H(U \mid V)}{H(U)}, \tag{C.6}$$

$$c(U,V) := 1 - \frac{H(V \mid U)}{H(V)}. \tag{C.7}$$

The conditional entropy of U given V is defined as

$$H(U \mid V) := -\sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{|U_i \cap V_j|}{|V_j|}, \tag{C.8}$$

and $H(V \mid U)$ is expressed symmetrically. The entropies $H(U)$ and $H(V)$ are computed as specified in Eq. C.5. Finally, the *V-measure* is computed as

$$v(U,V) := \frac{2h(U,V)c(U,V)}{h(U,V) + c(U,V)}. \tag{C.9}$$

C.2 Additional results

Table C.1: *Adjusted mutual information* averaged over 10 runs for all algorithm-kernel combinations. Abbreviations: Correlation clustering (CC), Spectral clustering (SC), Fisher kernel (F), Hamming kernel (H), scaled-Hamming kernel (SH).

	CC-F	CC-H	CC-SH	SC-F	SC-H	SC-SH
Breast Cancer	0,069	-0,001	**0,079**	-0,003	0,000	0,078
Hayes-Roth	**0,066**	-0,008	**0,066**	-0,002	-0,011	-0,002
House Votes-84	0,055	**0,485**	0,475	-0,001	0,467	0,464
Lymphography	0,089	0,177	0,202	0,007	**0,218**	0,210
Primary Tumor	0,107	0,116	**0,187**	0,145	0,175	0,165
SPECT	0,018	0,103	0,125	0,002	0,108	**0,126**
Tic-Tac-Toe	0,004	**0,006**	**0,006**	0,000	-0,001	-0,001

Table C.2: *V-measure* averaged over 10 runs for all algorithm-kernel combinations. Abbreviations: Correlation clustering (CC), Spectral clustering (SC), Fisher kernel (F), Hamming kernel (H), scaled-Hamming kernel (SH).

	CC-F	CC-H	CC-SH	SC-F	SC-H	SC-SH
Breast Cancer	0,075	0,002	**0,084**	0,000	0,003	0,081
Hayes-Roth	**0,078**	0,004	**0,078**	0,010	0,000	0,010
House Votes-84	0,058	**0,495**	0,484	0,001	0,476	0,472
Lymphography	0,138	0,218	0,268	0,039	**0,288**	0,273
Primary Tumor	0,224	0,195	0,286	0,308	**0,336**	0,330
SPECT	0,024	0,122	**0,146**	0,008	0,121	0,144
Tic-Tac-Toe	0,005	**0,007**	**0,007**	0,001	0,000	0,000

D

Classification

This chapter provides a short evaluation of the Fisher kernel, similar to the one in Section 5, but in a supervised setting. The approach is described and the chapter ends with an evaluation of the experimental results.

D.1 Task description and approach

The task of classification differs from clustering in the sense that it is supervised. In classification, a set of objects is provided, as well as the correct label associated with each object. The task is therefore to find a general function, which after training is able to map a previously unseen object to the correct class label.

There exist a large variety of methods intended for classification. However, since the end-goal is to evaluate the Fisher kernel and not the classification method, the methods being considered are all kernelized, i.e., utilizing a similarity matrix. The two classification methods that are presented here are k-Nearest Neighbor (kNN) and Support Vector Machine (SVM).

D.1.1 k-Nearest Neighbors

The technique of k-Nearest Neighbors being used here is in general the same as the one presented in Section 4.1.2. However, for this application all neighbors are weighed uniformly.

D.1.2 Support Vector Machine

The Support Vector Machine (SVM) classifier learns to classify entries by dividing the feature space into subspaces associated with the different classes [35]. In the kernelized case, the entries are implicitly mapped into a high-dimensional feature space, where the classes hopefully are easier to separate. The problem is to find the optimal hyperplanes separating the different classes. This can be solved efficiently using the standard optimization method of Lagrange multipliers [35].

The optimal set of hyperplanes only depends on entries close to the boundaries. These entries, or vectors, support the boundaries, hence the name Support Vector Machine. New entries are classified by simply comparing them to the boundaries separating the different classes.

D.2 Experimental setup

Both SVM and kNN are used in order to evaluate the performance of the Fisher kernel on the task of classification. In addition, the classifiers are also evaluated together with the Hamming kernel and the scaled-Hamming kernel. The implementation used for SVM is from *Scikit-learn* [22] and it is used with default parameter values, including the penalty term $C = 1$. The kNN classifier is used with seven neighbors, i.e., $k = 7$, together with uniform weighing. The metrics used for evaluating the performance of all classifier-kernel combinations are accuracy and cross-entropy loss, which are presented in 4.2.3.

D.3 Results

The average accuracy for each classifier-kernel combination is presented in Table D.1. The results have been produced using 80% of the data as training data and 20% as test data. Both *Lymphography* and *Primary Tumor* contain rare classes (only 1-2 instances) and are therefore not evaluated, since it is unlikely for the test and training sets to be similarly distributed.

Table D.1: Average accuracy over 10 runs for all combinations of classifier and kernel. Abbreviations: Fisher kernel (F), Hamming kernel (H), scaled-Hamming kernel (SH).

	kNN-F	kNN-H	kNN-SH	SVM-F	SVM-H	SVM-SH
Breast Cancer	0.671	**0.736**	0.705	0.693	0.700	0.693
Hayes-Roth	0.647	0.641	0.634	0.791	**0.825**	0.803
House Votes-84	0.915	0.924	0.907	0.876	**0.952**	0.943
Lymphography	-	-	-	-	-	-
Primary Tumor	-	-	-	-	-	-
SPECT	**0.831**	0.785	0.811	0.794	0.813	0.817
Tic-Tac-Toe	0.755	0.947	0.911	0.848	**0.981**	**0.981**

The difference between using the Fisher kernel and the other kernels has shrunk compared to the clustering evaluation in Section 5.3. In a classification setting, it appears that the results are less influenced by the choice of kernel. This is not surprising, since the models now are able to use the label when interpreting the kernel similarities. Nonetheless, the Hamming kernel and the scaled-Hamming kernel still appears to be the best choices among the three kernels.

