Title
Robust Decision Making with the Same-Decision Probability

Author
Chen, Suming Jeremiah

Publication Date
2015

Peer reviewed|Thesis/dissertation

University of California
Los Angeles

Robust Decision Making with the Same-Decision Probability

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Suming Jeremiah Chen

2015

Abstract of the Dissertation

Robust Decision Making with the Same-Decision Probability

by

Suming Jeremiah Chen

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2015

Professor Adnan Darwiche, Chair

When making decisions under uncertainty, the optimal choices are often difficult to discern, especially if not enough information has been gathered. Two key questions in this regard relate to whether one should stop the information gathering process and commit to a decision (stopping criterion), and if not, what information to gather next (selection criterion). The proposed thesis is concerned with addressing this problem in light of a new advance, known as the *Same–Decision Probability (SDP)*, which is the probability that we would make the same decision had we known what we currently do not know. In this thesis, we show how the SDP can be used to be an effective stopping criterion, and compare it to traditional criteria to demonstrate how it provides a fresh perspective in decision making under uncertainty. Additionally, we develop the first exact algorithm to compute the SDP so that it may be used as a stopping criterion. We demonstrate the effectiveness of these algorithms on real and synthetic networks, and show that our proposed stopping criterion can lead to an early stopping of information gathering. Furthermore, we demonstrate that the SDP can be used as a selection criterion. In particular, since there are many criteria for measuring the *value of information (VOI)*, each based on optimizing different objectives, we propose a new SDP-

based criterion for measuring the value of information — this criterion values information that leads to robust decisions (i.e., ones that are unlikely to change due to new information). We develop the first algorithm to optimize the value of information, given the SDP as the reward criterion, and show empirical results that prove the utility of this novel criterion. We further answer several questions regarding the computational complexity of the SDP, which is known to be PP^{PP}-complete. Finally, we present results of applying the SDP as an information gathering criterion in practical problems including tutoring systems (do we need to ask more questions?) and machine learning (do we have enough data?).

The dissertation of Suming Jeremiah Chen is approved.

Wei Wang

William Kaiser

Richard Korf

Adnan Darwiche, Committee Chair

University of California, Los Angeles

2015

To my parents, who taught me the importance of faith and sacrifice.

Table of Contents

List of Figures

List of Tables

Acknowledgments

The PhD process has included countless moments that were challenging and gratifying. There were many nights where I couldn't sleep because I was so immersed in a problem, but in contrast, there were also quite a few times where I remember the pure joy of making a new discovery. I am very grateful for this opportunity and will remember this period very fondly. I believe that my personal growth the past few years has been influenced heavily by working with so many intelligent and amazing people, whom I would like to thank.

First and foremost, I would like to thank my advisor Adnan Darwiche. I am deeply indebted to Adnan, as he encouraged me to never settle and to continually maintain the highest level of excellence in my research. He provided me a fascinating research topic — I believe that with a different topic I may have taken a few extra years in order to graduate. Adnan provided me with an excellent balance of guidance and expertise that allowed me to slowly learn how to independently conduct research without ever going too off-track. Under his tutelage, I have learned many crucial skills, such as how to communicate more effectively and how to be meticulous in my work. These skills extend beyond artificial intelligence research and I believe they will prove useful to me throughout my life. I have been incredibly fortunate to have Adnan as an advisor.

Next, thank you to Arthur Choi, a research scientist in the Automated Reasoning group. Throughout my studies, I was lucky enough to share the same office as him and to thus soak in his expertise and comprehensive knowledge. Arthur has helped revise countless technical reports and papers, helped me develop my proofs and algorithms, and has given me valuable feedback every step along the way. Every paper I have written, I have gotten great feedback from both Adnan and Arthur. They have helped me elevate myself as a researcher. To use a separate analogy: if I were a professional basketball player, I would consider Adnan

to be my coach and Arthur to be my trainer. This entire thesis would not be possible without them.

Additionally, I would like to thank the members of my doctoral committee: Richard Korf, William Kaiser, and Wei Wang. I can't begin to imagine how busy most professors are so I am very grateful for the time they spent while serving on my committee. I would like to especially thank Rich, as he employed his widespread knowledge of search in order to give me valuable feedback on my own research. In addition, his comments and edits for both my prospectus and dissertation have greatly improved the readability. He is a very passionate about his work and his enthusiasm for solving difficult problems is inspiring. In addition, I learned a great deal about teaching from his teaching assistant training seminar. I would also like to thank Alan Yuille - who was one of the original members on my committee.

Subsequently, thank you to all other members of the Automated Reasoning group, whom I have been very privileged to learn from and interact with. Whether it was sharing offices or sharing ideas, going out to dinner or going to talks together, I would like to thank Eunice, Doga, Umut, Khaled, Guy, Alex, Ertan, Knot, and Tiansheng. I know there were probably many times where I was frustrated with my lack of progress and thus in a bad mood, so I would like to thank them for their ideas, patience, understanding, support, and helping to check for typos in my papers. I would also like to thank Keith for helping me get familiarized with our group's reasoning software. Thank you to some of the other graduate students at UCLA that I have had the opportunity to interact with on campus or at a conference: Ethan, Joseph, Elias, Karthika, and Bryant Chen. Thank you to Isaac, Li-Yang, Christine, and Jui-Ting for inviting me to Taiwanese Student Association events and putting up with my broken Mandarin and very broken Taiwanese.

Thank you to colleagues in industry who also helped in various forms. I'd like

to thank Saam for bringing me aboard Apple one summer to work on a research project — it was a valuable experience and I really enjoyed it. In addition, thank you to Mark Chavira for giving me feedback about my research and advice about next steps after graduation. Thank you to Markus Iseli for bringing me aboard as a researcher for the National Center for Research on Evaluation, Standards, & Student Testing. Thank you to Brian Milch for providing feedback for various research projects and for serving as a sponsor for a Google Research Award with which I was partially funded by.

As much as I would have liked to, I couldn't stay on campus all the time and often needed a change of scenery. I firmly believe that a good support system outside of graduate school is important to surviving graduate school. Through my studies, I have been extremely fortunate to have had many friends in Los Angeles who were there to support me in different ways.

I first would like to thank my brother Sukai — although he is my younger brother, he serves as a role model for me. His calm, collected, and compassionate nature has been a positive influence on me. Thank you to Felicia and Sara for hosting me and cooking for me so many times. To Woorae for hosting me so many times in Irvine and for showing me which Korean restaurants to avoid. To Alvin Chan and Megan for being running buddies. To Will, Julian, Jon(s), for being a supportive church small group (as well as honorary members Michelle and Rebecca). To Robert Suh and Andrew for teaching me how to dance and providing a place for me to park my car in Downtown LA. To my roommate Bez for putting up with my grumpiness during paper deadlines. To Greg and Robert Liu for showing me first hand that I will never be a professional gamer. To Angel and Anita for being good study buddies. To Christy for being the guitar player in our band. To Ronald, Alvin Chen, Anthony, Lilliana, and Jessica Chao for all the carpooling back to Northern California and all the good conversation. Special thanks goes to Sun, who was there during some especially down times to come to

UCLA every Friday to play basketball with me — additional thanks goes to him for passing me the ball and showing me how to be clutch. A very special thanks goes to Jack and Alison, for always being so welcoming, letting me nap on their couch, and providing me a place to safely and freely do laundry.

At this point, I would be remiss if I didn't thank my close friends even outside of Los Angeles. To my home church friends Kevin Yen, Tony, Esther, Trang, Kevin Lee, Jeffrey, Eric Lee, and Hank — I am happy to have grown up with you and to have stayed in touch. To my high school and college classmates Bryant Law, Kevin Liu, Ray, Jack, Eric, John Hwang, Randy, David Luoh, Stephen, Joe, Dan Chen, Sarina, and James Hsu — I don't think any of us would have ever thought that I would end up staying in school the longest but I am thankful that during this process, I've had so many of you to turn to for help when needed. Also, thanks to Jenn for being supportive and giving good advice.

Finally, I would also like to thank my parents Abraham and Susan, as well as my godparents Kirk and Shirley, for providing such a welcome environment for me every time I went home for the holidays. Additional thanks goes to my grandparents and aunt Shuhui for helping to raise me from when I was young. They gave me so much support, were always there to listen to my incessant complaining, and provided unconditional love and care.

Vita

2005-2009	B.S. (Electrical Engineering & Computer Sciences), University of California, Berkeley
2008-2009	Software Developer, Zynga
2009-2011	M.S. (Computer Science), University of California, Los Angeles
2011	Software Developer, FiveStars
2010-2014	Teaching Assistant, Department of Computer Science, University of California, Los Angeles
2014	Applied Machine Learning Intern, Apple

Publications

Suming Chen, Arthur Choi, and Adnan Darwiche. "The Same-Decision Probability: A new tool for decision making." In *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models*, pp. 51–58, 2012.

Suming Chen, Arthur Choi, and Adnan Darwiche. "An exact algorithm for computing the Same-Decision Probability." In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 2525–2531, 2013.

Suming Chen, Arthur Choi, and Adnan Darwiche. "Algorithms and Applications

for the Same-Decision Probability." *Journal of Artificial Intelligence Research*, pp. 601–633, 2014.

Suming Chen, Arthur Choi, and Adnan Darwiche. "Value of Information Based on Decision Robustness" In *Proceedings of the 29th Conference on Artificial Intelligence*, 2015.

Suming Chen, Arthur Choi, and Adnan Darwiche. "How Much Data Do We Need?" In preparation, 2015.

Suming Chen, Arthur Choi, and Adnan Darwiche. "Computer Adaptive Testing Using the Same-Decision Probability" In preparation, 2015.

CHAPTER 1

Introduction

In this chapter, we introduce the problem of robust decision making and then provide an outline of the remainder of the thesis.

1.1 Robust Decision Making

Probabilistic graphical models have often been used to model a variety of decision problems, e.g., in medical diagnosis [PK80, KRS97, GC99], fault diagnosis [LP06], classification [FGG97, RS01], troubleshooting [HBR95], knowledge assessment [MS08, MDC13] and in intrusion detection [KMR03, MBL08]. In these and similar applications, we often have to make decisions under uncertainty. As such, there are often unobserved variables, such as medical tests, that upon observation may change the current decision. Clearly, the presence of uncertainty may greatly hinder the decision maker's ability to reach the correct decision. For example, say a physician performs a few tests and then strongly believes the patient is suffering from substance-abuse induced depression. However, by not gathering more information in the form of further testing, the doctor could unwittingly be making a grave misdiagnosis. After all, 41% to 83% of female patients being treated for psychiatric disorders, including depression, have been misdiagnosed and have an unresolved physical ailment that ranges from hypothyroidism to cancer [KL97]. If the patient was actually suffering from hypothyroidism, this misdiagnosis could have been prevented quite easily if the doctor had either checked the patient's neck carefully or performed a blood test. This example can be seen as a warning

to not make decisions that could easily change given some new information (i.e., decisions that may not be robust).

The *Same-Decision Probability (SDP)* was recently proposed by [DC10], in order to help *quantify* the robustness of a decision, in the context of decision-making with Bayesian networks. In short, the SDP is the probability that we would make the same decision, if we were to perform further observations that have yet to be made. As such, the SDP can be treated as a measure of a decision's *robustness* with respect to some unknown variables, quantifying our confidence that we would make the same decision, even if we made some further observations.

In this thesis, we thoroughly discuss the application of SDP as an information gathering tool. In particular, the thesis discusses:

- How does the SDP compare to traditional approaches for decision making under uncertainty?
- How can it be used to develop corresponding stopping and selection criteria for information gathering?
- How do we compute the SDP efficiently?
- How do we select observations to maximize the expected SDP efficiently?
- What is the computational complexity of the SDP?
- How can we apply the SDP in real–world applications, such as tutoring systems and machine learning?

We next provide an overview of the chapters of this thesis.

1.2 Overview

Chapter 2: Background and Related Work

In this chapter, we introduce the notation that is used consistently throughout this thesis. Next, we review the previously introduced notion of the *Same-Decision Probability (SDP)*, which has been previously introduced as a tool for quantifying the robustness of decisions. Finally, we review the traditional stopping criteria that is used to determine when to stop information gathering in probabilistic graphical models. Subsequently, we also review the traditional selection criteria that is used for determining which pieces of information should be gathered next.

Chapter 3: Using the SDP as a Stopping Criterion

In this chapter, we introduce a variety of scenarios where the Same-Decision Probability can be used as a stopping criterion. We contrast the usage of the SDP with classical criteria. In particular, we show that the usage of the SDP can allow us to distinguish between robust and non-robust decisions in ways that are indistinguishable by classical criteria. Additionally, we show that there are scenarios where classical criteria may call for performing further observations, but where the SDP indicates that our decision is unlikely to change.

Chapter 4: Computing the SDP

In this chapter, we propose the first exact algorithm for computing the SDP. This algorithm is based on a novel combination of branch-and-bound search with classical inference techniques for graphical models. The SDP has been shown to be highly intractable, and its exact computation was previously limited to toy examples, with few variables, through brute-force enumeration. Our introduced algorithm can be applied to real-world networks that are out of the scope of both brute-force enumeration and previously proposed approximation algorithms, and can be further applied to synthetic networks with as many as 100 variables. We present our algorithm and then discuss empirical results.

Chapter 5: Using the SDP as a Selection Criterion

In this chapter, we turn our attention to the use of the SDP as a criterion

for determining which variables should be selected. Throughout the literature, there are many criteria for measuring the *value of information (VOI)* of some unobserved variables, each based on optimizing different objectives. We show how to use the SDP as a criterion for measuring the VOI — using the SDP as a measure for the VOI will lead to selecting features that, when observed, would lead to a decision that is maximally robust against further observations. We compare the usage of the SDP as a selection criterion against traditional selection criteria, and present illustrative examples that clearly show the utility of our novel criterion.

Chapter 6: Maximizing the Expected SDP

In this chapter, we propose an algorithm to perform optimal feature selection in order to maximize the VOI with the SDP as a reward function in Naive Bayes networks, thus allowing us to attain the maximal *expected* SDP after observing those features. This allows us to select features that will lead to robust decisions. The algorithm is based on a combination of techniques used to solve the knapsack problem as well as a branch-and-bound search algorithm. After presenting our algorithm, we discuss empirical results. More specifically, we show how this algorithm can be applied to a variety of real-world problems, including 1) classification problems, where the usage of our algorithm can reduce budget expenditure significantly while reducing the classification accuracy only slightly, and 2) making robust decisions that minimize the liability of unseen observations.

Chapter 7: The Complexity of Computing the SDP

In this chapter, we present some recent complexity results on the SDP. These complexity results highlight both its relative intractability (even in naive Bayes networks), but also its relationship to a broader class of expectation computation problems, emphasizing the broader importance of developing effective algorithms for the SDP and related problems.

Chapter 8: SDP in Computer Adaptive Testing

In this chapter, we show how the SDP can practically be used in the real-world application of computer adaptive testing as an information gathering tool. We show how exactly the SDP can be applied and that its usage results in a reduction of the number of questions that need to be asked to evaluate the competency of a student, while maintaining the same level of precision and accuracy.

Chapter 9: SDP in Machine Learning

In this chapter, we show how the SDP can be used for machine learning practitioners. While constructing models and estimating parameters, a frequently asked question is whether one has enough data — how much can our model change, given that we observe more data? We show how the SDP can be used to provide answers that pinpoint exactly how much additional data is necessary in various scenarios so that one may do robust learning.

CHAPTER 2

Background and Related Work

In this chapter, we first introduce the notation that will be used throughout the thesis. Next, we review the notion of the *Same-Decision Probability*, a recently introduced notion that helps quantify the robustness of decisions [DC10, CXD12]. Finally, we review some common stopping criteria and selection criteria that are used for information gathering under uncertainty.

2.1 Notation

Throughout this thesis, we use standard notation for variables and their instantiations, where variables are denoted by upper case letters X and their instantiations by lower case letters x. Additionally, sets of variables are denoted by bold upper case letters $\mathbf{X}$ and their instantiations by bold lower case letters $\mathbf{x}$. We assume that the state of the world is described over random variables $\mathbf{X}$, where the evidence $\mathbf{E} \subseteq \mathbf{X}$ includes **all** known variables, and where hidden variables $\mathbf{U} \subseteq \mathbf{X}$ include **all** unknown variables. By definition, $\mathbf{E} \cap \mathbf{U} = \emptyset$ and $\mathbf{E} \cup \mathbf{U} = \mathbf{X}$. We often discuss the ramifications of observing a subset of hidden variables $\mathbf{H} \subseteq \mathbf{U}$ on decision making. Furthermore, we use $D \in \mathbf{U}$ to denote the main hypothesis variable that forms the basis for a decision.[1]

[1]The work presented in this thesis can be extended to the case of multiple hypothesis variables, but we focus here on the case of one hypothesis variable for simplicity.

D	$Pr(D)$
$+$	0.7
$-$	0.3

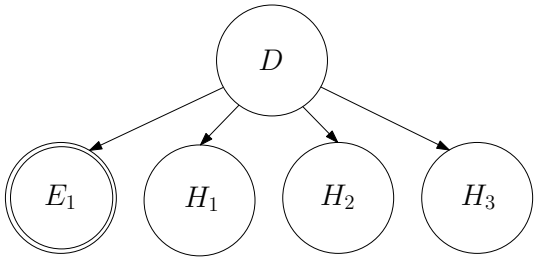


Figure 2.1: A Naive Bayes network (CPTs defined in Table 2.1).

2.2 Same-Decision Probability

The *Same-Decision Probability* (SDP) was initially introduced by [DC10] as a confidence measure for threshold-based decisions in Bayesian networks under noisy sensor readings. Prior to formally defining the SDP, we first show an example to provide intuition. Consider now the Naive Bayes network in Figure 2.1, where we have a class variable D that can either be $+$ or $-$, as well as feature variables E_1, H_1, H_2, and H_3. When the values of the feature variables are known, they act as *evidence* that can influence our belief in the hypothesis variable D. Networks such as this are typically used to compute the belief in the hypothesis given that we have observed some evidence, $Pr(d \mid \mathbf{e})$. The basis of whether or not to make a decision often depends on whether or not the posterior probability of that hypothesis d surpasses some threshold T [HCK92, HBR95, KMR03, LP06]. For example, in this case we may want to make a *positive* decision if $Pr(D = + \mid \mathbf{e}) \geq 0.80$.

Table 2.1: CPTs for the network in Figure 2.1. Note that the CPTs for $Pr(H_3 \mid D)$, $Pr(E_1 \mid D)$ and $Pr(H_2 \mid D)$ are identical.

D	H_1	$Pr(H_1 \mid D)$
$+$	$+$	0.80
$+$	$-$	0.20
$-$	$+$	0.10
$-$	$-$	0.90

D	H_2	$Pr(H_2 \mid D)$
$+$	$+$	0.70
$+$	$-$	0.30
$-$	$+$	0.30
$-$	$-$	0.70

In Figure 2.1, we have the case where a variable E_1 has been observed, a particular reading of two sensors and the resulting belief $Pr(D{=}+ \mid E_1{=}+) = 0.8448$. Suppose our threshold is $T = 0.8$, then as $Pr(d \mid \mathbf{e}) \geq T$, we would make a positive decision. Notice that there are still three unobserved feature variables H_1, H_2, and H_3. We want to make a more-informed decision based on the probability $Pr(d \mid \mathbf{h}, \mathbf{e})$ instead of making a decision based on just $Pr(d \mid \mathbf{e})$.

Consider Table 2.2, which enumerates all of the possible instantiations of observing those variables. In five of these cases the probability of the hypothesis passes the threshold (in bold), leading to the same decision. In the other three scenarios, a different decision would have been made. The SDP over variables H_1, H_2, and H_3 is thus the probability of the five scenarios in which the same decision would have been made. For this example, the SDP is:

$$0.332569 + 0.145189 + 0.145189 + 0.068430 + 0.095362 = 0.786739$$

indicating that even if we were to discover the values of the remaining three variables, there is still a 78.67% chance that we would make the same decision.

In [CXD12], the SDP is defined formally as:

Definition 1 (*Same-Decision Probability*)**.** *Let $\mathcal{N}$ be a Bayesian network that is conditioned on evidence $\mathbf{e}$, where we are further given a hypothesis d, a threshold T, and a set of unobserved variables $\mathbf{H}$. Suppose we are making a decision that is confirmed by the threshold $Pr(d \mid \mathbf{e}) \geq T$. The* ***Same-Decision Probability*** *in this scenario is*

$$SDP(d, \mathbf{H}, \mathbf{e}, T) = \sum_{\mathbf{h}} [Pr(d \mid \mathbf{e}, \mathbf{h}) \geq T] Pr(\mathbf{h} \mid \mathbf{e}), \tag{2.1}$$

where $[Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T]$ is an indicator function such that

$$[Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T] = \begin{cases} 1 & \text{if } Pr(d \mid \mathbf{e}, \mathbf{h}) \geq T \\ 0 & \text{otherwise.} \end{cases}$$

Table 2.2: Scenarios $\mathbf{h}$ for the network in Figure 2.1, where $\mathbf{H} = \{H_1, H_2, H_3\}$. Cases above the threshold $T = 0.8$ are in bold.

H_1	H_2	H_3	$Pr(\mathbf{h} \mid \mathbf{e})$	$Pr(d \mid \mathbf{h}, \mathbf{e})$
+	+	+	0.332569	**0.9958**
+	+	−	0.145189	**0.9976**
+	−	+	0.145189	**0.9976**
+	−	−	0.068430	**0.8889**
−	+	+	0.095362	**0.8682**
−	+	−	0.064810	0.5475
+	−	+	0.064810	0.5475
−	−	−	0.083638	0.1818

In other words, the Same-Decision Probability is the expectation that, even after observing some variables $\mathbf{H}$*, we would still make the same decision as if those variables had not been observed.*

The SDP is notably hard to compute — computing the SDP is proven to be in general PP^{PP}-complete [CXD12]. First, observe the following relationship between complexity classes:

$$\text{NP} \subseteq \text{PP} \subseteq \text{NP}^{\text{PP}} \subseteq \text{PP}^{\text{PP}}$$

Note that MPE, a query where we find the most probable instantiation of network variables, is the prototypical NP-complete problem for probabilistic inference [Shi94]. MAP [PD04], a query where we find the most probable instantiation of a *subset* of network variables, is the prototypical NP^{PP}-complete problem. SDP was shown to be PP^{PP}-complete, where PP^{PP} can be thought of as a counting variant of the class NP^{PP}, hence highly intractable.

From previous work on the SDP [DC10, CXD12], the two options for computing the SDP are

1. An *approximate* algorithm developed by [CXD12]. This algorithm uses an augmented variable elimination algorithm that produces a potentially weak bound based on the one-sided Chebyshev inequality.

2. A *naive* brute-force method that enumerates over all possible instantiations and sums up the probability of all the instantiations where the same decision would be made.

We will improve upon these methods — in Chapter 4 we present an exact algorithm for computing the SDP that is far faster than the naive brute-force method and has comparable running time to the approximate algorithm.

2.3 Stopping and Selection Criteria

When making decisions under uncertainty, it may be difficult to finalize a decision in the presence of unobserved variables. Given these unobserved variables, there are two fundamental questions. The first question is whether, given the current observations, the decision maker is ready to commit to a decision. We will refer to this as the *stopping criterion* for making a decision. Assuming the stopping criterion is not met, the second question is what additional observations should be made before the decision maker is ready to make a decision. This typically requires a *selection criterion* based on some measure for quantifying an observation's *value of information* (VOI). In this section, we first introduce some necessary notation, and then review some commonly used stopping and selection criteria.

2.3.1 Stopping Criterion

Given that there are hidden variables in our model and we have the choice of whether or not to observe some subset of them, a stopping criterion determines when we stop the process of information gathering and commit to a decision. Note that we are concerned with making a decision based on some hypothesis variable, such as the state of a patient's health. For a stopping criterion, the most basic approach used in a variety of domains is to commit to a decision once the belief about a certain event crosses some threshold, as is done by [PK80, KMR03, LP06]. However, this approach may not be robust, as further observations may cause the belief about the event to fall below the threshold. [GB11] note the possibility of this and pose the `STOP` problem, which asks whether or not the present evidence gathered is sufficient for diagnosis, or if there exists further relevant evidence that can and should be gathered.

Other approaches involve ensuring that the uncertainty surrounding the decision variable is sufficiently reduced. For instance, [GK11a] stop information gathering and commit to a decision when 1) the conditional entropy of the interest variable is reduced beyond some arbitrary threshold or 2) the margin between the first and second most likely states of the interest variable is above some threshold. In any case, it is clear that threshold-based stopping criteria are ubiquitous for decision making under uncertainty.

Alternatively, there are also several stopping criteria that involve the existence of a *budget*, which can be an abstract quantity to represent the available resources that can be used for information gathering. The budget may be representative of the number of observations that are allowed [MBL08, MS08, YKR09, CLT12], or in terms of a "monetary" amount that may be spent on observations of varying cost [GGR02, KG09, BG11]. In the context of a budget, the general stopping criterion is then to continue to make observations until the budget is completely

expended, as is done by [MBL08, MS08]. In [KG09, BG11], the budget is expended with the caveat that the *value of information* of an observation is at *least* the *cost* of the observation.

2.3.2 Selection Criterion: Value of Information

Should the stopping criterion determine that further observations are necessary, a *selection criterion* is then used to determine which variables should be selected for observation. Ideally, we want to observe all variables that will give us additional information with regards to our decision variable. However, due to resource constraints (such as a limited budget) this is often not possible. In this basic approach, a common selection criterion is to then select observations that will minimize the conditional entropy of the decision variable [Vom04, LP06, KG09, YKR09, ZJ10, GK11a, OD13, SS13]. The entropy of a variable X is defined as:

$$H(X) = -\sum_{x} Pr(x) \log Pr(x) \tag{2.2}$$

and is a measure of the uncertainty of the variable's state — if the entropy of a variable is high, that means that there is much uncertainty about what value that variable takes.[2] The uncertainty of the decision variable's true state makes it difficult to make a decision. Thus, a natural selection criterion is to observe variables to minimize the conditional entropy of the decision variable, where the conditional entropy of variable D *given* variable X is defined as:

$$H(D \mid X) = \sum_{x} H(D \mid x) Pr(x) \tag{2.3}$$

The conditional entropy is thus an *expectation* of what the entropy would be after observing X. A similar selection criterion is to observe variables that will

[2]In information theory, the logarithm is typically assumed to be base-2 [CT91], which we also assume throughout this thesis for convenience.

most greatly increase the margin between the posterior probabilities of the first and second most-likely states of the decision variable [KG09].

These selection criteria involve utilizing the notion of *value of information (VOI)* in order to quantify the value of various observations [Lin56, Str65, How66, Rai68]. The VOI of a set of variables can depend on various measures. In the two example selection criteria we have discussed, those measures would be entropy and margins of confidence. For instance, if observing a variable X would reduce the conditional entropy $H(D \mid X)$ more than observing variable X' would ($H(D \mid X) < H(D \mid X')$), then the value of observing X would be higher.

A general notion of VOI that is based on different reward functions is defined in [KG09]. In particular, given an arbitrary reward function R,[3] a hypothesis variable D, and evidence $\mathbf{e}$, the VOI of observing hidden variables $\mathbf{H}$ is:

$$\mathcal{V}(R, D, \mathbf{H}, \mathbf{e}) = \mathcal{E}R(R, D, \mathbf{H}, \mathbf{e}) - R(Pr(D \mid \mathbf{e})) \tag{2.4}$$

where

$$\mathcal{E}R(R, D, \mathbf{H}, \mathbf{e}) = \sum_{\mathbf{h}} R(Pr(D \mid \mathbf{h}, \mathbf{e}))Pr(\mathbf{h} \mid \mathbf{e}) \tag{2.5}$$

is the expected reward of observing variables $\mathbf{H}$ and $R(Pr(D \mid \mathbf{e}))$ is the reward had we not observed variables $\mathbf{H}$. By this definition, the reward function used by [LP06] and [KG09] to select variables in order to minimize the conditional entropy is then $R(Pr(D \mid \mathbf{e})) = -H(D \mid \mathbf{e})$, so maximizing the expected reward of observing variables $\mathbf{H}$ is then equivalent to minimizing the conditional entropy $H(D \mid \mathbf{H})$. Some other possible reward functions involve utility-based reward functions or threshold-based reward functions [MS08].[4]

[3] A reward function is assumed to take as input the probability distribution of the hypothesis variable, $Pr(D)$, and return some numeric value.

[4] For more on reward functions, see the list provided by [KG09].

Note that the vast majority of selection criteria use a myopic approach, in which out of all possible observations, just one observation is considered at a time, and the observation with the highest VOI is selected each time. This approach is greedy and short-sighted — the *optimal* VOI can only be computed by computing it non-myopically [BG11].

Myopic value of information is often used in many applications as it is easy to compute [DJ97, Vom04, GK11a]. However, the problem with myopic selection is that it is not optimal, as at times the "whole is greater than the sum of its parts", as each individual observation in a set $\mathbf{H}$ seemingly may not provide significant value, but the VOI of observing $\mathbf{H}$ can be very high. For instance, take the function $D = X_1 \oplus X_2$, where alone neither X_1 nor X_2 is useful, but together they are determinative of D [BG11]. Only by computing the non-myopic VOI can the the optimal VOI be obtained.

Due to the aforementioned problems with using myopic VOI, more recently, researchers have recently suggested using the non-myopic VOI instead of myopic VOI and have proposed various methods to compute the non-myopic VOI. [HHM93, LJ08, KG09, ZJ10, BG11]. Computing the non-myopic VOI of some hidden variables $\mathbf{H}$ is difficult as it involves computing an expectation over the possible values of $\mathbf{H}$, which quickly becomes intractable as $\mathbf{H}$ becomes larger.

Existing algorithms for computing the non-myopic VOI are approximate algorithms [HHM93, LJ08] or relatively limited algorithms that are restricted to tree networks with few leaf variables [KG09]. Additionally, the Value of Information Lattice (VOILA) has been introduced by [BG11]. VOILA is a framework in which **all subsets** of hidden variables $\mathbf{H}$ are examined, and the optimal subset of features can be found to increase classification accuracy while meeting some budget constraint.

For the remainder of the thesis, we show how the Same-Decision Probability can be used as both a stopping criterion and as a selection criterion.

CHAPTER 3

Using the SDP as a Stopping Criterion

By the definition of SDP, we can see that calculating a high SDP, in contrast to calculating a low SDP, would indicate a higher degree of readiness to make a decision, as the chances of the decision changing, given further evidence gathering, is lower. In this chapter we show that computing the SDP can provide additional insight and thus can distinguish scenarios that are otherwise indistinguishable based on standard stopping criteria. Note that the material in this chapter is based on the work published in [CCD12, CCD14].

3.1 Threshold-Based Decisions

The threshold-based decision is a classical notion in decision making under uncertainty, and it is commonly used as it requires no utilities to be elicited. Examples of threshold-based decisions are very prevalent in educational diagnosis [GCV98, CGV02, MP02, BHM04, Xen04, AW05, MS08], intrusion detection [KMR03, MBL08], classification [GK11a], fault diagnosis [HBR95, LP06], and medical diagnosis [PK80, KRS97, GC99].

Consider the sensor network in Figure 3.1, which may correspond to an intrusion detection application as discussed by [KMR03]. Here, the hypothesis variable is $D = \{+, -\}$ with $D = +$ implying an intrusion. Suppose we commit to a decision, and stop performing observations, when our belief in the event $D = +$ surpasses some threshold T, say $T = 0.55$. There are four sensors in this model,

D	$Pr(D)$
$+$	0.5
$-$	0.5

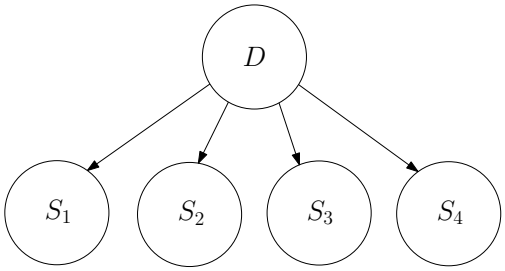


Figure 3.1: A Bayesian network for intrusion detection, with its CPTs given in Table 3.1

S_1, S_2, S_3 and S_4, whose readings may affect this decision.

Consider the two following scenarios:

1. $S_1 = +$ and $S_2 = +$.
2. $S_3 = +$ and $S_4 = +$.

As in the first scenario,

$$Pr(D = + \mid S_1 = +, S_2 = +) = 0.60 > 0.55,$$

and in the second scenario,

$$Pr(D = + \mid S_3 = +, S_4 = +) = 0.74 > 0.55,$$

it is clear that in both cases that the threshold has been crossed. We deem that no further observations are necessary based on our beliefs surpassing our threshold. Hence, when using thresholds as a stopping criterion (as is commonly done in [KMR03, LP06, GK11a]), the two scenarios are identical in that no more information is gathered and a decision is made.

From the viewpoint of SDP, however, these two scenarios are very different. In particular, the first scenario leads to an SDP of 52.97%. This means that there is a 47.03% chance that a different decision would be made if we were to further observe the two unobserved sensors S_3 and S_4. The second scenario, however,

Table 3.1: CPTs for the network in Figure 3.1. Parameterization 1.

D	S_1	$Pr(S_1 \mid D)$
$+$	$+$	0.55
$+$	$-$	0.45
$-$	$+$	0.45
$-$	$-$	0.55

D	S_3	$Pr(S_3 \mid D)$
$+$	$+$	0.60
$+$	$-$	0.40
$-$	$+$	0.40
$-$	$-$	0.60

D	S_2	$Pr(S_2 \mid D)$
$+$	$+$	0.55
$+$	$-$	0.45
$-$	$+$	0.45
$-$	$-$	0.55

D	S_4	$Pr(S_4 \mid D)$
$+$	$+$	0.65
$+$	$-$	0.35
$-$	$+$	0.35
$-$	$-$	0.65

leads to an SDP of 100%. That is, we would with certainty know that we would make the same decision if we were to also observe the two unobserved sensors S_1 and S_2: no matter what the readings of S_1 and S_2 could be, our beliefs in the event $D = +$ would always surpass our threshold 0.55. Indeed, as we can see in Table 3.1, the sensors S_1 and S_2 are not as strong as sensors S_3 and S_4, and in this example, they are not strong enough to reverse our decision.

This example provides a clear illustration of the utility of the SDP as a stopping criterion. However, some may argue that it is clear that in the second case, we should stop gathering information as $Pr(D = + \mid S_3 = +, S_4 = +) = 0.74$ has a larger margin from the threshold than $Pr(D = + \mid S_1 = +, S_2 = +) = 0.60$.[1] However, we show with the following example that deciding to stop based solely on the margin does not lead to robust decision making.

Consider once again the sensor network in Figure 3.1 and the parameterizations

[1]Thus using the aforementioned *margins of confidence* stopping criterion used by [GK11a].

of the sensor network shown in Table 3.2 and Table 3.3, which we respectively refer to as Case 1 and Case 2.[2] Note that in this example, we use a threshold of $T = 0.5$.

Table 3.2: CPTs for the network in Figure 3.1. Parameterization 2.

D	S_1	$Pr(S_1 \mid D)$
$+$	$+$	0.65
$+$	$-$	0.35
$-$	$+$	0.30
$-$	$-$	0.70

D	S_3	$Pr(S_3 \mid D)$
$+$	$+$	0.65
$+$	$-$	0.35
$-$	$+$	0.35
$-$	$-$	0.65

D	S_2	$Pr(S_2 \mid D)$
$+$	$+$	0.60
$+$	$-$	0.40
$-$	$+$	0.30
$-$	$-$	0.70

D	S_4	$Pr(S_4 \mid D)$
$+$	$+$	0.65
$+$	$-$	0.35
$-$	$+$	0.35
$-$	$-$	0.65

In both cases, when $S_3 = +$ and $S_4 = +$ are observed, $Pr(D = + \mid S_3 = +, S_4 = +) \geq T$. In particular,

1. Case 1: $Pr(D = + \mid S_3 = +, S_4 = +) = 0.775$.

2. Case 2: $Pr(D = + \mid S_3 = +, S_4 = +) = 0.599$.

By using the previously discussed margin stopping criterion, it would seem that in Case 1 we could stop information gathering, whereas for Case 2 more information gathering is necessary. However, we can compute the SDP for both cases for more insights on the nature of robustness in these settings. For Case

[2]Note that the exact numbers of the CPTs are not necessary to grasp these examples — CPTs are provided so that readers may reconstruct these networks.

Table 3.3: CPTs for the network in Figure 3.1. Parameterization 3.

D	S_1	$Pr(S_1 \mid D)$
+	+	0.50
+	−	0.50
−	+	0.40
−	−	0.60

D	S_3	$Pr(S_3 \mid D)$
+	+	0.55
+	−	0.45
−	+	0.45
−	−	0.55

D	S_2	$Pr(S_2 \mid D)$
+	+	0.50
+	−	0.50
−	+	0.40
−	−	0.60

D	S_4	$Pr(S_4 \mid D)$
+	+	0.55
+	−	0.45
−	+	0.45
−	−	0.55

1, we find that the SDP is 0.781, whereas for Case 2, we find that the SDP is 1.0 — even though in Case 1 the margin is higher, there is a greater chance that the decision would change given further information. This demonstrates that we cannot solely use the margin to determine whether or not to stop information gathering, and indicates the importance of having more powerful comprehensive stopping criteria.

It is clear from these examples that SDP is a useful stopping criterion for threshold-based decisions. First, the SDP can pinpoint situations where further observations are unnecessary as they would never reverse the decision under consideration. Second, the SDP can also identify situations where the decision to be made is not robust, and is likely to change upon making further observations.

3.2 Utility-based Decisions

In some cases the expected *utility* of different decisions, as well as the cost of reducing uncertainty (making observations), is quantified. This is common in the decision-theoretic setting [How66, HM84], where *influence diagrams* are commonly used. Influence diagrams can be seen as Bayesian networks that incorporate decision and utility nodes [HM84, Zha98, KM08]. The selection criterion for the decision-theoretic setting is clear: the observations that lead to the greatest increase of expected utility are selected. This approach is valued because it provides the expressability to model the unique desirability of different events. As such, the usage of utilities and observation costs is now prevalent; however, numerous researchers have noted the difficulty of coming up with the actual numerical quantities [GH89, LP06, BG11].

To show how the SDP can be used as a stopping criterion in the decision-theoretic context of expected-utility decisions and influence diagrams [HM84], we extend the definition of the SDP to a more general setting to allow for more applications. In particular, we assume that $\mathcal{F}$ is a polytime computable decision function that outputs a decision d based on the distribution $Pr(D \mid \mathbf{e})$. For instance, the decision function most commonly used in classification is to select the class with the highest posterior probability $\arg\max_d Pr(d \mid \mathbf{e})$ [FGG97], whereas for threshold-based decisions, the decision function would simply be to select a decision d if $Pr(D = d \mid \mathbf{e}) \geq T$.

The SDP is thus defined as the probability that the same decision would be made if the hidden states of variables $\mathbf{H}$ were known [CCD12].

Definition 2 (*Same-Decision Probability — Generalized*)**.** *Given a decision function $\mathcal{F}$, hypothesis variable D, unobserved variables $\mathbf{H}$, and evidence $\mathbf{e}$, the* ***Same-***

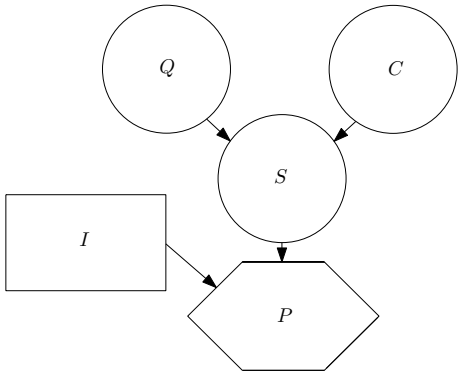


Figure 3.2: An influence diagram for an investment problem.

Decision Probability *(SDP) is defined as*

$$SDP(\mathcal{F}, D, \mathbf{H}, \mathbf{e}) = \sum_{\mathbf{h}} [\mathcal{F}(Pr(D \mid \mathbf{h}, \mathbf{e}))]_{\mathbf{h}} Pr(\mathbf{h} \mid \mathbf{e}) \tag{3.1}$$

where $[\mathcal{F}(Pr(D \mid \mathbf{h}, \mathbf{e}))]_{\mathbf{h}}$ *is an indicator function that*

$$= \begin{cases} 1 & \text{if } \mathcal{F}(Pr(D \mid \mathbf{h}, \mathbf{e})) = \mathcal{F}(Pr(D \mid \mathbf{e})) \\ 0 & \text{otherwise.} \end{cases}$$

The original SDP definition, however, assumed that D is a binary variable, where $\mathcal{F}(Pr(D \mid \mathbf{e})) = d$ when $Pr(d \mid \mathbf{e}) \geq T$ for some threshold T [DC10]. This more general definition captures the original definition of the SDP yet allows us to analyze decision-making scenarios outside threshold-based decision making.

We now consider the use of SDP as a stopping criterion in the context of expected-utility decisions and influence diagrams [HM84]. In particular, we show that by using the SDP, we can distinguish high-risk, high-reward scenarios from low-risk, low-reward scenarios that are otherwise indistinguishable when we consider the usage of VOI/utilities alone.

Consider the influence diagram in Figure 3.2, which consists of a Bayesian network with three variables (C, Q and S), a decision node I, and a utility node P that is a direct function of the utility function u. This influence diagram models an investment problem in which a venture capital firm is deciding whether

to invest an amount of \$5 million in a tech startup ($I = T$) or allowing the money to collect interest in the bank ($I = F$). In this example, the profit of the investment (P) depends on the decision (I) and the success of the company (S), which in turn depends on two factors: (1) whether the existing competitor companies are successful (C) and (2) whether the the co-founders of the startup have a high quality, original idea (Q). Both C and Q are unobserved initially and independent of each other. Variable S is the latent hypothesis variable in this case and thus cannot be observed. Variables C and Q, however, can be observed for a price.

The goal here is to choose the decision $I = i$ with the maximum expected utility:

$$\mathcal{E}U(i \mid \mathbf{e}) = \sum_{s} Pr(s \mid \mathbf{e})u(i, s),$$

where $u(i, s)$ is the utility of decision $I = i$ given evidence $\mathbf{e}$ on variables C and Q.

Figures 3.3 and 3.4 contain two different parameterizations of the influence diagram in Figure 3.2. We will refer to these as different scenarios of the investment problem.