Another aspect is that the scaled-Hamming similarities appear to produce competitive results. Similar to the situation in clustering, it seems like adding arcs and dependencies between variables yields worse results.

Looking at the cross-entropy loss, presented in Table D.2, the results follow the same pattern. The Fisher kernel is the worst performing kernel, even though the gap to the other kernels has shrunk in comparison to the unsupervised setting. Furthermore, the Hamming kernel is the best performing kernel according to this metric, but scaled-Hamming still produces competitive results in all of the datasets.

Table D.2: Average cross-entropy loss over 10 runs for all combinations of classifier and kernel. Abbreviations: Fisher kernel (F), Hamming kernel (H), scaled-Hamming kernel (SH).

	kNN-F	kNN-H	kNN-SH	SVM-F	SVM-H	SVM-SH
Breast Cancer	0.897	1.013	0.951	0.611	**0.595**	0.616
Hayes-Roth	0.705	0.835	0.584	0.453	**0.337**	0.396
House Votes-84	0.264	0.329	0.366	0.355	**0.134**	0.164
Lymphography	-	-	-	-	-	-
Primary Tumor	-	-	-	-	-	-
SPECT	0.566	0.576	0.617	0.449	**0.381**	0.386
Tic-Tac-Toe	0.569	0.247	0.298	0.343	0.107	**0.089**

From these results, one can conclude that providing the label helps the Fisher kernel to close the performance gap, although, it still performs worse than the other kernels. It is apparent that the resulting network structure still impedes the performance of the Fisher kernel. This can be seen because the Fisher kernel with an empty network, i.e., the scaled-Hamming kernel, performs better than the Fisher kernel using the resulting structure from the data.

E

Proof

This chapter presents a full proof on the equivalency between the Fisher kernel of a fully connected Bayesian network and that of a single collapsed variable.

Claim: A Bayesian network structure that is a fully connected DAG yields a Fisher similarity equal to that of a single collapsed variable.

Proof: The claim can be formalized by assuming an arbitrary fully connected DAG of m variables as well as considering two arbitrary entries, u and v. The variables are denoted as X_i for $i \in \{1, \ldots, m\}$ and are indexed in their topological ordering, which is unique for a fully connected DAG. The total Fisher similarity between u and v can then be decomposed into

$$K(u, v) = \sum_{i=1}^{m} K_i(u, v).$$

Using the notation $\Pi(k) := P(X_1, \ldots, X_k)$ to represent the joint probability of the first k variables, the claim states that

$$\sum_{i=1}^{m} K_i(u, v) = \begin{cases} \frac{1}{\Pi(m)} - 1 & \text{if } u_i = v_i \ \forall i \in \{1, \ldots, m\} \\ -1 & \text{otherwise.} \end{cases}$$

Given the network of m variables, it can be reconstructed by adding the variables one at a time. This is done by adding the variables according to the topological order, with incoming edges from all of the previous variables. Thus, in order to prove the claim, it needs to hold whenever a new variable is added to the network. This will be showed using an induction proof.

In the first step of the induction, when considering only the first variable in the topological order, the claim is trivially true. The next step is to assume that the claim is true after k variables have been added in total, i.e,

$$\sum_{i=1}^{k} K_i(u, v) = \begin{cases} \frac{1}{\Pi(k)} - 1 & \text{if } u_i = v_i \ \forall i \in \{1, \ldots, k\} \\ -1 & \text{otherwise.} \end{cases}$$

In the case of $k + 1$ variables, the fully connected DAG has arcs from all of the original k variables to X_{k+1}. The contribution to the similarity score from all of the previous k variables remain the same, since the contribution of each variable only depends on its parents and not its children.

The contribution of the new variable is expressed using $\varphi := P(X_{k+1} \mid X_1, \ldots, X_k)$. Four cases have to be considered:

All $k+1$ variables match: $$\sum_{i=1}^{k+1} K_i(u,v) = \frac{1}{\Pi(k)\varphi} - 1$$

X_{k+1} match, but not all other variables: $$\sum_{i=1}^{k+1} K_i(u,v) = -1$$

X_{k+1} does not match, but all other variables do: $$\sum_{i=1}^{k+1} K_i(u,v) = -1$$

X_{k+1} does not match and neither do all other variables: $$\sum_{i=1}^{k+1} K_i(u,v) = -1$$

Note that $\Pi(k)\varphi = P(X_{k+1} \mid X_1, \ldots, X_k)P(X_1, \ldots, X_k) = P(X_1, \ldots, X_{k+1}) = \Pi(k+1)$. In addition, since the last three cases equal -1, the similarity can be written in the more compact form:

$$\sum_{i=1}^{k+1} K_i(u,v) = \begin{cases} \frac{1}{\Pi(k+1)} - 1 & \text{if } u_i = v_i \ \forall i \in \{1, \ldots, k+1\} \\ -1 & \text{otherwise,} \end{cases}$$

which corresponds to the claim remaining satisfied after adding one additional variable. In other words, the network of $k+1$ variables can be collapsed into a single variable, given that the network of k variables does the same.

Thus, by induction, variables can continuously be added until the network is fully reconstructed, while still satisfying the claim. Consequently, since the fully connected DAG of m variables is chosen arbitrarily, the proof is concluded. ■