In both scenarios, given no evidence on variables C and Q, the best decision is $I = F$, with an expected utility of $\$500K$. A decision maker may commit to this decision or decide to observe variables C and Q, with the hope of finding a better decision in light of the additional information. The classical stopping criterion here is to compute the *maximum* expected utility given that we observe variables C and Q [HHM93, DJ97]:

$$\max_{i} \sum_{c,q} \mathcal{E}U(i \mid c, q)Pr(c, q).$$

In both scenarios, the maximum expected utility comes out to $\$1,180K$, showing that further observations may lead to a better decision.[3]

[3]According to the formulation of [KG09], we have computed the VOI for variables C and Q

Q	$Pr(Q)$
T	0.4
F	0.6

Q	C	$Pr(S = T \mid .)$
T	T	0.60
T	F	0.90
F	T	0.20
F	F	0.30

C	$Pr(C)$
T	0.6
F	0.4

I	S	$u(I, S)$
T	T	$\$5 \times 10^6$
T	F	$-\$5 \times 10^6$
F	T	$\$5 \times 10^5$
F	F	$\$5 \times 10^5$

Figure 3.3: A parameterization of the influence diagram in Figure 3.2.

Up to this point, the above two scenarios are indistinguishable from the viewpoint of classical decision making tools. Remember that [KG09, BG11] remark that the budget for observations should be expended so long as the value of information of the observation is greater than the cost of observation. According to those selection criteria, both of the variables should thus be observed, as the expected financial gain could very well increase.

The SDP, however, finds these two scenarios very different. In particular, with respect to variables C and Q, the SDP is 60% in the first scenario and is 99% in the second scenario. That is, even though we stand to make a better decision in both scenarios upon observing variables C and Q (at least with respect to financial gain), and even though the expected benefit from such observations is the same in both scenarios, it is very unlikely that we would change the current decision of $I = F$ in the second scenario in comparison to the first. Hence, given the additional information provided by the SDP, a decision maker may act quite

using the reward function.

Q	$Pr(Q)$
T	0.1
F	0.9

Q	C	$Pr(S = T \mid .)$
T	T	0.05
T	F	0.98
F	T	0.01
F	F	0.05

C	$Pr(C)$
T	0.9
F	0.1

I	S	$u(I, S)$
T	T	$\$7 \times 10^7$
T	F	$-\$5 \times 10^6$
F	T	$\$5 \times 10^5$
F	F	$\$5 \times 10^5$

Figure 3.4: A parameterization of the influence diagram in Figure 3.2.

differently in these two scenarios. Indeed, when we take a closer look at the second scenario, there is a state of the world (when $S = T$) where deciding to invest would yield a very large financial gain. However, the chance of this state manifesting itself is extremely small (analogous to a lottery), meaning that a risk-conscious decision maker may be more averse to "gamble" in the second scenario and even waste resources to observe the variables C and D. Note that for this example we have assumed that the utility does not incorporate any "risk-factor", as if it did a rational decision maker would then always choose to gather more information despite a low probability of changing the current decision.

This illustrates the usefulness of SDP as a stopping criterion in the context of expected-utility decisions and influence diagrams. Namely, using SDP, we can distinguish between two very different scenarios, that are otherwise indistinguishable when we consider utilities alone.

CHAPTER 4

Computing the SDP

We have presented multiple examples in Chapter 3 that demonstrate that the SDP is useful as a stopping criterion for information gathering. We now discuss how to compute the SDP efficiently, as previous work only allowed for an approximate or brute-force solution [CXD12]. Notably, the SDP has been shown to be highly intractable, being PP^{PP}-complete,[1] by [CXD12], and the exact computation of the SDP has been limited to toy examples, with few variables, through brute-force enumeration. In this chapter , we propose the first exact algorithm for computing the SDP. This algorithm can be applied to real-world networks that are out of the scope of both brute-force enumeration and previously proposed approximation algorithms, and can be further applied to synthetic networks with as many as 100 variables. After presenting the algorithm, we discuss its complexity and present empirical results. The material in this chapter is based on the work published in [CCD13, CCD14].

4.1 Computing the Same-Decision Probability

Computing the SDP involves computing an expectation over the hidden variables $\mathbf{H}$. The naive brute-force algorithm would enumerate and check whether $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T$ for all instantiations $\mathbf{h}$ of $\mathbf{H}$. We now present an algorithm that can save us the need to explore every possible instantiation of $\mathbf{h}$. To make the algorithm

[1] The class PP^{PP} can be thought of as a counting variant of the NP^{PP} class, which contains the polynomial time hierarchy PH and for which the MAP problem is complete [PD04].

D	$Pr(D)$
$+$	0.3
$-$	0.7

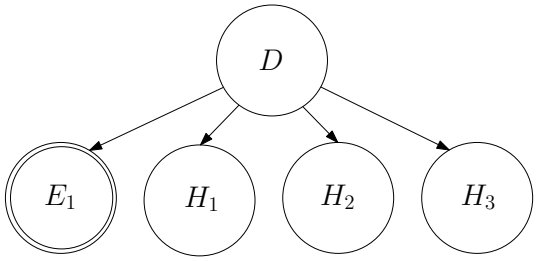


Figure 4.1: A Naive Bayes network (CPTs defined in Table 4.1).

easier to understand, we will first describe how to compute the SDP in a Naive Bayes network, where all the observations are probabilistically independent given the decision variable. This is no trivial problem — we show in Chapter 7 that even computing the SDP in a Naive Bayes network is NP-hard. We then generalize our algorithm to arbitrary networks.

4.1.1 Computing the SDP in Naive Bayes Networks

We will find it more convenient to implement the test $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T$ in the *log-odds* domain, where:

$$\log O(d \mid \mathbf{h}, \mathbf{e}) = \log \frac{Pr(d \mid \mathbf{h}, \mathbf{e})}{Pr(\overline{d} \mid \mathbf{h}, \mathbf{e})} \tag{4.1}$$

We then define the *log-odds threshold* as $\lambda = \log \frac{T}{1-T}$ and, equivalently, test whether $\log O(d \mid \mathbf{h}, \mathbf{e}) \geq \lambda$.

In a Naive Bayes network with D as the class variable, $\mathbf{H}$ and $\mathbf{E}$ as the leaf variables, and $\mathbf{Q} \subseteq \mathbf{H}$, the posterior log-odds after observing a *partial instantiation* $\mathbf{q} = \{h_1, \ldots, h_j\}$ can be written as:

$$\log O(d \mid \mathbf{q}, \mathbf{e}) = \log O(d \mid \mathbf{e}) + \sum_{i=1}^{j} w_{h_i} \tag{4.2}$$

where w_{h_i} is the *weight of evidence* h_i and defined as:

Table 4.1: CPTs for the network in Figure 4.1. Note that $Pr(H_3 \mid D)$, $Pr(E_1 \mid D)$ and $Pr(H_2 \mid D)$ are identical.

D	H_1	$Pr(H_1 \mid D)$
+	+	0.80
+	−	0.20
−	+	0.10
−	−	0.90

D	H_2	$Pr(H_2 \mid D)$
+	+	0.70
+	−	0.30
−	+	0.30
−	−	0.70

Table 4.2: Weights of evidence for the attributes in Figure 4.1.

i	w_{h_i}	$w_{\overline{h}_i}$
1	3.0	-2.17
2	1.22	-1.22
3	1.22	-1.22

$$w_{h_i} = \log \frac{Pr(h_i \mid d, \mathbf{e})}{Pr(h_i \mid \overline{d}, \mathbf{e})} \tag{4.3}$$

The weight of evidence w_{h_i} is then the contribution of evidence h_i to the quantity $\log O(d \mid \mathbf{q}, \mathbf{e})$ [CD03]. Note that all weights can be computed in time and space linear in $|\mathbf{H}|$ using a floating point representation.[2] Table 4.2 depicts the weights of evidence for the network in Figure 4.1.

One can then compute the SDP by enumerating the instantiations of variables $\mathbf{H}$ and then using Equation 4.2 to test whether $\log O(d \mid \mathbf{h}, \mathbf{e}) \geq \lambda$. Figure 4.2 depicts a search tree for the Naive Bayes network in Figure 4.1, which can be used for this purpose. The leaves of this tree correspond to instantiations $\mathbf{h}$ of variables $\mathbf{H}$. More generally, every node in the tree corresponds to an instantiation $\mathbf{q}$, where

[2]Additionally, note that for Equation 4.3, since in Naive Bayes networks H_i is d-separated from $\mathbf{E}$ given d, the term $\mathbf{e}$ can be dropped from the equation. We leave the term in because for general networks, H_i may not be d-separated from $\mathbf{E}$.

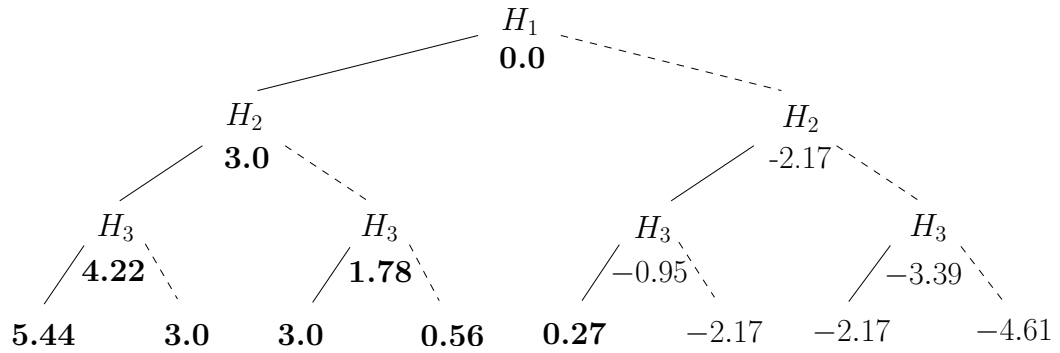


Figure 4.2: The search tree for the network of Figure 4.1. A solid line indicates + and a dashed line indicates −. The quantity $\log O(d \mid \mathbf{q}, \mathbf{e})$ is displayed next to each node $\mathbf{q}$ in the tree. Nodes with $\log O(d \mid \mathbf{q}, \mathbf{e}) \geq \lambda = 0$ are shown in bold.

$\mathbf{Q} \subseteq \mathbf{H}$.

A brute-force computation of the SDP would then entail:

1. Initializing the total SDP to 0.
2. Visiting every leaf node $\mathbf{h}$ in the search tree.
3. Checking whether $\log O(d \mid \mathbf{h}, \mathbf{e}) \geq \lambda$, and if so, adding $Pr(\mathbf{h} \mid \mathbf{e})$ to the total SDP.

Figure 4.2 depicts the quantity $\log O(d \mid \mathbf{q}, \mathbf{e})$ for each node $\mathbf{q}$ in the tree, indicating that five leaf nodes (i.e., five instantiations of variables $\mathbf{H}$) will indeed contribute to the SDP.

We now state the key observation underlying our proposed algorithm. Consider the node corresponding to instantiation $H_1 = +$ in the search tree, with $\log O(d \mid H_1 = +, \mathbf{e}) = 3.0$. All four completions $\mathbf{h}$ of this instantiation (i.e., the four leaf nodes below it) are such that $\log O(d \mid \mathbf{h}, \mathbf{e}) \geq \lambda = 0$. Hence, we really do not need to visit all such leaves and add their contributions $Pr(\mathbf{h}|\mathbf{e})$ individually to the SDP. Instead, we can simply add $Pr(H_1 = + \mid \mathbf{e})$ to the SDP, which equals the sum of $Pr(\mathbf{h} \mid \mathbf{e})$ for these leaves. More importantly, we can detect that all

such leaves will contribute to the SDP by computing a lower bound using the weights depicted in Table 4.2. That is, there are two weights for variable H_2, the *minimum* of which is -1.22. Moreover, there are two weights for variable H_3, the *minimum* of which is -1.22. Hence, the lowest contribution to the log-odds made by any leaf below node $H_1 = +$ will be $-1.22 - 1.22 = -2.44$. Adding this contribution to the current log-odds of 3.0 will lead to a log-odds of 0.56, which still passes the given threshold.

A similar technique can be used to compute upper bounds, allowing us to detect nodes in the search tree where no leaf below them will contribute to the SDP. Consider for example the node corresponding to instantiation $H_1 = -, H_2 = -$, with $\log O(d \mid H_1 = -, H_2 = -, \mathbf{e}) = -3.39$. Neither of the leaves below this node will contribute to the SDP as their log-odds do not pass the threshold. This can be detected by considering the weights of evidence for variable H_3 and computing the *maximum* of these weights (1.22). Adding this to the current log-odds of -3.39 gives -2.17, which is still below the threshold. Hence, no leaf node below $H_1 = -, H_2 = -$ will contribute to the SDP and this part of the search tree can also be pruned.

If we apply this pruning technique based on lower and upper bounds, we will actually end up exploring only the portion of the tree shown in Figure 4.3. The pseudocode of our final algorithm is shown in Algorithm 1. Note that it takes linear time to compute the upper and lower bounds. Additionally, note that the specific ordering of $\mathbf{H}$ in which the search tree is constructed is directly linked to the amount of pruning. We use an ordering heuristic that ranks each query variable H_i by the difference of its corresponding upper and lower bound — $\mathbf{H}$ is then ordered from greatest difference to lowest difference as to allow for earlier pruning.

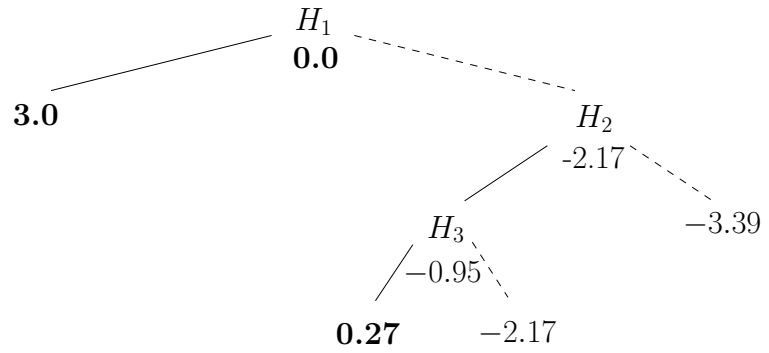


Figure 4.3: The reduced search tree for the network of Figure 4.1.

4.1.2 Computing the SDP in Arbitrary Networks

We will generalize our algorithm to arbitrary networks by viewing such networks as Naive Bayes networks but with aggregate attributes. For this, we first need the following notion.

Definition 3. *A partition of* $\mathbf{H}$ *given* D *and* $\mathbf{E}$ *is a set* $\mathbf{S}_1, \ldots, \mathbf{S}_k$ *such that:* $\mathbf{S}_i \subseteq \mathbf{H}$*;* $\mathbf{S}_i \cap \mathbf{S}_j = \emptyset$*;* $\mathbf{S}_1 \cup \ldots \cup \mathbf{S}_k = \mathbf{H}$*; and* $\mathbf{S}_i$ *is independent (d-separated) from* $\mathbf{S}_j$*,* $i \neq j$*, given* D *and* $\mathbf{E}$*.*

Figure 4.4 depicts an example partition.

The intuition behind a partition is that it allows us to view an arbitrary network as a Naive Bayes network, with class variable D and aggregate attributes $\mathbf{S}_1, \ldots, \mathbf{S}_k$. That is, each aggregate attribute $\mathbf{S}_i$ is viewed as a variable with states $\mathbf{s}_i$, allowing us to view each instantiation $\mathbf{h}$ as a set of values $\mathbf{s}_1, \ldots, \mathbf{s}_k$. We now have:

Proposition 1. *For a partial instantiation* $\mathbf{q} = \{\mathbf{s}_1, \ldots, \mathbf{s}_j\}$*,*

$$\log O(d \mid \mathbf{q}, \mathbf{e}) = \log O(d \mid \mathbf{e}) + \sum_{i=1}^{j} w_{\mathbf{s}_i}, \tag{4.4}$$

where

$$w_{\mathbf{s}_i} = \log \frac{Pr(\mathbf{s}_i, \mid d, \mathbf{e})}{Pr(\mathbf{s}_i \mid \overline{d}, \mathbf{e})} \tag{4.5}$$

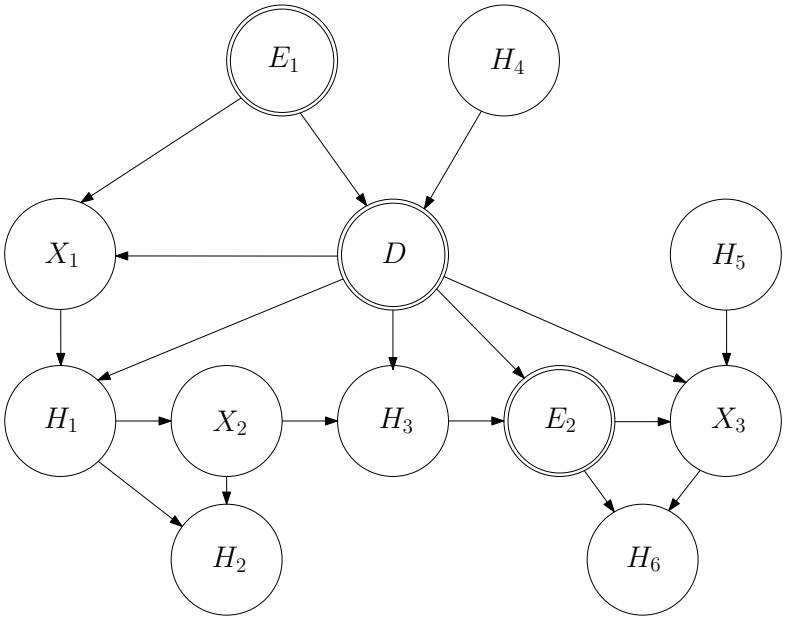


Figure 4.4: The partition of $\mathbf{H}$ given D and $\mathbf{E}$ is: $\mathbf{S}_1 = \{H_1, H_2, H_3\}$, $\mathbf{S}_2 = \{H_4\}$, $\mathbf{S}_3 = \{H_5, H_6\}$.

Proof.

$$\begin{aligned}
\log O(d \mid \mathbf{q}, \mathbf{e}) &= \log \frac{Pr(d \mid \mathbf{q}, \mathbf{e})}{Pr(\overline{d} \mid \mathbf{q}, \mathbf{e})} \\
&= \log \frac{Pr(d \mid \mathbf{e}) Pr(\mathbf{s}_1 \mid d, \mathbf{e}) \ldots Pr(\mathbf{s}_j \mid d, \mathbf{e})}{Pr(\overline{d} \mid \mathbf{e}) Pr(\mathbf{s}_1 \mid \overline{d}, \mathbf{e}) \ldots Pr(\mathbf{s}_j \mid \overline{d}, \mathbf{e})} \\
&= \log O(d \mid \mathbf{e}) + \sum_{i=1}^{j} w_{\mathbf{s}_i}
\end{aligned}$$

□

Since Equations 4.4 and 4.5 are analogous to Equations 4.2 and 4.3, we can now use Algorithm 1 on an arbitrary network. This usage, however, requires some auxiliary computations that were not needed or were readily available for Naive Bayes networks. We discuss these computations next.

4.1.2.1 Finding a Partition

We first need to compute a partition $\mathbf{S}_1, \ldots, \mathbf{S}_k$, which is done by pruning the network structure as follows: we delete edges outgoing from nodes in evidence $\mathbf{E}$ and hypothesis D, and delete (successively) all leaf nodes that are neither in $\mathbf{H}$, $\mathbf{E}$ or D. We then identify the components $\mathbf{X}_1, \ldots, \mathbf{X}_k$ of the resulting network and define each non-empty $\mathbf{S}_i = \mathbf{H} \cap \mathbf{X}_i$ as an element of the partition. This guarantees that in the original network structure, $\mathbf{S}_i$ is d-separated from $\mathbf{S}_j$ by D and $\mathbf{E}$ for $i \neq j$ (see [Dar09]). In Figure 4.4, network pruning leads to the components $\mathbf{X}_1 = \{X_1, X_2, E_2, H_1, H_2, H_3\}$, $\mathbf{X}_2 = \{D, E_1, H_4\}$ and $\mathbf{X}_3 = \{X_3, H_5, H_6\}$.

4.1.2.2 Computing Posterior Log-Odds, Probability and Weights of Evidence

The quantities $O(d \mid \mathbf{e})$, $Pr(\mathbf{q} \mid \mathbf{e})$ and $w_{\mathbf{s}_i}$, which are referenced on Lines 2, 3, and 7 of the algorithm, have simple closed forms in Naive Bayes networks. For arbitrary networks, however, computing these quantities requires inference which we do using the algorithm of variable elimination as described by [Dar09]. Note that the network pruning of deleting edges and removing the leaf nodes, as discussed above, guarantees that each factor used by variable elimination will have all its variables in some component $\mathbf{X}_i$. Hence, variable elimination can be applied to each component $\mathbf{X}_i$ in isolation, which is sufficient to obtain all needed quantities.

4.1.2.3 Computing the Min and Max of Evidence Weights

We finally show how to compute the upper and lower bounds, $\max_{\mathbf{s}_i} w_{\mathbf{s}_i}$ and $\min_{\mathbf{s}_i} w_{\mathbf{s}_i}$, which are referenced on Lines 2 and 3 of the algorithm. These quantities can also be computed using variable elimination, applied to each component $\mathbf{X}_i$ in isolation.

More precisely, for every component $\mathbf{X}_i$, we inject evidence $d, \mathbf{e}$ into its factors to yield a set of factors Ψ^i. Similarly, we inject evidence $\overline{d}, \mathbf{e}$ to yield a set of factors Φ^i. Summing out variables $\mathbf{X}_i \setminus \mathbf{S}_i$ from factors Ψ^i leads to a set of factors $\psi_1^i, \ldots, \psi_n^i$ whose product is the marginal $Pr(\mathbf{S}_i, d, \mathbf{e})$. Similarly, summing out variables $\mathbf{X}_i \setminus \mathbf{S}_i$ from factors Φ^i (using the same variable order) leads to a set of factors $\phi_1^i, \ldots, \phi_n^i$ whose product is the marginal $Pr(\mathbf{S}_i, \overline{d}, \mathbf{e})$. Since we used the same elimination order, there is a one-to-one correspondence between the two sets of resulting factors. Hence, we can now define a new set of factors $\chi_j^i = \psi_j^i / \phi_j^i$ for $j = 1, \ldots, n$. We finally calculate $w_{\overline{\mathbf{s}}_i}$ and $w_{\underline{\mathbf{s}}_i}$ using variable elimination, except that this time we maximize/minimize out variables $\mathbf{S}_i$:

$$w_{\overline{\mathbf{S}}_i} = \max_{\mathbf{S}_i} \prod_{j=1}^{n} \chi_j^i \qquad \text{and} \qquad w_{\underline{\mathbf{S}}_i} = \min_{\mathbf{S}_i} \prod_{j=1}^{n} \chi_j^i.$$

That is, the upper bounds $w_{\overline{\mathbf{s}}_i}$ can be recovered from factor $w_{\overline{\mathbf{S}}_i}$, while the lower bounds $w_{\underline{\mathbf{s}}_i}$ can be recovered from factor $w_{\underline{\mathbf{S}}_i}$.

This is very similar to the variable elimination algorithm used by [Dec99] in order to solve MAP, except that we perform both maximization and minimization simultaneously. Moreover, we do this on factors χ_j^i instead of factors ψ_j^i or ϕ_j^i.

4.1.3 Complexity Analysis

Let n be the number of variables in the network, $h = |\mathbf{H}|$, and $w = \max_i w_i$, where w_i is the width of constrained elimination order used on component $\mathbf{X}_i$. The best-case time complexity of our algorithm is then $O(n \exp w)$ and the worst-case time complexity is $O(n \exp (w + h))$. The intuition behind these bounds is that computing the maximum and minimum weights for each aggregate attribute takes time $O(n \exp w)$. This also bounds the complexity of computing $O(d|\mathbf{e})$, $Pr(\mathbf{q}|\mathbf{e})$ and corresponding weights $w_{\mathbf{s}_i}$. Moreover, depending on the weights and the threshold T, traversing the search tree can take anywhere from constant time to $O(\exp h)$. Since depth-first search can be implemented with linear space, the

space complexity is $O(n \exp w)$.

4.2 Experimental Results

We performed several experiments on both real and synthetic networks to test the performance of our algorithm across a wide variety of network structures, ranging from simple Naive Bayes networks to highly connected networks. Real networks were either learned from datasets provided by the UCI Machine Learning Repository [BL13] or provided by HRL Laboratories and CRESST.[3] The majority of the real networks used were diagnostic networks, which made it clear which variable should be selected as the decision variable as it would either be the "knowledge" or "fault" variable. For the unclear cases, the decision variable was picked at random. Both query and evidence variables were selected at random for all real networks.

Besides this algorithm, there are two other options available to compute the SDP: 1. the *naive* method to brute-force the computation by enumerating over all possible instantiations or 2. the *approximate* algorithm developed by [CXD12]. To compare our algorithm with these two other approaches, we compute the SDP over the real networks. For each network we selected at least 80% of the total network variables to be query variables so that we could emphasize how the size of the query set greatly influences the computation time. Each computation was given 20 minutes to complete. As we believe that the value of the threshold can greatly affect running time, we computed the SDP with thresholds $T = [0.01, 0.1, 0.2, \ldots, 0.8, 0.9, 0.99]$ and took the worst-case time. The results of our experiments with the three algorithms are shown in Table 4.3. Note that $|\mathbf{H}|$ is the number of query variables and $|\mathbf{h}|$ is the number of instantiations the naive algorithm must enumerate over. Moreover, ϕ indicates that the computation did not

[3] `http://www.cse.ucla.edu/`

Table 4.3: Algorithm comparison on real networks. We show the time, in seconds, it takes each algorithm, *naive*, *approx*, and *new* to compute the SDP in different networks. Note that ϕ indicates that the computation did not complete in the 20 minute time limit that we constrained. Moreover, * indicates that there was not sufficient memory to complete the computation.

Network	source	$\|\mathbf{H}\|$	$\|\mathbf{h}\|$	*naive*	*approx*	*new*
car	UCI	6	144	0.131	0.118	0.049
emdec6g	HRL	8	256	0.407	0.245	0.294
tcc4e	HRL	9	512	0.470	0.257	0.149
ttt	UCI	9	19683	6.234	0.133	0.091
caa	CRESST	14	16384	6.801	0.145	0.167
voting	UCI	16	65536	21.35	0.176	0.128
nav	CRESST	20	1572864	642.88	0.856	0.178
fire	CRESST	24	16777216	ϕ	0.183	0.508
chess	UCI	30	1610612736	ϕ	*	15.53

complete in the 20 minute time limit and * indicates that there was not sufficient memory to complete the computation. The networks {*car,ttt,voting,nav,chess*} are Naive Bayes networks whereas the networks {*caa,fire*} are polytree networks and the others are more general networks.

Given the real networks that we tested our algorithm on, it is clear that the algorithm outperforms both the naive implementation and the approximate algorithm for both Naive Bayes networks and polytree networks. Note that the approximation algorithm is based on variable elimination but can only use certain constrained orders. For a Naive Bayes network with hypothesis D being the root, the approximation algorithm will be forced to use a particularly poor ordering, which explains its failure on the `chess` network.

To analyze how a more general network structure and the selected threshold af-

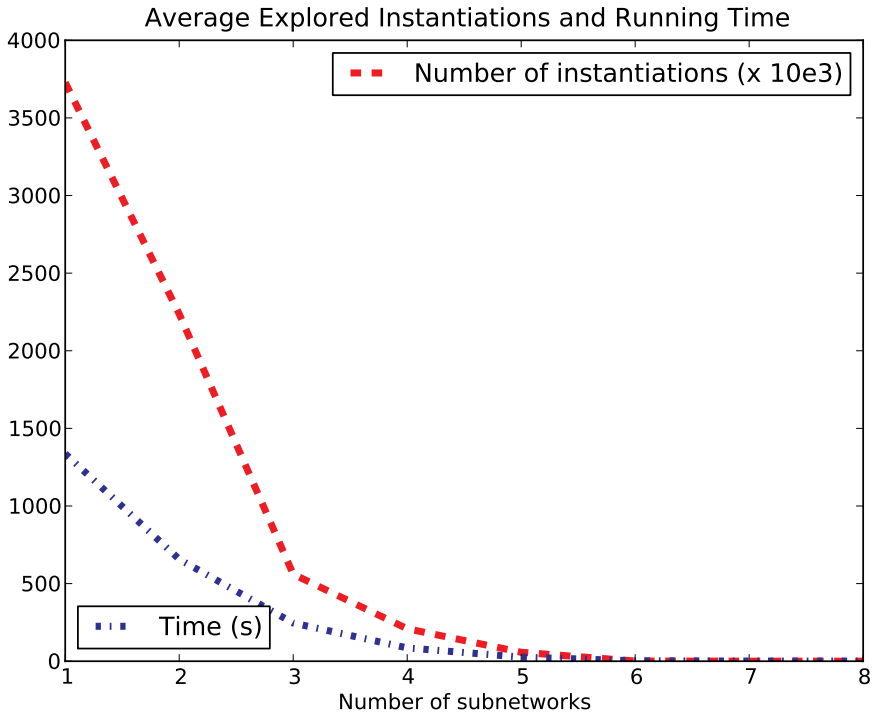


Figure 4.5: Synthetic network average running time and average number of instantiations explored by number of connected components.

fects the performance of our algorithm, we generated synthetic networks with 100 variables and varying treewidth using *BNGenerator* [ICR04]. For each network, we randomly selected the decision variable, 25 query variables, and evidence variables.[4] We then generated a partition for each network and grouped the networks by the size of obtained partition (k). Our goal was to test how our algorithm's running time and ability to prune the search-space depends on k. The average time and average number of instantiations explored are shown in Figure 4.5.

In general, we can see that as k increases, the number of instantiations explored by the algorithm decreases and its runtime improves. The network becomes more similar to a Naive Bayes structure with increasing k. Moreover, the larger k is, the more levels there are in the search tree, which means that our algorithm will

[4]As the synthetic networks are binary, a brute-force approach would need to explore 2^{25} instantiations.

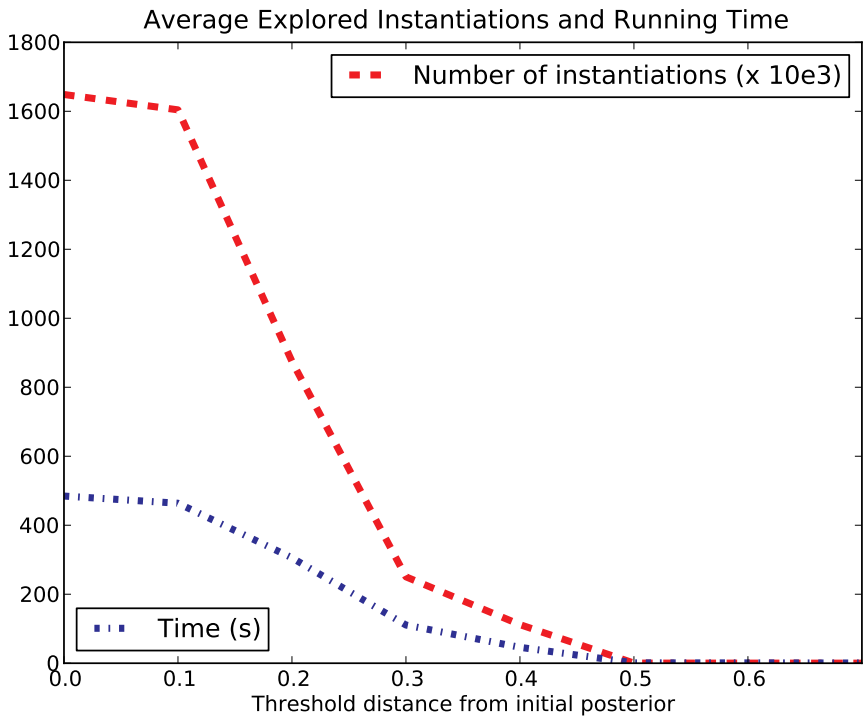


Figure 4.6: Synthetic network average running time and average number of instantiations explored by threshold distance from the initial posterior probability.

have more opportunities to prune. In the worst case, a network may be unable to be disconnected at all ($k = 1$). However, even in this case our algorithm is still, on average, more efficient compared to the brute-force implementation — for some cases, after computing the maximum and minimum weight of observing $\mathbf{H}$, it will find that there does not exist any $\mathbf{h}$ that will change the decision. We found that, given a time limit of 2 hours, the brute-force algorithm could not solve any synthetic networks, whereas our approach solved more than 70% of such networks.

We also test how the threshold affects computation time. Here, we calculate the posterior probability of the decision variable and then run repeatedly our algorithm with thresholds that are varying increments away. The average running time for all increments can be seen in Figure 4.6. It is evident that when the threshold is set to be further away from the initial posterior probability, the algorithm finishes much faster, which is perhaps expected since the usage of more

extreme thresholds would allow for more search space pruning.

Overall, our experimental results show that our algorithm is able to solve many SDP problems that are out of reach of existing methods. We also confirm that our algorithm completes much faster when the network can be disconnected or when the threshold is far away from the initial posterior probability of the decision variable.

Algorithm 1 Computing the SDP in a Naive Bayes network. Note: For $\mathbf{q} = \{h_1, \ldots, h_j\}$, $w_{\mathbf{q}}$ is defined as $\sum_{i=1}^{j} w_{h_i}$.

input:

$\mathcal{N}$: Naive Bayes network with class variable D

$\mathbf{H}$: attributes $\{H_1, \ldots, H_k\}$

λ: log-odds threshold

$\mathbf{e}$: evidence

output: Same-Decision Probability p

main:

global $p \leftarrow 0.0$ (initial probability)

$\mathbf{q} \leftarrow \{\}$ (initial instantiation is empty set)

$depth \leftarrow 0$ (initial depth of search tree)

`DFS_SDP`($\mathbf{q}$, $\mathbf{H}$, $depth$)

return p

1: **procedure** `DFS_SDP`($\mathbf{q}$, $\mathbf{H}$, $depth$)
2: $\quad UpperBound \leftarrow \log O(d \mid \mathbf{e}) + w_{\mathbf{q}} + \sum_{i=depth+1}^{k} \max_{h_i} w_{h_i}$
3: $\quad LowerBound \leftarrow \log O(d \mid \mathbf{e}) + w_{\mathbf{q}} + \sum_{i=depth+1}^{k} \min_{h_i} w_{h_i}$
4: $\quad$ **if** $(UpperBound < \lambda)$
5: $\quad\quad$ **return**
6: $\quad$ **else if** $(LowerBound \geq \lambda)$
7: $\quad\quad$ add $Pr(\mathbf{q} \mid \mathbf{e})$ to p,
8: $\quad\quad$ **return**
9: $\quad$ **else**
10: $\quad\quad$ **if** $depth < k$
11: $\quad\quad\quad$ **for** each value $h_{depth+1}$ of attribute $H_{depth+1}$ **do**
12: $\quad\quad\quad\quad$ `DFS_SDP`($\mathbf{q}h_{depth+1}$, $\mathbf{H} \setminus H_{depth+1}$, $depth + 1$)

CHAPTER 5

Using the SDP as a Selection Criterion

We have established that the SDP is useful as a stopping criterion as it allows us to calculate an exact measure of how robust our decision is against further observations. In this chapter, we show how the SDP can be used as a selection criterion for determining *which* observations should be selected, assuming that some stopping criterion (such as the SDP) indicates that further observations are necessary. Our goal is to select observations to maximize the robustness of any decision we will make post observation. Note that the material in this chapter is based on the work published in [CCD14, CCD15].

5.1 Decision Robustness: A Novel Selection Criterion

Our proposal is based on using VOI as the selection criterion (see Equation 2.4), while choosing the SDP as the reward function. We call this the *SDP gain*, and it is formally defined as:

Definition 4. *Given Definition 2 of an SDP, the* ***SDP gain*** *of observing variables* $\mathbf{G}$ *out of variables* $\mathbf{H}$ *is defined as the* ***expected SDP*** *of observing* $\mathbf{G} \subseteq \mathbf{H}$ *subtracted by the SDP over* $\mathbf{H}$*:*

$$\mathcal{G}(\mathbf{G}) = \mathcal{E}(\mathbf{G}, \mathbf{H}, \mathbf{e}, T) - SDP(d, \mathbf{H}, \mathbf{e}, T), \tag{5.1}$$

where the expected SDP, also referred to as the ***Decision Robustness****, is defined as:*

$$\mathcal{E}(\mathbf{G}, \mathbf{H}, \mathbf{e}, T) = \sum_{\mathbf{g}} SDP(d, \mathbf{H} \setminus \mathbf{G}, \mathbf{ge}, T) Pr(\mathbf{g} \mid \mathbf{e}), \tag{5.2}$$

and d is defined as the decision made given the current evidence.

We can see that maximizing the SDP gain is equivalent to maximizing the expected SDP, which we often refer to as *Decision Robustness (DR)*. Note that if we observe some variables $\mathbf{G} \subseteq \mathbf{H}$ such that the expected SDP is 1.0, that indicates that after observing $\mathbf{G}$ and making a decision, the remaining variables $\mathbf{H} \setminus \mathbf{G}$ will be rendered completely redundant — their observation will have no effect on the decision. The expected SDP (E-SDP) of features $\mathbf{G}$ can be thought of as a measure for the similarity between the decision made after observing $\mathbf{G} \subseteq \mathbf{H}$ and the one made after observing all of $\mathbf{H}$. If the E-SDP of a set $\mathbf{G}$ is high, we know that after observing $\mathbf{G}$, there is little chance that observing $\mathbf{H} \setminus \mathbf{G}$ will change our decision. This means that observing $\mathbf{G}$ will lead to high Decision Robustness. The goal of using Decision Robustness as a selection criterion is to observe those variables which, on average, will allow for the most stable decision given the collected observations. Note that per Equation 2.4, maximizing the Decision Robustness is **equivalent** to computing the VOI with the SDP as a reward function.

To demonstrate its effectiveness, we will next provide an example of using SDP as a selection criterion, contrasting it with two other selection criteria: One based on reducing entropy of the hypothesis variable D, and another based on maximizing the gap between the decision probability $Pr(d \mid \mathbf{e})$ and the given threshold T [KG09]. While both criteria can be motivated as reducing uncertainty, we show that both can indeed lead to less stable decisions than if the SDP were to be used.

The example is given by the Bayesian network in Figure 5.1, where D is the hypothesis variable and S_1/S_2 are sensors. A decision is triggered when $Pr(D =$

D	$Pr(D)$
$+$	0.5
$-$	0.5

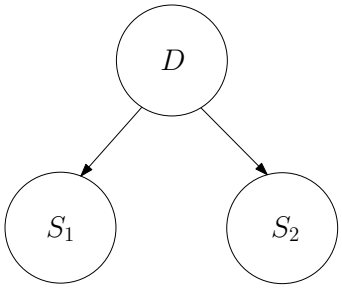


Figure 5.1: A Bayesian network with its CPTs given in the Table 5.1.

Table 5.1: CPTs for the Bayesian network in Figure 5.1.

D	S_1	$Pr(S_1 \mid D)$
$+$	$+$	0.8
$+$	$-$	0.2
$-$	$+$	0.2
$-$	$-$	0.8

D	S_2	$Pr(S_2 \mid D)$
$+$	$+$	0.75
$+$	o	0.2
$+$	$-$	0.05
$-$	$+$	0.05
$-$	o	0.2
$-$	$-$	0.75

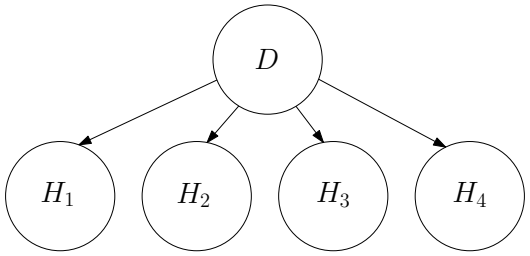
$+ \mid \mathbf{e}) \geq 0.80$, where evidence $\mathbf{e}$ is over sensors S_1 and S_2. With no observations (empty evidence $\mathbf{e}$), the SDP is 0.595, suggesting that further observations may be needed. Assuming a limited number of observations [HBR95], and using a myopic approach of observing one variable at a time [DJ97], we now need to select the next variable to observe.

Note that maximizing VOI with negative entropy as the reward function amounts to maximizing mutual information, as $H(D, X) = H(D) - H(D \mid X)$ [CT91, KG05]. The mutual information between variable D and sensor S_2 is 0.53 whereas the mutual information between D and sensor S_1 is 0.278. Hence, observing S_2 will reduce the entropy of D the most. In terms of margin of confidence, another reward function used by [KG09], observing S_2 will on average lead

to a 0.7 margin between the states $D = +$ and $D = -$, whereas observing S_1 will only lead to a 0.6 margin between the two states.

However, if we compute the corresponding SDP gains, $\mathcal{G}(S_1)$ and $\mathcal{G}(S_2)$, we find that observing S_1 will, on average, lead to improving the decision stability the most. In particular, observing S_1 would give us an SDP of either 1 or 0.81—for an expected SDP of 0.905, whereas observing S_2 would give us an SDP of either 0.7625, 0.5, or 1—for a Decision Robustness level of 0.805. Therefore, $\mathcal{G}(S_1) = 0.31$ and $\mathcal{G}(S_2) = 0.21$. Hence, observing S_1 will on average allow us to make a decision that is less likely to change due to additional information (beyond S_1).

Some intuition to why this occurs is that although observing S_2 leads to greater information gain than observing S_1, it is *superfluous information*. Note that $Pr(D = + \mid S_2 = -) = 0.0625$, whereas $Pr(D = + \mid S_1 = -) = 0.2$. Clearly we can see that observing S_2 can lead to a more skewed distribution with minimal conditional entropy. However, in the context of threshold-based decisions, we make a decision based solely on whether $Pr(D = + \mid \mathbf{e})$ is above or below the threshold, meaning that we may not put as much emphasis on how much below or above the threshold $Pr(D = + \mid \mathbf{e})$ is. In this case, although observing S_2 can on average lead to a more extreme distribution, observing $S_2 = \mathsf{o}$ leads to making an extremely nonrobust decision (a decision that would change 50% of the time with observation of S_1). Observing S_1 before making a decision leads to a much more robust decision. This example demonstrates the usefulness of SDP as a selection criterion for threshold-based decisions, as the SDP can be used to select observations that lead to more robust decisions.



D	$Pr(D)$
$+$	0.50
$-$	0.50

Figure 5.2: A Naive Bayes network.

5.2 Selection with a Constrained Budget

In many domains we often have a limited budget which we may expend on information gathering, and differing feature costs for each observation [KG09, BG11] We now provide another example that demonstrates the use of maximizing the SDP gain in this context — we show that compared to other criteria, our criterion will allow us to gather fewer features while making a decision that is maximally robust against any further observations.

Consider a probabilistic graphical model with some features $F_1, \ldots, F_n$ that can be observed at a cost. Consider also a decision variable D in the model, whose posterior distribution will be used to make a decision. Given a budget on observations, a classical problem is that of finding a set of features $\mathbf{G}$ from $F_1, \ldots, F_n$, where the *cost* of observing $\mathbf{G}$ is within budget, and the *value* obtained from observing $\mathbf{G}$ is maximal. This problem appears in a variety of applications, including active sensing [GGR02, GK11a], medical diagnosis [YKR09], fault identification [BSB13], knowledge assessment [MS08, MDC13], and computer vision [AY13].

Consider the Naive Bayes network in Figure 5.2, which will serve as a running example. Here, D is a decision variable and H_1, H_2, H_3, H_4 are tests that bear on this decision. The decision threshold is 0.80, leading to a positive decision if $Pr(D{=}+ \mid \mathbf{e}) \geq 0.8$.[1]

We are interested in the following problem: given a budget, which features

[1]This threshold is often used in educational and medical diagnosis [VN01, CWM14].

D	H_1	$Pr(H_1 \mid D)$
+	+	0.508
+	−	0.482
−	+	0.016
−	−	0.984

D	H_3	$Pr(H_3 \mid D)$
+	+	0.61
+	−	0.39
−	+	0.39
−	−	0.61

D	H_2	$Pr(H_2 \mid D)$
+	+	0.80
+	−	0.20
−	+	0.20
−	−	0.80

D	H_4	$Pr(H_4 \mid D)$
+	+	0.543
+	−	0.456
−	+	0.456
−	−	0.543

Figure 5.3: Additional CPTs for network in Figure 5.2.

should we observe? Remember that feature selection is typically based on maximizing the *value of information (VOI)*, where the reward may be measured with different criteria, the most prevalent criterion being information gain. The information gain and costs for the different features in Figure 5.2 are shown in Table 5.2.

Feature	Cost ($)	IG
H_1	4.0	0.2708
H_2	4.0	0.2781
H_3	1.0	0.0350
H_4	1.0	0.0054

Table 5.2: Costs and information gain for various features.

Suppose that we have a budget of $5 to expend. Using information gain as the reward criterion, features H_2 and H_3 would be the optimal features to observe as they maximize information gain (with a combined gain of 0.2815) while being

within budget. Note that myopic (greedy) feature selection [YKR09, GK11a] agrees with non-myopic (optimal) feature selection [KG09, BG11] in this example.

Let us now apply the SDP in order to analyze this scenario. First, we want to see how robust our initial decision is (when we make a decision with no features observed at all). Intuitively, we can compute the SDP by considering every possible instantiation $\mathbf{h}$ of features $\mathbf{H}$, computing the decision under each instantiation, and then adding up the probability of each instantiation $\mathbf{h}$ that leads to the same decision (i.e., $Pr(d \mid \mathbf{h}) < 0.8$). The process of enumerating instantiations is shown in Table 5.3, leading to an SDP of 0.738. Hence, the probability of reaching a different decision after observing features $\mathbf{H} = H_1, \ldots, H_4$ is only 0.262, meaning that our decision is very prone to change, so further features should be observed in order to ensure a more robust decision.

If we were to maximize the SDP gain in this example (e.g. maximize the VOI with SDP the reward criterion), we would observe feature H_1 instead of both H_2 and H_3 (which would be selected by information gain). In particular, if we observe $H_1{=}+$, the decision made will not change regardless of what the other features are observed to be — since $Pr(D{=}+ \mid H_1{=}+, h_2, h_3, h_4)$ will always be greater than 0.8. Technically, this means that the SDP after observing $H_1{=}+$ is 1.0. Similarly, if we observe $H_1{=}-$, the SDP will be 1.0 as well. As such, the *expected SDP* is 1.0 after observing H_1. This means that *whatever decision* we make after observing H_1, that decision stays the same *after* observing H_2, H_3 and H_4. By a similar calculation, the expected SDP of observing both H_2 and H_3 is 0.697. This shows that maximizing information gain does not necessarily maximize decision robustness. This is an extreme example, which is meant to highlight the proposed criterion. More generally, however, we seek to identify a set of features that maximize the expected SDP, even though the expected SDP may not necessarily be 1.0.

We have thus shown the benefit of finding the set of features that maximize

H_1	H_2	H_3	H_4	$Pr(\mathbf{h} \mid \mathbf{e})$	$Pr(d \mid \mathbf{h}, \mathbf{e})$
+	+	+	+	0.0676	**0.995**
+	+	+	−	0.0430	**0.989**
+	+	−	+	0.0570	**0.994**
+	+	−	−	0.0366	**0.985**
+	−	+	+	0.0179	**0.937**
+	−	+	−	0.0125	**0.858**
+	−	−	+	0.0155	**0.913**
+	−	−	−	0.0111	**0.810**
−	+	+	+	0.0828	0.788
−	+	+	−	0.0690	0.602
−	+	−	+	0.0757	0.725
−	+	−	−	0.0676	0.517
−	−	+	+	0.0863	0.189
−	−	+	−	0.1202	0.086
−	−	−	+	0.0969	0.141
−	−	−	−	0.1393	0.062

Table 5.3: Scenarios $\mathbf{h}$ for the network in Figure 5.2. The cases where $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq 0.8$ are bolded. Note that whether $H_1 = +$ or $H_1 = -$ is entirely indicative of whether $Pr(d \mid \mathbf{h}, \mathbf{e}) \geq 0.8$.

the E-SDP. It should be clear that choosing features based on maximizing the E-SDP amounts to maximizing an expected reward, where $SDP(d, \mathbf{H} \setminus \mathbf{G}, \mathbf{ge}, T)$ is the reward function. In the next chapter, we provide an algorithm for identifying the subset $\mathbf{G}$ of hidden features that maximizes decision robustness, and thus *maximizing* the expected SDP that we can see with respect to some budget.

CHAPTER 6

Maximizing the Expected SDP

We have presented multiple examples in Chapter 5 that demonstrate how the SDP can be used as a selection criterion for information gathering to maximize Decision Robustness. In this chapter, we present a novel algorithm that we developed, `MaxDR` that selects features to *maximize* the *expected* SDP. We first discuss the technical details of `MaxDR` and subsequently present empirical results for when `MaxDR` is practically applied to classification and decision making tasks. In particular, we show that `MaxDR` can be used to tradeoff the expended budget with classification accuracy (`MaxDR` can reduce the expended budget significantly, while reducing the classification accuracy only slightly). For decision making tasks, we show that `MaxDR` leads to much more robust decisions, under a limited budget, compared to some traditional VOI criteria. This makes `MaxDR` particularly suitable to applications where the decision maker wishes to reduce the liability of their decisions against unknown information. Note that the material in this chapter is based on the work published in [CCD15].

6.1 MaxDR: Maximizing Decision Robustness

We restrict our attention to Naive Bayes networks, as computing the SDP is PP^{PP}-complete for Bayesian networks [CXD12] but only NP-hard for Naive Bayes networks, which we subsequently prove in Chapter 7.

Given a Naive Bayes network $\mathcal{N}$ with class variable D, threshold T, observation

e, unobserved features **H**, a cost C_i for each individual feature H_i, and a total budget B, our algorithm `MaxDR` will output a set of features $\mathbf{G}^\star \subseteq \mathbf{H}$ where

- the cost of observing $\mathbf{G}^\star$ is within the budget B:

$$\big(\sum_{H_i \in \mathbf{G}^\star} C_i\big) \leq B,$$

- the E-SDP of observing $\mathbf{G}$ is maximal:

$$\mathbf{G}^\star = \arg\max_{\mathbf{G} \subseteq \mathbf{H}} \mathcal{E}(D, \mathbf{G}, \mathbf{H}, \mathbf{e}, T).$$

Intuitively, observing the set of features identified by `MaxDR` will best render the remaining features redundant. Note the similarity between our problem and the 0-1 **knapsack problem:** we have a set of items, each with an associated cost and profit, and a budget B, where our goal is to maximize the profit while keeping the total cost within budget B. The knapsack problem is difficult: the decision problem is NP-complete [KPP04]. There is a key difference, however, between our problem and the typical knapsack problem: computing the total profit of a set for the knapsack problem can be done in linear time, while computing the total profit of a set for our problem involves computing E-SDP, which is NP-hard even in Naive Bayes networks.[1]

We will first show how to compute E-SDP, and then show how to solve the knapsack component of the problem.

6.1.1 Computing the Expected SDP.

Computing the E-SDP of $\mathbf{G} \subseteq \mathbf{H}$ involves the test $Pr(d \mid \mathbf{h}, \mathbf{e}) =_T Pr(d \mid \mathbf{g}, \mathbf{e})$. That is, we test whether both posterior probabilities are on the "same side" of

[1] We prove in Chapter 7 that computing the SDP in Naive Bayes networks is NP-hard. Computing E-SDP is then NP-hard as well since computing the SDP is just a special case of computing the E-SDP where $\mathbf{G} = \{\}$.

threshold T. Once again, akin to Chapter 4.1, we find it more convenient to implement this test in the *log-odds* domain, where

$$\log O(d \mid \mathbf{h}, \mathbf{e}) = \log \frac{Pr(d \mid \mathbf{h}, \mathbf{e})}{Pr(\overline{d} \mid \mathbf{h}, \mathbf{e})},$$

and then subsequently we define the *log-odds threshold* as

$$\lambda = \log \frac{T}{1-T},$$

and finally, we have the log-odds test:

$$\log O(d \mid \mathbf{h}, \mathbf{e}) =_{\lambda} \log O(d \mid \mathbf{g}, \mathbf{e}).$$

In a Naive Bayes network we have D as the class variable and $\mathbf{H} \cup \mathbf{E}$ as the leaf variables. The posterior log-odds after observing an arbitrary *partial instantiation* $\mathbf{q} = \{h_1, \ldots, h_j\}$ of variables $\mathbf{Q} \subseteq \mathbf{H}$ can be written as

$$\log O(d \mid \mathbf{q}, \mathbf{e}) = \log O(d \mid \mathbf{e}) + \sum_{i=1}^{j} w_{h_i},$$

where w_{h_i} is the *weight of evidence* h_i and defined as:

$$w_{h_i} = \log \frac{Pr(h_i \mid d)}{Pr(h_i \mid \overline{d})}.$$

The weight of evidence w_{h_i} is then the contribution of evidence h_i to the quantity $\log O(d \mid \mathbf{q}, \mathbf{e})$ [CD03]. Note that all weights can be computed in time and space linear in $|\mathbf{H}|$ (for Naive Bayes networks). For our example network shown in Figure 5.2, the weights of evidence $\{w_{h_i}, w_{\overline{h}_i}\}$ are shown in Table 6.1.

i	1	2	3	4
w_{h_i}	2.00	5.00	0.65	0.25
$w_{\overline{h}_i}$	-2.00	-1.00	-0.65	-0.25

Table 6.1: Weights of evidence.

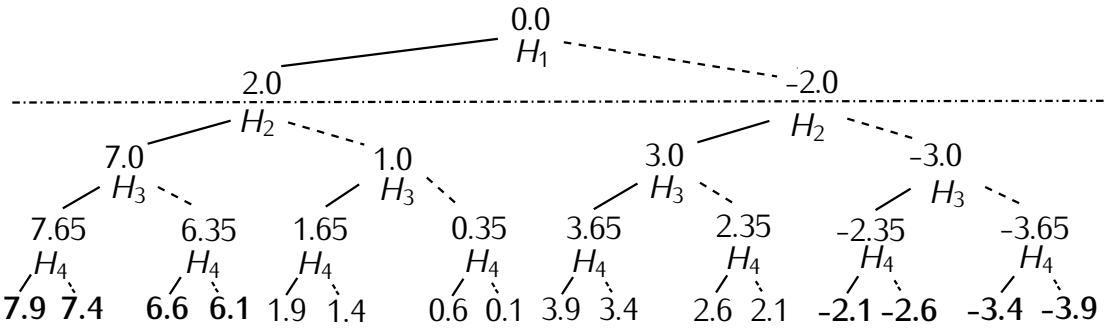


Figure 6.1: This figure depicts a two-tiered search tree to compute the E-SDP with $\mathbf{G} = \{H_2\}$ and $\lambda = 2.0$ for the network shown in Figure 5.2. A solid line indicates + and a dashed line indicates −. Note the horizontal line dividing $\mathbf{G}$ and the remaining features. The quantity $\log O(d \mid \mathbf{q}, \mathbf{e})$ is displayed next to each node $\mathbf{q}$ in the tree. Leaf nodes where $\log O(d \mid \mathbf{h}, \mathbf{e}) =_\lambda \log O(d \mid \mathbf{g}, \mathbf{e})$ are bolded.

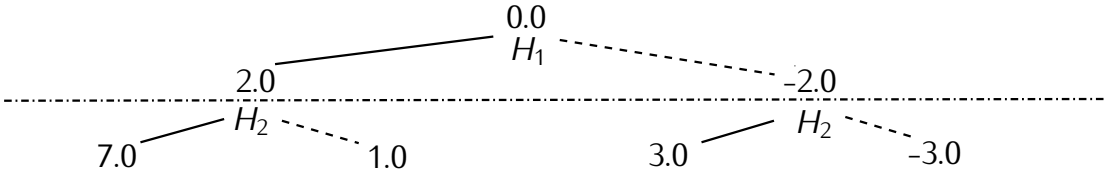


Figure 6.2: The pruned search tree when computing the E-SDP for $\mathbf{G} = \{H_1\}$.

For convenience, we also define the quantities UB and LB that respectively represent the upper and lower bounds on the total weight of evidence of observing variables $\mathbf{Q}$:

$$UB(w_{\mathbf{Q}}) = \sum_{H_i \in \mathbf{Q}} \max_{h_i} w_{h_i}$$

$$LB(w_{\mathbf{Q}}) = \sum_{H_i \in \mathbf{Q}} \min_{h_i} w_{h_i}$$

One brute-force method to compute the E-SDP is similar to the brute force method discussed in Chapter 4.1 for computing the SDP, where we first initialize the total SDP to 0, then enumerate the instantiations of $\mathbf{H}$ into a search tree, then check whether $\log O(d \mid \mathbf{h}, \mathbf{e}) =_\lambda \log O(d \mid \mathbf{e})$, and if so, add $Pr(\mathbf{h} \mid \mathbf{e})$ to the total SDP.

Our proposed algorithm, `CalcESDP`, utilizes a two-tiered search tree. The first

tier of the search tree includes $\mathbf{G}$, whereas the second tier includes $\mathbf{H} \setminus \mathbf{G}$. An example two-tiered search tree with $\mathbf{G} = \{H_1\}$ can be seen in Figure 6.1. To reduce the number of nodes that must be explored, we use a branch-and-bound approach similar to Algorithm 1 to detect when the children of an intermediate node can be pruned. Consider the node corresponding to instantiation $\{H_1 = +, H_2 = +\}$ in the search tree, with $\log O(d \mid H_1 = +, H_2 = +, \mathbf{e}) = 7.0$. All four completions $\mathbf{h}$ of this instantiation (i.e., the four leaf nodes below it) are such that $\log O(d \mid \mathbf{h}, \mathbf{e}) =_\lambda \log O(d \mid H_1 = +, \mathbf{e})$. Therefore, we can avoid visiting all such leaves and adding their contributions $Pr(\mathbf{h} \mid \mathbf{e})$ individually to the expected SDP. Instead, we simply add $Pr(H_1 = +, H_2 = + \mid \mathbf{e})$ to the SDP, which equals the sum of $Pr(\mathbf{h} \mid \mathbf{e})$ for these leaves. More importantly, we can detect that all such leaves will contribute to the SDP by computing a lower bound using the weights depicted in Table 6.1. Note that for variables H_3 and H_4 we can compute a lower bound — the lowest contribution to the log-odds made by any leaf below $\{H_1 = +, H_2 = +\}$ in the search tree will be -0.90. Adding this contribution to the current log-odds of 7.0 will lead to a log-odds of 6.10, which still is $=_\lambda \log O(d \mid H_1 = +, \mathbf{e})$.

In addition, our algorithm uses a novel *dynamic* variable order that leads to significant improvements to the amount of pruning. The pseudocode of `CalcESDP` is shown in Algorithm 2. When applied to the previous example, the algorithm will only explore the tree in Figure 6.2.

Note that the dynamic ordering heuristic (Line 20) will select the variable that upon observation, can maximize the margin between the current posterior log-odds and the log threshold. For instance, in our running example, if $\log O(d \mid \mathbf{e}) = -2$, then H_1 would be selected, as the observation of $H_1 = -$ would result in a minimal posterior log-odds of -4. Whereas if $\log O(d \mid \mathbf{e}) = 1$, then H_2 would be selected as the observation of $H_2 = +$ would result in a maximal posterior log-odds of 6. This allows us to first explore certain parts of the search tree where we can aggressively prune. Note that we keep track of the upper bound on the final value

of the E-SDP. This quantity, as well as our dynamic ordering heuristic, will be further discussed in the next section.

Algorithm 2: CalcESDP Computing E-SDP in a Naive Bayes network with class variable D, evidence $\mathbf{e}$, and attributes $\mathbf{H} = \{H_1, \ldots, H_n\}$. Note $w_{\mathbf{q}} = \sum_{h_i \in \mathbf{q}} w_{h_i}$.

input:

$\mathcal{N}$: Naive Bayes network

$\mathbf{G}$: a subset of $\mathbf{H}$

λ: log-odds threshold

output: Expected Same-Decision Probability p

main:

$p \leftarrow 0.0$ *a lower-bound on E-SDP*

$m \leftarrow 1.0$ *an upper-bound on E-SDP*

$\mathbf{q} \leftarrow \{\}$ *initial instantiation (empty)*

`DFS_E_SDP`($\mathbf{q}$)

return p *$p = m$ on return*

```
1:  procedure DFS_E_SDP(q)
2:      t ← log O(d | e) + w_q                          current log-odds
3:      if |Q| ≤ |G|                                    enumerating top-half of tree
4:          if t + UB(w_{H\Q}) < λ or λ ≤ t + LB(w_{H\Q})
5:                                                      decision is now fixed
6:              p ← p + Pr(q | e)
7:              return
8:      else                                            searching bottom-half of tree
9:          if t + UB(w_{H\Q}) < λ or λ ≤ t + LB(w_{H\Q})
10:                                                     decision is now fixed
11:             if t =_λ log O(d | e) + w_g             same decision
12:                 p ← p + Pr(q | e)
13:             else                                    different decision
14:                 m ← m − Pr(q | e)
15:             return
```

16: **if** $|\mathbf{Q}| < |\mathbf{H}|$ *continue traversing tree*

17: $H_i \leftarrow$ `NEXT_VAR`$(\mathbf{q}, t)$ *pick next variable*

18: **for** each value h_i of attribute H_i **do**

19: `DFS_E_SDP`$(\mathbf{q}, h_i)$ *recurse*

20: **procedure** `NEXT_VAR`$(\mathbf{q}, t)$

21: $\mathbf{C} \leftarrow \mathbf{H} \setminus \mathbf{Q}$ *candidate variables*

22: **if** $|\mathbf{Q}| < |\mathbf{G}|$ *if enumerating top half*

23: $\mathbf{C} \leftarrow \mathbf{C} \setminus \mathbf{H}$ *remove bottom half*

24: **return** $\arg\max_{H_i \in \mathbf{C}} |(t + w_{h_i \in H_i} - \lambda)|$

6.1.2 Finding Valid Sets of Features.

Feature selection algorithms usually exploit submodularity and/or monotonicity [KG05, KG09, GK11b]. However, as the expected SDP is neither monotonic, nor submodular (shown in Chapter 5.2), a novel approach must be devised. A brute-force computation enumerates all $2^{|\mathbf{H}|}$ sets of features and then computes E-SDP for each set that respects the budget.

We can greatly improve upon this method using a key observation: if observing variables $\mathbf{G} \subseteq \mathbf{H}$ cannot change the original decision, then the E-SDP of observing variables $\mathbf{G}$ is *the same* as the original SDP with respect to variables $\mathbf{H}$ (without the expectation). Hence, we need only compute this "expected" SDP once, and focus only on those subsets $\mathbf{G}$ that are not so vacuous.

More precisely, if our decision is initially below the threshold, $\log O(d \mid \mathbf{e}) < \lambda$, then we only need to consider sets of features $\mathbf{G}$ that are able to cross the threshold: $\lambda \leq \log O(d \mid \mathbf{e}) + UB(w_{\mathbf{G}})$. Similarly, if our decision is initially above the threshold, i.e., $\lambda \geq \log O(d \mid \mathbf{e})$, then we only need to consider features $\mathbf{G}$ that satisfy $\log O(d \mid \mathbf{e}) + LB(w_{\mathbf{G}}) \leq \lambda$. Hence, to obtain a non-trivial expected SDP

problem, we must select enough features in $\mathbf{G}$ so that:

$$UB(w_{\mathbf{G}}) = \sum_{H_i \in \mathbf{G}} \max_{h_i} w_{h_i} \geq \lambda - \log O(d \mid \mathbf{e}), \tag{6.1}$$

if our decision is initially below the threshold, or else if the decision is initially above the threshold:

$$LB(w_{\mathbf{G}}) = \sum_{H_i \in \mathbf{G}} \min_{h_i} w_{h_i} \leq \lambda - \log O(d \mid \mathbf{e}). \tag{6.2}$$

We can find these sets by solving the following variant of the knapsack problem:

Definition 5. *Consider n items X_1 to X_n, where each item has cost* $\mathsf{cost}(X_i)$ *and profit* $\mathsf{profit}(X_i)$*. The Knapsack Problem asks: "Is there a subset of items with total cost at most B and total profit at least V?"*

Here, our cost function is the same cost as in our feature selection, but our profit function is either $\mathsf{profit}(H_i) = \max_{h_i} w_{h_i}$ or $\mathsf{profit}(H_i) = \min_{h_i} w_{h_i}$ (depending on the initial decision). We then want subsets of items (features) that yield enough profit for our decision to cross its threshold, and hence yield a non-trivial expected SDP problem. Our proposed algorithm, `FindCands`, uses a branch-and-bound approach similar to [Pis95, ZBB10], and builds an inclusion/exclusion tree, as done by [Kor09], in order to find all possible solutions.

In order to increase the amount of pruning that can be done, we build the inclusion/exclusion tree of items by *efficiency*, defined as the ratio $\frac{\mathsf{profit}(X_i)}{\mathsf{cost}(X_i)}$. After the items are sorted, and as we traverse the tree, we keep track of (1) all costs accrued (by the features currently selected), (2) and the remaining profit potential that is available (that we could get by selecting from the remaining features). We backtrack if we find that (1) the budget has been surpassed, or (2) the profit potential is too low, which is based on an efficiently computable upper bound on our profit, UBP — we add all items as allowed by the budget in order of decreasing efficiency. At the point where the next item's cost exceeds the remaining budget,

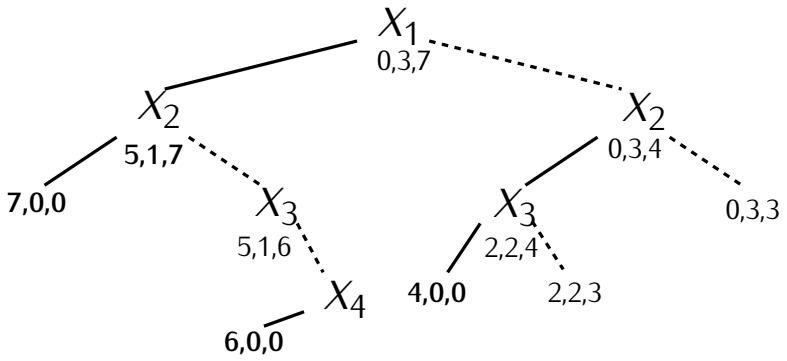


Figure 6.3: A pruned inclusion/exclusion tree for the knapsack problem instance shown in Table 6.2. Note that a left branch corresponds to the *inclusion* of an item whereas a right branch corresponds to the *exclusion* of an item. At every node there is a tuple of 3 items: the profit so far, the budget remaining, and an upper bound for the remaining profit (UBP). Due to pruning, we avoid traversing the full inclusion/exclusion tree.

we take, according to the remaining budget, the fraction of the item's profit, which yields an upper bound on the maximum obtainable profit [KPP04, ZBB10].

The pseudocode for `FindCands` can be found in Algorithm 3. In Figure 6.3, we provide an example of the inclusion/exclusion tree after using `FindCands` for the following knapsack problem instance found in Table 6.2. The candidate knapsack items include X_1, X_2, X_3, X_4, and their respective {`profit`, `cost`} values can be found in Table 6.2. The final algorithm `MaxDR` to find the feature set $\mathbf{G}^\star$ with highest expected SDP is: use `FindCands` to generate valid sets, and for each candidate set $\mathbf{G}$, use `CalcESDP` to compute the expected SDP. Note that `CalcESDP` maintains an upper bound for the current value of the expected SDP — if that value falls below the highest expected SDP of some previous candidate set, we can abort the computation and continue our traversal through the inclusion/exclusion tree. This is where the previously introduced dynamic ordering heuristic is especially useful, as it allows us to swiftly detect when the upper bound falls below some threshold.

	X_1	X_2	X_3	X_4
Profit	5.0	2.0	2.0	1.0
Cost	2.0	1.0	2.0	1.0

Table 6.2: A knapsack problem instance with budget = 3 and profit = 4.

6.1.3 Complexity analysis

Let n be the number of features in the network, where c is the number of states of a feature variable. In the best-case scenario, we can detect that the E-SDP will never change regardless of what instantiation of features is observed — in this trivial scenario, no search needs to be performed. However, for the worst-case time complexity, we may need to traverse all of the $O(2^n)$ possible candidate subsets of features and take $O(c^n)$ time to compute the E-SDP of each subset. Therefore, the worst-case time complexity is is $O(c^n 2^n)$.

6.2 Applications and Experimental Results

We now discuss the application of our novel feature selection algorithm, `MaxDR` to practical classification and decision making tasks. We performed experiments on Naive Bayes networks from a variety of sources: UCI Machine Learning Repository [BL13], BFC (`http://www.berkeleyfreeclinic.org/`) and CRESST (`http://www.cse.ucla.edu/`). Each network has an associated dataset containing labelled examples, which we used in some of the experiments.

6.2.1 Trading Off Budget Expenditure with Classification Accuracy

Consider the problem of classification with Naive Bayesian networks. Let D be the class variable and let $\mathbf{H} = H_1, \ldots, H_n$ be the features. In a classical setting, one classifies by observing all the available features. That is, one computes $Pr(D \mid$

$H_1, \ldots, H_n$) and then chooses a class depending on a given threshold T. If $Pr(d \mid h_1, \ldots, h_n) \geq T$, one classifies the example $h_1, \ldots, h_n$ positively; otherwise, one classifies it negatively.

Suppose now that we wish to classify based on only a subset of the features, due to feature cost. Our approach involves using `MaxDR` to find the smallest subset $\mathbf{G}$ of features where the expected SDP of observing $\mathbf{G}$ crosses a certain level L. In other words, we want the subset $\mathbf{G}$ that satisfies $\arg\min_{\mathbf{G}} |\mathbf{G}|$ and $\mathcal{E}(D, \mathbf{G}, \mathbf{H}, \mathbf{E}, T) \geq L$. Note here that we are not maximizing the expected SDP, but only ensuring that it passes a given threshold L. We do this so we can tradeoff the number of selected features with the classification accuracy. The larger L is, the more features `MaxDR` will select, and the better the classification accuracy. The smaller L is, the fewer features that `MaxDR` will select, but at the expense of less classification accuracy. As we shall see next, one can choose L so as to obtain a very interesting balance: a relatively small number of features can be selected while ensuring a very small penalty in classification accuracy.

We experimented with thresholds L ranging from $[0.70, 0.75, 0.80, 0.85, 0.90, 0.95]$. For each L, and each network, `MaxDR` found the smallest subset $\mathbf{G}$ that satisfies the above conditions. We then computed the classification accuracy based on observing only $\mathbf{G}$ versus observing all features $\mathbf{H}$. We classified each example in each dataset based on $\mathbf{G}$ and $\mathbf{H}$, and then computed for each the proportion of correctly labeled instances to total number of instances.

Table 6.3 displays the percentage of features ($\frac{|\mathbf{G}|}{|\mathbf{H}|}$) that `MaxDR` selected for different levels L of decision robustness, across all networks. We also list the average percentage of selected features and the average difference of classification accuracy (accuracy of $\mathbf{H}$ - accuracy of $\mathbf{G}$) for each threshold L. The table reveals an interesting tradeoff between the number of features selected and the classification accuracy, which can be controlled by varying the decision robustness threshold L. For example, for $L = 0.95$, we select about 60% of the features on average, while

Network	NM-DT (s)	NM-IG (s)	MaxDR (s)
bupa	0.028	0.021	0.032
pima	0.137	0.124	0.044
ident	0.129	0.119	0.147
anatomy	0.431	0.419	0.424
heart	0.704	0.682	3.230
voting	28.146	22.734	4.612
hepatitis	284.873	276.143	147.887
nav	342.767	328.874	121.715

Figure 6.4: Running time.

incurring only a 0.64% reduction in classification accuracy. This clearly shows that many features are redundant given other features. MaxDR is particularly useful in this scenario as it is able to identify those redundant features and excludes them from the selected features.

6.2.2 Decision Making

We now discuss another application of MaxDR to decision making. Given a Naive Bayes network with decision variable D, features $\mathbf{H} = H_1, \ldots, H_n$, and budget B, we wish to make a decision that is within budget, and that is maximally robust against additional observations. For example, if we can only observe three features, H_i, H_j and H_k, we wish to select them such that when observed, there is a low probability of reaching a different decision if features $\mathbf{H} \setminus \{H_i, H_j, H_k\}$ were later observed. The goal here is to *minimize our liability* against what we chose not to observe. Contrary to the previous application, we have no data here. Therefore, the notion of classification accuracy is not meaningful in this setting, and the quality of a decision is measured solely based on its robustness.

For each Naive Bayes network, budget B and decision threshold T, we used `MaxDR` to select features $\mathbf{G}$ and considered the expected SDP of decisions based on observing $\mathbf{G}$ (`MaxDR` computes the expected SDP as a side effect). For each network, the budget was set to 1/3 the number of features, with decision thresholds in $[0.1, 0.2, \ldots, 0.8, 0.9]$.[2]

Figure 6.5 provides further comparisons between `MaxDR` and the two baselines: non-myopic information gain (`NM-IG`) and non-myopic, generalized classification loss (`NM-DT`).[3] We select features $\mathbf{G}$, within budget, using each criteria and then measure the robustness of decisions based on observing the selected features (by computing the expected SDP). Our results show a number of patterns. First, the proposed criteria selects features leading to the most robust decisions. Moreover, the difference between `MaxDR` and the other criteria becomes more pronounced for extreme thresholds and for large networks. These results show that `MaxDR`, as a feature selection algorithm, is very suitable for applications in which the decision maker wishes to minimize their liability against information that they chose not to observe. That is, we select features $\mathbf{G}$ using each of these feature selection criteria and then measure the robustness of decisions based on observing the selected features. Note that we also experimented with the minimum Redundancy Maximum Relevance (mRMR) feature selection criterion [DP03]. We found the criterion to be trivial for Naive Bayes structures as it always prefers sets of features over their supersets.

[2]We selected a budget of 1/3 the total number of features because we found that the results would be trivial if the budget was set too high or too low, as all algorithms would then behave similarly.

[3]Suppose we are making decisions based on the following utilities of positive and negative decisions: $U_p = Pr(d|\mathbf{e})$ and $U_n = Pr(\overline{d}|\mathbf{e})T/(1-T)$. This corresponds to a threshold-based decision for arbitrary T. `NM-DT` selects features using the reward function $\max(U_p, U_n)$. When $T = 1/2$, we get classification loss.

6.2.3 Running time.

Assuming n binary features, a budget of m, and a cost of 1 per feature, `NM-IG` and `NM-DT` can be implemented in $O(\binom{n}{m}2^m)$ time. In particular, one only needs to consider candidate features $\mathbf{G}$ of size m due to the monotonicity of corresponding objective functions (i.e., $\mathbf{G}$ will dominate all its strict subsets). For each candidate $\mathbf{G}$, one needs to compute an expectation over $\mathbf{G}$, which takes $O(2^m)$ time. In fact, we know of no algorithm that does better than $\Theta(\binom{n}{m}2^m)$ for these criteria. `MaxDR` has a worst-case time complexity of $O(2^{n+m})$. In this case, we need to consider all 2^m candidates $\mathbf{G}$ with size $\leq m$ since the SDP is not strictly monotonic. We also need $O(2^n)$ time to compute the expected SDP by enumerating all feature states. This is much worse than the running time for `NM-IG` and `NM-DT`. However, Figure 6.4 shows that `MaxDR` does significantly better than this on average and also does better than both `NM-IG` and `NM-DT`, especially for larger networks. This is due to the knapsack and branch-and-bound pruning techniques employed by `MaxDR`.

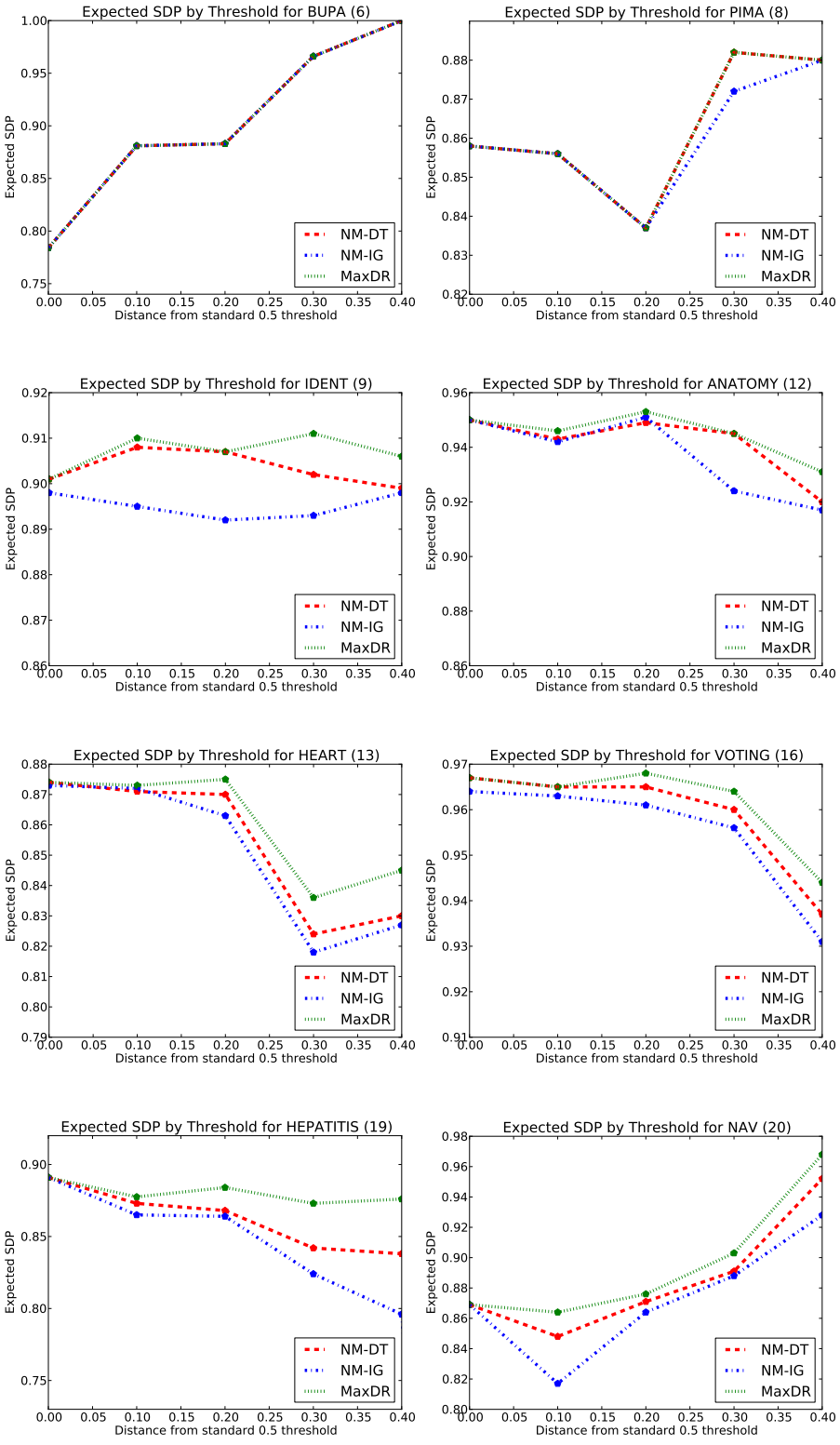


Figure 6.5: Additional experiments comparing `MaxDR` and the two baselines: non-myopic information gain (`NM-IG`) and generalized classification loss (`NM-DT`).

Algorithm 3: FindCands Finding candidate subsets to compute E-SDP over. A subset $[T, F, F, T]$ represents a subset that includes features H_1 and H_4. For a candidate set to be valid, the total profit must $\geq V$ while the total cost $\leq B$.

input:

H: $\{H_1, \ldots, H_n\}$, by decreasing efficiency

V: target threshold (see Equations 6.1 & 6.2)

B: budget

output: All subsets of features where total profit exceeds V

main:

$p \leftarrow 0.0$ *initial profit*

$\mathbf{q} \leftarrow [F, \ldots, F]_n$ *initially, no features selected*

$d \leftarrow 0$ *initial depth*

$b \leftarrow B$ *initial budget*

candidates $\leftarrow$ [ ] *list of candidate feature sets*

`DFS_KS`($\mathbf{q}$, p, d, b)

return *candidates*

1: **procedure** `DFS_KS`($\mathbf{q}$, p, d, b)
2: **if** $p + UBP(\mathbf{H}, d, b) < V$ *no profit potential*
3: **return**
4: **else if** $b < 0$ *budget exceeded*
5: **return**
6: **else if** $p \geq V$ *enough profit potential*
7: add $\mathbf{q}$ to *candidates*
8: `DFS_KS`($\mathbf{q}$, p, $d + 1$, b)
9: set feature $H_d = T$ in $\mathbf{q}$ *include* H_d
10: `DFS_KS`($\mathbf{q}$, $p + \mathsf{profit}(H_d)$, $d + 1$, $b - \mathsf{cost}(H_d)$)

Network	bupa	pima	ident	anatomy	heart	voting	hepatitis	nav		
# features	6	8	9	12	13	16	19	20		
Threshold	% features selected								avg % selected	avg diff C.A.
0.75	33.3	12.5	11.1	8.3	15.4	6.2	5.3	30.0	15.2	0.0518
0.80	66.7	25.0	11.1	16.6	23.1	6.2	5.3	30.0	23.0	0.0191
0.85	66.7	37.5	22.2	16.6	30.8	12.5	15.8	30.0	29.0	0.0162
0.90	100	50.0	33.3	25.0	38.5	12.5	26.3	45.0	41.3	0.0109
0.95	100	87.5	55.5	41.6	61.5	18.7	47.4	60.0	59.1	0.0064
0.98	100	100	88.8	66.6	76.9	31.2	68.4	75.0	75.9	0.0048
0.99	100	100	88.8	83.3	92.3	43.7	89.5	80.0	84.7	0.0042
1.00	100	100	100	100	100	100	100	100	100	0.0000

Table 6.3: This table shows 1) the percentage of features selected by `MaxDR` for different levels of decision robustness 2) the average percentage of features selected and 3) the average difference of classification accuracy.

CHAPTER 7

The Complexity of Computing the SDP

In this chapter, we first discuss the relationship between different complexity classes as well as review previous complexity results on the SDP. Subsequently, we present new complexity results for the SDP. The material in this chapter is based on the work published in [CCD13, CCD14].

7.1 Introduction

First, observe the following relationship between complexity classes:

$$\text{NP} \subseteq \text{PP} \subseteq \text{NP}^{\text{PP}} \subseteq \text{PP}^{\text{PP}}$$

Note that MPE is the prototypical NP-complete problem for probabilistic inference [Shi94]. MAP [PD04] is the prototypical NP^{PP}-complete problem — MPE involves finding the most likely instantiation over all variables, whereas computing MAP finds the most likely instantiation of some subset of variables. SDP was shown to be PP^{PP}-complete [CXD12], where PP^{PP} can be thought of as a counting variant of the class NP^{PP}, hence highly intractable.

We next present new complexity results for the SDP. We first prove that the complexity of computing the SDP in Naive Bayes structures is NP-hard. We then show that the general complexity of computing the SDP (PP^{PP}-complete) lies in the same complexity class as a general expectation computation problem that is applicable to a wide variety of queries in graphical models, such as the computation of non-myopic value of information.

7.2 Computing the SDP in Naive Bayes

SDP is known to be PP^{PP}-complete [CXD12]. We prove that computing the SDP remains hard even for Naive Bayes networks. Naive Bayes networks are typically tractable for basic queries, yet remain NP-hard for MAP [De 11], and PP-complete for computing the value of information [KG09].

Theorem 1. *Computing the Same-Decision Probability in a Naive Bayes network is* NP*-hard.*

The proof can be found in Appendix A.

7.3 The Complexity of Computing the Non-myopic VOI

The SDP was shown to be a PP^{PP}-complete problem by [CXD12]. The class PP^{PP} is essentially the counting variant of the NP^{PP} class, which contains the polynomial hierarchy PH and for which the MAP problem is complete [PD04]. We show in this section that a general problem of computing expectations is also PP^{PP}-complete, with the non-myopic VOI and SDP being an instance of such an expectation. Thus, the development of algorithms to compute the SDP will be beneficial to problems in the PP^{PP} class, which in turn benefits computing an assortment of expectations, including non-myopic VOI.

The proposed expectation computation is based on using a reward function R with some properties that we review next. In particular, the function R is assumed to map a probability distribution $Pr(D \mid \mathbf{e})$ to a numeric value. We also assume that the minimum l and maximum u of this range are polytime computable. These assumptions are not too limiting—for example, both entropy and utility can be expressed using reward functions that fall in this category [KG09].

We now consider the following computation of expectations.

D-EPT: Given a polynomial-time computable reward function R, hypothesis variable D, unobserved variables $\mathbf{H}$, evidence $\mathbf{e}$, a real number N, and a distribution Pr induced by a Bayesian network over variables $\mathbf{X}$,[1] the **expectation decision problem** asks: Is

$$\mathbb{E} = \sum_{\mathbf{h}} R(Pr(D \mid \mathbf{h}, \mathbf{e}))Pr(\mathbf{h} \mid \mathbf{e})$$

greater than N?

Note that the SDP falls as a special case when the reward function R is the SDP indicator function (see Definition 2). For example, in the definition used by [CXD12], the decision function outputs one of two decisions depending on whether $Pr(d \mid \mathbf{e}) > T$ for some value d of D and some threshold T.

We now have the following theorems, with proofs in Appendix A.

Theorem 2. ***D-EPT*** *is* $\mathrm{PP}^{\mathrm{PP}}$*-hard.*

Theorem 3. ***D-EPT*** *is in* $\mathrm{PP}^{\mathrm{PP}}$*.*

This shows that **D-EPT** is $\mathrm{PP}^{\mathrm{PP}}$-complete and implies that computational problems such as computing the non-myopic VOI using a variety of reward functions is also $\mathrm{PP}^{\mathrm{PP}}$-complete.

[1] This proof also holds for influence diagrams constrained to have only one decision node.

CHAPTER 8

SDP in Computer Adaptive Testing

Computer Adaptive Tests dynamically allocate questions to students based on their previous responses. This involves several challenges, such as determining when the test should terminate, as well as which questions should be asked. In this chapter, we apply the SDP as an information gathering criterion in the context of Computer Adaptive Testing. In particular, we show that it can act as a stopping criterion for determining if further questions are needed. If further questions are needed, we also show that the SDP can also serve as a selection criterion for determining which questions should be asked. We shown empirically that the SDP is a viable and intuitive approach for Bayesian-based Computer Adaptive Tests, as its usage allows us to ask fewer questions while still maintaining the same level of precision and recall in terms of classifying competent students.

8.1 Introduction to Computer Adaptive Testing

Computer Adaptive Tests have recently become increasingly popular, as they have the ability to adapt to each individual unique user [Vom04, ADM07]. This in turn allows for the test to tailor itself specifically based on user responses and its current estimation of the user's knowledge level. For example, if the user answers a series of questions correctly, the test can adjust and curate some more difficult questions. On the other hand, if the user answers a series of questions incorrectly, the test can adjust and present easier questions.

Bayesian networks have been used as the principal base model for many Computer Adaptive Tests [MP02, Vom04, ADM07, MS08], as they offer powerful approaches to inferring a user's knowledge level given uncertain information (unanswered questions). Additionally, a significant body of work has been done on using Bayesian networks to perform a wide array of tasks in the field of educational diagnosis [VN01, CGV02, SH06].

A key question in Computer Adaptive Tests is when the test should terminate. Some students may perform so well or so poorly that the system can recognize that further testing is unnecessary, as asking further questions would have very little probability of reversing the initial diagnosis. In [MP02], there is further discussion on stopping criteria that are used to determine when further questioning is needed, as well as selection criteria that are used to determine which questions are actually asked.

In this chapter, we take the SDP and show its usefulness when applied as both a stopping criterion and selection criterion for question selection in a Computer Adaptive Test in contrast to the standard stopping and selection criteria. We first present some motivation for the constructed Computer Adaptive Test. We then discuss some related work in the educational diagnosis field. Following that, we then show how the SDP can be used as a stopping and selection criterion for our Computer Adaptive Test. Finally, we discuss experiment setup, present empirical results demonstrating the usefulness of the SDP, and then conclude the chapter.

8.2 Motivation Behind Computer Adaptive Testing

The Berkeley Free Clinic[1] is a clinic staffed entirely by volunteers and offers medical and dental treatment for the surrounding community. In particular, the dental section of the clinic offers a wide variety of services ranging from cleanings, x–ray

[1] `http://www.berkeleyfreeclinic.org/`

services, fillings, and extractions. Due to the expertise required for these services, the volunteers need to be highly trained to apply their knowledge in a practical setting. The volunteers are thus required to undergo a training period in order to prepare them to serve at the clinic.

Recently, it was decided that in order to better evaluate the knowledge of the volunteers, the volunteers would go through a *competency exam* to determine whether or not they had sufficient knowledge. The idea of this test is that if a volunteer was found to be inadequate, that he/she would have to undergo further instruction. The coordinators of the clinic worked in conjunction with the volunteer dentists in order to create a written test that thoroughly tested the different concepts and skills necessary for a volunteer. The written test consists of 63 questions that includes questions ranging from showing a portrait of a certain instrument (e.g. a perioprobe) and asking "Name this instrument" to "What is in a bridge and crown set up tray?". Portions of this test can be seen in Figures 8.1 and 8.2.

This test was given to 22 subjects. 16 of the subjects were evaluated prior to the test–taking to be competent volunteers, whereas the other 6 subjects were evaluated to be non–competent volunteers. The clinic coordinators set the pass threshold to be at 60%, meaning that volunteers needed to correctly answer 60% of the questions in order to be considered competent. The test proved to be effective in that of the 16 competent volunteers, 15 of them passed, and of the 6 non–competent volunteers, none of them passed. The test taking duration ranged from 30 minutes to 1 hour. Feedback from the participants indicated that they felt this test was fair and covered the necessary bases to be a volunteer, but that the test was too time–consuming.

The complaints about the duration of the test is in line with the discoveries of [GAS07], who discover that a big problem with web based tests is that they are too long, which may hinder students' ability to perform as they simply become

bored and careless. This serves as the chief motivation for our work, as we want to turn this test into a Computer Adaptive Test (CAT), so that we can present students a test with fewer questions, but just as many *relevant* questions — this way we can decrease overall test duration without compromising the integrity of the test.

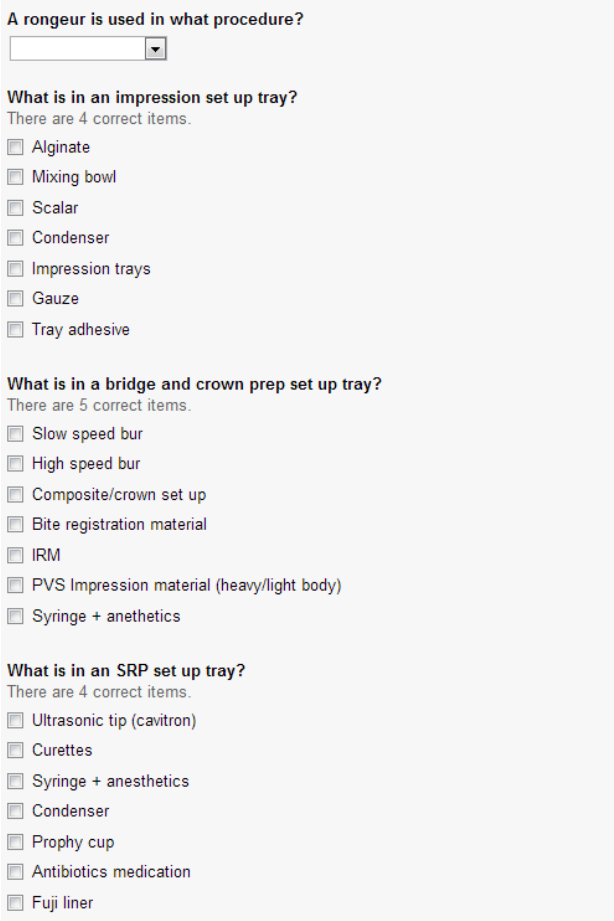


Figure 8.1: A portion of the competency test that measures general dental clinic knowledge.

8.3 Related Work

There has been a surge of interest in utilizing various recent artificial intelligence techniques to increase the quality of educational diagnosis, a broad field that covers

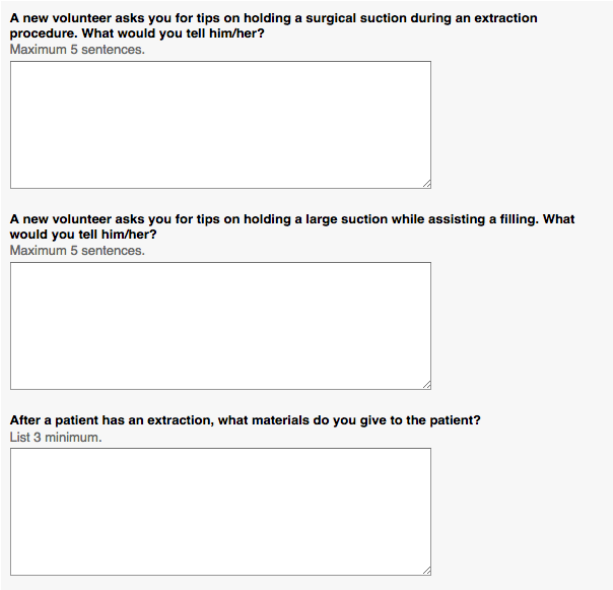


Figure 8.2: A portion of the competency test that measures procedural knowledge.

Interactive Tutoring Systems (ITS) [CGV02, SH06, GS12], as well as Computer Adaptive Tests (CAT) [MS08, MDC13]. These applications have overall found to be to be a significant improvement over traditional techniques [MP02, Vom04, Sin06, BM07, BAC10, MDC13].

In the work of [MP02], they use Bayesian networks as the framework for constructing Computer Adaptive Tests (CATs). They introduce a model for representing student knowledge, where knowledge is modeled as different interrelated concepts. They noted that student knowledge can be diagnosed (inferred) by treating student answers as evidence. In addition to this model, they introduced an adaptive testing algorithm. To evaluate their model, they used 180 simulated students. They found that the introduction of adaptive question selection method improves behavior for both in terms of accuracy and efficiency, as using an adaptive algorithm resulted in fewer questions and higher diagnosis accuracy. Their model incorporates notions such as "slipping", where a student might answer a question incorrectly even if the concept is known. They generate students randomly by sampling from the network, and are assigned to have different knowledge

of various concepts.

Similar results were shown in [Vom04], who discuss applications of Bayesian networks in educational diagnosis, especially in skill diagnosis. They show that modeling dependence between various skills allows for higher quality of diagnosis. Additionally, they show that Computer Adaptive Tests allow them to best select a fitting question to test for competency in certain skills. They found that computer adaptive testing substantially reduces the number of questions that may need to be asked.

Just like [MP02] and [Vom04], [ADM07] found that Bayesian networks to be useful to model the links between various related proficiencies, and modeled unseen questions as unobserved variables. They use diagnostic Bayesian networks in order to model uncertainty, where experts build a Bayesian network and calibrate it from data.

Another application of Bayesian networks to model student knowledge is discussed in [MS08]. They use a Bayesian network to model qualifying exams for graduate students at Stanford, where their goal is to select the questions (observations) that can best measure the knowledge in order to determine whether or not the student should pass. They also use a threshold function and stress that reducing the number of questions asked is important. They also prove the NP-hardness of deciding an optimal set of questions to assign a student.

Additionally, a variety of work has been done on using Bayesian networks in the educational diagnosis field for threshold-based decision making under uncertainty, such as 1) [Xen04], where the authors model the educational experience of various students and use threshold-based decisions to determine whether or not a student is likely to fail, 2) [AW05], where the goal is to determine the type of learner a student was (in terms of attitude) 3) [GCV98], where the goal is to infer what part of a problem a student is having trouble with, in order to provide hint-based remediation, 4) [BHM04], where the constructed ITS can assess knowledge using a

BN as well as recommending certain remediation, and a threshold-based decision is used to determine if a student is competent in an area or not.

Some other work in the field of educational diagnosis can be found in [BM07], which describes a model that details the relationship between the knowledge of certain concepts and how the student will perform on certain questions. They note that if a user demonstrates lack of knowledge, the model can be used to locate the most likely concepts that will remedy the situation. They found that using a Bayesian model allows them to compute the probability of a student answering a question correctly given competency in some knowledge. Also of interest is the work of [WRM09], who show that it is possible to model the progression of student learning from novice to expert.

More recently, the work on educational diagnosis modeling has focused on even more intricate modeling and remediation methods. For instance, [RIM13] has focused on modeling user emotional states, in order to detect when users are frustrated as well as the cause of the frustration. By finding the root cause of the frustration, special measures may be taken to remediate. Similarly, [CV13] has done significant work in modeling a student's performance, progress, and behavior. The developed e-learning system has been deployed into production. In the work of [CV14], it is noted that modeling user's detailed characteristics, such as knowledge, errors, and motivation can improve quality of a student's learning process as it allows for better remediation. The need for personalized remediation is also further stressed as [VC14] studies the importance of giving learners instruction and support directly. They stress that learner models need to be adjusted and updated with new information about the learners knowledge, affective states, and behavior in order to be maximally effective.

8.4 Computer Adaptive Testing

8.4.1 Using a Bayesian Network

To address the problem of having such a time–consuming test, we believe that using a Computerized Adaptive Test (CAT) could differentiate the competent students from the non–competent students with **fewer** questions than a standard test. Asking fewer questions while still accurately measuring a student's ability is an oft–studied problem, and is discussed thoroughly in [MP02, GAS07, MS08, MDC13]. Our main goal is to best measure whether or not a student is competent with a *limited* subset of questions. This means that we have to determine when enough questions have been asked, as well as is to be more *which* questions are asked, so as to ensure that overall, the Computer Adaptive Test is comprised of fewer questions.

Bayesian networks have been found to be especially useful for decision making under uncertainty [Dar09]. We believe that they provide us a natural mechanism to model both 1) student knowledge and 2) questions that may be potentially asked to measure student knowledge. In these models, Bayesian networks can model the relationships between various proficiencies. These models detail the relationship between knowledge of certain concepts and how the student will perform on certain questions.

Using a Bayesian network thus allows us to predict how likely a student is to have knowledge in a certain concept based on the student's answers. Hence, we worked with the clinic coordinators and experts to create a Bayesian network structure and elicit the parameters for this exam. Our developed model is similar to the models developed by [MP02, Vom04, ADM07, MS08, BM07, MDC13], where Bayesian networks are used to evaluate and in a sense, "diagnose" a student's degree of knowledge, or competency, in various fields of knowledge.

In our network, we have a main variable of interest that is representative of

the student's total level of knowledge. We refer to this variable as the *decision variable* (D), and it serves as an overall measure of our belief that a student is competent. The decision variable is surrounded by a variety of *concept* variables that are latent. The final type of variable is a *question* variable that represents a question that may be asked to the student. An answered question, whether correct or incorrect, will act as *evidence* that can influence our belief on the student's competency. The graphical structure of the model can be seen in Figure 8.3.

Similarly to [GCV98, VN01, CWM14], the clinic coordinators/experts determined that the "pass threshold" should be set at 0.8, meaning that if given some evidence (questions answered), the *posterior* probability of the decision variable was found to be over 0.8, then the student would be considered as competent.[2] According to this pass threshold, we found once again that of the 16 competent volunteers, 15 of them passed, and of the 6 non–competent volunteers, none of them passed. The volunteers' scores are shown in Table 8.1. From this table we can see that our constructed Bayesian model allows us to accurately predict a student's competency.

8.4.2 Stopping Criterion for Information Gathering

Although the test has a total of 63 questions, there may be a point during the adaptive test where we can determine that it is unnecessary to ask any further questions. For example, if a volunteer answers 40 questions correctly in a row, we can then be very certain that the student is competent and can thus not bother to ask the other 23 questions — therefore cutting the total required test time.

On the contrary, say a volunteer answers 40 consecutive questions *incorrectly*. Clearly, in this case we can be certain that the student is incompetent and we can again terminate the test early. Even though these scenarios are unlikely, it is

[2]Another commonly used threshold in the educational diagnosis field is 0.7, as is used by [BHM04, AW05].

clear that at times early test termination is intuitive.

For educational diagnosis in Bayesian networks, [MP02, MDC13] shows there are two standard ways to determine if more information gathering (questions posed to the student) is necessary. The first involves a fixed test that asks a set number of questions, and then determines if the student is competent after all the questions have been answered. This method is clearly ineffective, as it does not take into account at all the students' answers.

The second way involves terminating the test once the posterior probability of the decision variable is above or below some threshold. This stopping criteria is also seen in [HCK92, HBR95, KMR03, LP06]. Note that [MP02] compares an approach that utilizes a set number of questions with a threshold-based approach — they found that with a set number of questions they were able to diagnose students correctly 90.27 percent of the time, whereas by using an adaptive criterion they were able to diagnose students correctly 94.54 percent of the time, while requiring fewer number of questions to be asked. This is a clear indication that using a less trivial stopping criterion can be ultimately beneficial.

Another possibility of a stopping criterion involves computing the value of information of some observations, and if that value is not high enough, to then stop information gathering. However, this involves either computing the *myopic* value of information and just computing the usefulness of making one observation, or computing the *non-myopic* value of information and computing the usefulness of making several observations. The former is fairly easy to compute, but can prove to be very limited as often the combined usefulness of some observations is greater than its parts. Computing the *non-myopic* value of information is useful in determining if a set of observations has significant value, but is intractable to compute if the set of observations is large [KG09].

To decide whether or not enough information has been gathered, we use a non-myopic stopping criterion and compute the *Same–Decision Probability* (SDP)

[DC10, CCD14]. In short, the SDP is a measure of how robust a decision is with respect to some unobserved variables and will help us determine how much more information needs to be gathered.

In this case, based on the student's responses, at any point in time, we can compute the posterior probability that they are competent and thus determine a decision of whether or not they are competent. Keep in mind that this decision is temporary and can be reversed if more questions are asked — we can compute the SDP over the remaining unasked questions to determine how likely it is that the current decision (deciding whether a student is competent or non–competent) would remain the same even if the remaining questions were asked. If the SDP is high, that is an indication that we can terminate the test early. We found that using the SDP allowed us to cut the test duration significantly while maintaining diagnosis accuracy. Details of our experiment setup and experimental results can be found in Section 8.5.1.

8.4.3 Selection Criterion for Information Gathering

We have now established the usefulness of the SDP as a stopping criterion for Computer Adaptive Tests, as it tells us when we can terminate the test. However, the question remains: if computing the SDP indicates that more questions are necessary, *which* questions should we ask?

In a standard, non–adaptive test, the ordering of the questions cannot be controlled. In the adaptive setting we have more control — based on a test taker's answers on the test so far, we can select questions that have the most potential to give us further insight on the test taker. Our goal here is to select questions such that the *expected* SDP after observing those questions is maximal. In other words, we want to ask questions such that no matter how are answered, will minimize the usefulness of any *remaining* questions. By maximizing this objective, we reduce

the overall number of questions that need to be asked before test termination.

As stated before, the use of computing the *value of information* (VOI) is essential to the process of information gathering. Since for our model it is intractable to compute the non–myopic value of information, that leaves only computing the myopic VOI as an available possibility. The VOI of observing a variable may depend on various objective functions, for instance, how much the observation reduces the entropy of the decision variable. There is a comprehensive overview of these different objective functions in [KG09].[3] In Chapter 6, we compared the approach of maximizing these standard objective functions to maximizing the expected SDP and found that maximizing the expected SDP is more effective, particularly in cases when the decision threshold is extreme.

Our approach involves selecting the question that leads to the highest expected SDP. We found that selecting variables to maximize the expected SDP allowed us to substantially decrease the number of questions selected compared to other selection criteria. Details of experiment setup and experimental results can be found in Section 8.5.2.

8.5 Experiments

In this section, we discuss the experiment setup and experimental results on the usage of the SDP as a stopping criterion and selection criterion for our adaptive test. We compare the SDP against the standard criteria used by other Bayesian Computer Adaptive Tests. First, we introduce some notation used throughout the experimental section.

We use standard notation for variables and their instantiations, where variables

[3]Additionally, Additionally, [GK11b] studies the usage of adaptive submodularity for selection criteria and shows that objective functions that satisfy this notion can be readily approximated. Unfortunately, for our problem adaptive submodularity not applicable here because there are no proxies to the objective function (such as information gain) that are adaptive submodular.

are denoted by upper case letters (e.g. X) and their instantiations by lower case letters (e.g. x). Sets of variables are then denoted by bold upper case letters (e.g. $\mathbf{X}$) and their instantiations by bold lower case letters (e.g. $\mathbf{x}$). The primary decision variable, which measures a student's overall competency, is denoted by D with states $+$ and $-$ to respectively mean that the student is competent/non-competent.

Note that the complete test has $n = 63$ questions — the data we have consists of the 22 fully completed tests

$$\mathbf{T} = \mathbf{e}_1, \ldots, \mathbf{e}_{22},$$

where each $\mathbf{e}_i \in \mathbf{T}$ is equivalent to an instantiation of n answers for a student S_i.

$$Pr(D = + \mid \mathbf{e})$$

thus denotes the posterior probability that student S_i is competent given their test responses. Note that for each student, the quantity $Pr(D = + \mid \mathbf{e})$ can be found in Table 8.1. To determine whether or not the student passes, we would check to see if $Pr(D = + \mid \mathbf{e}) \geq 0.80$.

To simulate partially completed tests, we randomly sampled answers without replacement from each $\mathbf{e}_i \in \mathbf{T}$ to generate a collection $\mathbf{Q}_i$ of partially completed tests $\mathbf{q}_{i,1}, \ldots, \mathbf{q}_{i,n}$. Each $\mathbf{q}_{i,j} \in \mathbf{Q}_i$ consists of an instantiation of answers where:

- $1 \leq i \leq 22$
- $10 \leq j \leq n$
- $\mathbf{q}_{i,j} \subset \mathbf{q}_{i,j'}$ for $j < j'$
- $\mathbf{q}_{i,n} = \mathbf{e}_i$

Note that the minimum number of answers for any partial completed test is 10 answers, as the coordinators believed that that at the very least, even a Computer

Adaptive Test should have 10 questions. The partial completed test $\mathbf{q}_{4,10}$ can then be thought of as a test for student S_4 with 10 randomly selected answers, whereas $\mathbf{q}_{4,11}$ would be the same test, but with an additional selected answer. This allows us to perform experiments where we can check $\mathbf{q}_{4,10}$, determine if we want to stop or not, and then simulate the student's response to the next question with $\mathbf{q}_{4,11}$. Note that we since we are sampling answers randomly, it is possible for us to sample a $\mathbf{Q}_i$ and $\mathbf{Q}'_i$ that are not equivalent.

8.5.1 Stopping Criterion Experiments

We simulate partially completed tests and compare how well the SDP does as a stopping criteria versus more traditional methods. Therefore, for each test $\mathbf{Q}_i$, for students from S_1 to S_{22}, we check all possible partial test completions. From $\mathbf{q}_{i,1}$ to $\mathbf{q}_{i,n-1}$, we determine whether or not it is valid to stop. For $\mathbf{q}_{i,j}, \forall j$, we refer to the remaining $n-j$ unanswered questions as $\mathbf{H}$.

For stopping criteria, the standard methods would involve stopping after a number of questions are asked (so stopping after $\mathbf{q}_{i,j}$ for some arbitrary j), or stopping once the posterior probability that student S_i is competent, $Pr(D = + \mid \mathbf{q}_{i,j})$ is above or below some threshold T.

The results of using a set number of question stopping criteria are shown in Table 8.2. The results of using a posterior probability threshold stopping criteria [MP02, Xen04, AW05, MS08] are shown in Table 8.3. In both of these tables we can see that there is a clear trade off between the accuracy of our model to the number of questions that is asked.

Using SDP as a stopping criteria would involve computing the SDP over some subset of $\mathbf{H}$ to see how likely a decision is to be stable if those variables were to be observed. We can use the SDP as a stopping criteria as well. We stop based on a threshold T: if the SDP $\geq T$, we can stop information gathering. Note that

if $T = 1.0$, then that means there is absolutely zero probability that any further observation will change the decision. The results of using SDP as a stopping criteria are shown in Table 8.4.

We can see that even when this threshold is chosen to be fairly small (0.85), we can still attain precision and recall that are comparative to the best results from using either of the other two stopping criterions. In fact, for the standard posterior probability threshold stopping criterion, whose results are shown in Table 8.3, we can see that although using that criterion allows us to attain a high precision, the recall performance is fairly poor.

Notice that if the threshold is chosen to be very large (0.999), the precision and recall will be the equivalent to the case where $n - 1$ (62) questions are asked (for the other stopping criteria of asking a set number of questions). However, in this case, the average number of questions that needs to be asked is 42.05 as opposed to 62, meaning that SDP as a stopping criterion has the same robustness as the "set number of questions" stopping criterion while asking nearly 20 less questions.

It is clear from our results that if our goal is to ask fewer questions while maintaining competitive precision and recall rates, using the SDP as a stopping criterion proves more useful than the traditional stopping criteria of 1) asking a set number of questions and 2) checking to see if the posterior probability of competency surpasses a threshold — using the SDP as a stopping criterion allows us to reduce the number of questions asked while still maintaining the same precision and recall.

8.5.2 Selection Criterion Experiments

We implemented experiments to compare various selection criteria such as 1) random selection, where we pick the next question randomly (as is done in non-

adaptive tests), 2) information gain [MP02, Vom04], and 3) margins of confidence [KG09]. Note that as this model does not fit the decision theoretic setting (there are no assigned utilities), we do not use the maximization of utility as a selection criteria. In addition to these standard selection criteria, we compare the SDP as a selection criterion.

Since our adaptive test involves selecting one question at a time, our goal is to select a variable $H \in \mathbf{H}$ that leads to the greatest gain in the SDP (SDP gain). The problem with solely using this approach is that occasionally the SDP gain of observing a variable is 0 (which indicates the need for multiple questions to be asked in order for us to change our decision to be changed). This means that solely using the SDP gain of one variable as a selection criteria will often lead to many ties between variables. We remedy this by using a hybrid approach where the first priority selection criteria is the SDP gain and the second priority is information gain. For this hybrid approach, in order to search for the variable that leads to the highest SDP gain, we used the algorithm described in Chapter 4.

We compare how quickly these selection criteria allow us to stop information gathering in Table 8.5. We find that for our simulations, on average, the random selection criterion would ask an additional **8.18** questions, whereas using information gain would ask an additional **3.67** questions, using margins of confidence would ask an additional **3.57** questions, and using the hybrid approach would necessitate the fewest additional questions: **2.77** questions.

Note that while the random question selection criterion clearly has the poorest performance, there is only a small difference between using the hybrid approach and using standard information gain [MP02, Vom04] or margins of confidence [KG09]. However, the benefit of using the hybrid approach still can be clearly seen, thus showing how computing the SDP gain is useful as a selection criteria. We believe that the results of considering the SDP gain of *asking multiple questions* at a time would drastically improve the usefulness of computing the SDP gain.

8.5.3 Running Times

The SDP has been shown to be highly intractable, being PP^{PP}-complete [CXD12]. Therefore, the running times of both the SDP stopping criterion algorithm and SDP selection criterion algorithm are much slower than using the standard information gathering criteria. The runtime is heavily dependent on the number of unanswered questions that must be considered. Figure 8.4 show a plot of the runtimes of utilizing the SDP stopping criterion and SDP selection criterion in comparison to standard measures. Note that since the SDP selection criterion algorithm uses a hybrid approach, it runs faster than the SDP stopping criterion algorithm. The difficulty of computing the SDP over large sets of unobserved variables motivates the need for further algorithms to compute the SDP.

8.6 Conclusion

In this chapter, we have shown how we created an adaptive test using a Bayesian network as the underlying model. Additionally, we show how the notion of the SDP can be used to gather information in this context. In particular, we have shown empirically that the SDP is a viable and intuitive approach for determining question selection, as its usage allows us to ask fewer questions while still maintaining the same level of precision and recall for diagnosing competent students.

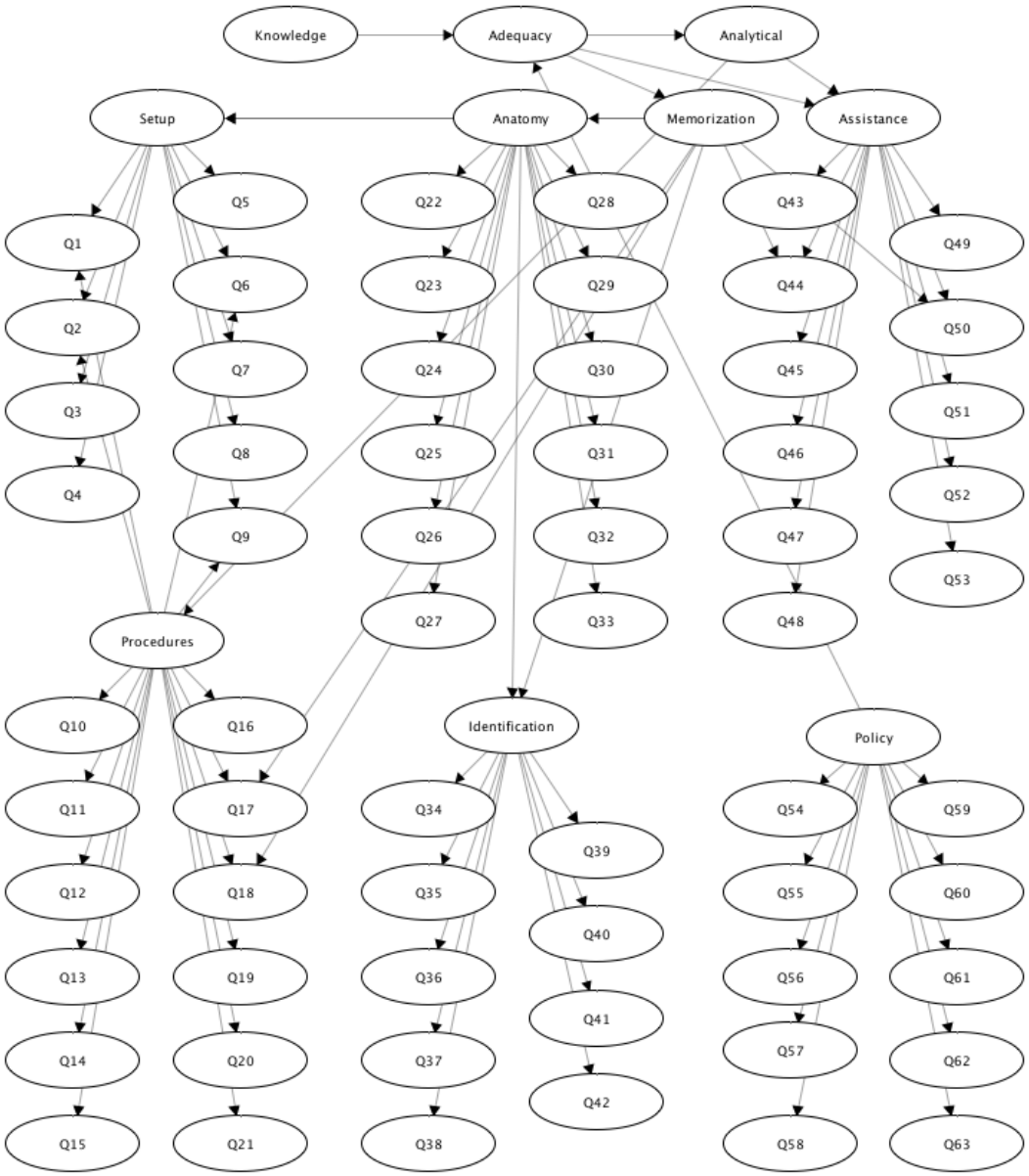


Figure 8.3: Bayesian network modeling clinic volunteer knowledge. The 63 questions are labeled from Q1 to Q63. The primary decision variable *Adequacy* is located in the top center.

Student #	Percentage Correct	Posterior Probability
1	0.952	0.957
2	0.921	0.941
3	0.714	0.951
4	0.539	0.704
5	0.762	0.956
6	0.746	0.950
7	0.825	0.957
8	0.619	0.948
9	0.777	0.952
10	0.857	0.954
11	0.857	0.957
12	0.809	0.956
13	0.349	0.153
14	0.238	0.023
15	0.539	0.178
16	0.809	0.923
17	0.603	0.883
18	0.635	0.954
19	0.873	0.954
20	0.524	0.135
21	0.492	0.153
22	0.413	0.314

Table 8.1: This table shows, for each student, the percentage of questions answered correctly (out of 63 questions), and the posterior probability of the student being competent given all the answered questions. The non–competent volunteers are bolded.

# Questions	Precision	Recall
10	0.566	0.403
15	0.752	0.659
20	0.805	0.755
25	0.815	0.812
30	0.835	0.835
35	0.874	0.875
40	0.901	0.909
45	0.915	0.914
50	0.918	0.926
55	0.924	0.9375
60	0.950	0.9375
62	0.954	0.9375

Table 8.2: Precision and recall when using a set number of questions

T	Precision	Recall	# Questions Asked
0.75	0.803	0.758	10.13
0.775	0.813	0.821	13.04
0.80	0.836	0.832	16.02
0.825	0.867	0.872	19.26
0.85	0.903	0.915	23.13
0.875	0.919	0.912	27.14
0.90	0.908	0.911	31.57
0.925	0.968	0.884	36.82
0.95	0.974	0.8175	44.25

Table 8.3: Precision, recall, and average number of questions asked for different thresholds.

SDP T	Precision	Recall	# Questions Asked
0.85	0.927	0.932	39.10
0.875	0.938	0.9375	39.73
0.90	0.942	0.9375	39.61
0.925	0.946	0.9375	39.77
0.95	0.946	0.9375	40.09
0.975	0.950	0.9375	40.61
0.99	0.950	0.9375	41.14
0.995	0.950	0.9375	41.41
0.999	0.954	0.9375	42.05

Table 8.4: Precision, recall, and average number of questions asked for different SDP thresholds.

SDP T	Random	IG	Margins	SDP
0.85	4.77	2.98	2.71	2.05
0.875	4.78	3.08	2.87	2.14
0.90	5.08	3.20	3.12	2.20
0.925	5.95	3.19	3.04	2.31
0.95	6.81	3.37	3.44	2.47
0.975	7.86	3.73	3.56	2.96
0.99	10.64	4.19	4.29	3.18
0.995	12.95	4.32	4.22	3.43
0.999	14.80	4.97	4.91	4.25

Table 8.5: Number of additional observations necessary for the random selection criterion, information gain criterion, margins of confidence criterion, and the SDP hybrid criterion.

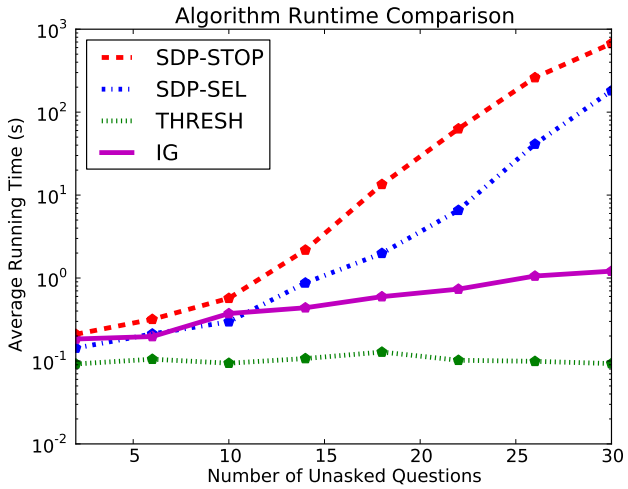


Figure 8.4: Runtimes of different algorithms. `SDP-STOP` and `SDP-SEL` are respectively the stopping and selection criterion algorithms, `THRESH` represents the standard posterior stopping criterion, and `IG` represents the standard information gain selection criterion.

CHAPTER 9

SDP in Machine Learning

In this chapter, we consider the robustness of learning Bayesian network parameters — a common challenge in Bayesian networks. In particular, we seek to answer two questions that arise when we learn such parameters from a dataset. First, do we have enough data? Second, if we do not have enough data, how much more data do we need? We show how the SDP can be be leveraged to answer these and related questions about robust learning, not only efficiently, but also in a simple yet precise way. The material in this chapter is based on the work published in [CCDew].

9.1 Introduction

For machine learning practitioners, a frequently asked question is whether one has enough data. For example, if one learns a classifier from data, how much can the classifier change, if one collected a bit more data? For another example, consider A/B testing, where we have to decide between two different versions of an online advertisement, with two different, but unknown, click-through rates. By randomly serving one of the ads to the visitors of a website, one can collect data in order to estimate each ad's click-through rate (and hence, profitability). However, there is an opportunity cost in delaying the adoption of the more profitable ad. In such cases, a practitioner has to decide whether one has seen enough data to make a confident decision. Such questions naturally arise broadly in machine learning, from decision-making under uncertainty, to active learning [TK00, TK01] and

online learning [HBB10].

These questions deal more broadly with the robustness of probabilistic models that are learned from data. In this chapter, we seek to provide answers to various questions about such robustness in the context of learning the parameters of a Bayesian network under complete data. The first question that we ask is: *Do we have enough data?* More precisely, how robust are the parameters of a Bayesian network that we learned from a given dataset? If we do not have enough data, the next question we ask is: *How much more data do we need?* That is, can we pin down a specific number of additional examples, which if we were to obtain, would grant us sufficient confidence in the robustness of our learned model?

In *practice*, these questions are often answered in less than rigorous terms. For example, one may stop collecting data when one decides that their results look "smooth enough", or by inspecting whether the learned parameters have "sufficiently" stabilized. In *theory*, there are sample complexity bounds for learning the parameters of a Bayesian network [Jor02, KF09], which state that one can learn a Bayesian network with a number of samples that is polynomial in the number of variables and parents (among other parameters). However, these are generally upper bounds, and they do not necessarily provide sharp answers to our questions in practice.

We propose to quantify the robustness of learned parameters by framing the above questions in terms of the SDP. In the original context, the *decision* corresponds to whether some probability crosses a given threshold, and *information* corresponds to the value of some unobserved variables.

We propose in this chapter a new instance of the SDP, where the *decision* corresponds to whether the distance between two learned parameters crosses a given threshold, and where *information* corresponds to additional data. We refer to this instance of the SDP as the *same-parameter probability (SPP)*. The SPP allows us to formalize the question "*Do we have enough data?*" as "What is the

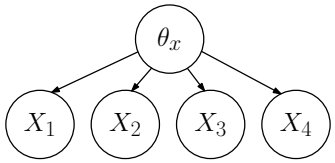


Figure 9.1: Learning from observations $X_1, \ldots, X_4$.

probability that our parameter estimates remain within a given threshold, given more data?" In order to answer the second question, "*How much more data do we need?*", we appeal to the *expected* SPP (E-SPP), which is a new instance of the expected-SDP. that we introduce. In this case, the question is formalized as "How much more data do we need, until the SPP is expected to be at least p?" As we shall illustrate in this chapter, the SPP and the expected-SPP, as probabilistic queries, can efficiently provide simple yet precise answers to questions about the robustness of parameters learned from data.

This chapter is structured as follows, we first review parameter learning in Bayesian networks, then introduce the SPP, a new query for quantifying the robustness of parameter learning. We then discuss how to compute the SPP efficiently and then show multiple examples of use cases. We then show empirical results that validate our approach. Finally, we review related work and then present concluding thoughts.

9.2 Bayesian Parameter Learning

A Bayesian network is a directed acyclic graph G with a conditional probability table (CPT) associated with each node X and its parents $\mathbf{U}$. When a variable X is binary, with a positive state x, and a negative state $\bar{x}$, the CPT of variable X consists of parameters $\theta_{x|\mathbf{u}}$, one for each parent instantiation $\mathbf{u}$. This parameter represents the probability $Pr(X{=}x|\mathbf{U}{=}\mathbf{u})$. Further, $\theta_{\bar{x}|\mathbf{u}} = 1 - \theta_{x|\mathbf{u}}$. For the rest of this chapter, we assume our variables X are binary.

We are interested in learning the parameters of a Bayesian network, from a given dataset, where a *dataset* is a multi-set of *examples*. In this chapter, we assume a dataset is *complete*, in that each example is an instantiation of all network variables. We will use $\mathcal{D}$ to denote a dataset and $\mathbf{d}_1, \ldots, \mathbf{d}_N$ to denote its N examples. The following is a dataset of three examples, over four binary variables:

example	A	B	C	D
$\mathbf{d}_1$	a	b	c	d
$\mathbf{d}_2$	a	$\bar{b}$	$\bar{c}$	d
$\mathbf{d}_3$	$\bar{a}$	$\bar{b}$	c	d

When we have complete data, the parameters $\theta_{x|\mathbf{u}}$ of a Bayesian network can be estimated independently. Moreover, estimating parameter $\theta_{x|\mathbf{u}}$ can be further simplified as follows. Instead of estimating $\theta_{x|\mathbf{u}}$ from the original dataset $\mathcal{D}$, we estimate θ_x from a simpler dataset $\mathcal{D}_{\mathbf{u}}$ obtained by projecting all examples with parent instantiation $\mathbf{u}$ on the variable X. For example, the simplified dataset $\mathcal{D}_{\mathbf{u}} = \{x, \bar{x}, x, x\}$ implies that the original dataset had four examples with parents $\mathbf{U}$ set to $\mathbf{u}$. From now on, we focus on the simplified problem of learning the individual parameters θ_x as this is technically sufficient for our purpose.

In *Bayesian* parameter estimation, the network parameters are themselves represented as random variables as in Figure 9.1. Here, the root corresponds to parameter θ_x and the leaves correspond to observations of X (i.e., examples). This is a naive Bayes structure, as in Figure 9.2. It has a prior density on the root, $\rho(\theta_x)$ and conditional distributions for the leaves, $\mathbb{P}(X_i{=}x \mid \theta_x) = \theta_x$. Since variable X is binary, we assume a Beta prior:

$$\rho(\theta_x) = \alpha \cdot [\theta_x]^{\alpha-1} \cdot [1 - \theta_x]^{\beta-1}$$

where α and β are the meta-parameters of the Beta prior. Further, $\alpha = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)+\Gamma(\beta)}$ is a normalizing constant, where Γ is the gamma function. To simplify our notation,

we will assume a non-informative Beta prior, where $\alpha = \beta = 1$. All of our results can be extended to arbitrary meta-parameters α and β for the Beta prior.

We now have all we need to perform Bayesian learning based on the posterior density $\rho(\theta_x \mid \mathcal{D})$, where $\mathcal{D}$ is an instantiation of some leaf nodes in Figure 9.1. For example, the MAP estimate corresponds to

$$\theta_x^{\text{map}} = \arg\max_{\theta_x} \rho(\theta_x \mid \mathcal{D}).$$

We will use $\mathbb{P}$ to represent the probability distribution over leaf nodes. Hence, $\mathbb{P}(\mathcal{D})$ is the marginal likelihood. There are closed forms for both $\rho(\theta_x \mid \mathcal{D})$ and $\mathbb{P}(\mathcal{D})$ that we omit here, but see [Dar09, KF09, Mur12, Bar12] for details.

Finally, for the purposes of learning, it suffices to know how many times X appears positively or negatively in a dataset (and not the order in which they appeared). Hence, when needed, we may denote a dataset by $\mathcal{D}_N$, to indicate the size of the dataset N, and also by $\mathcal{D}_{N;i}$, to further indicate the number of positive cases i that appear in it. Subsequently, when a dataset has i positive examples out of N, we also know that $N - i$ of them are negative. Finally, we let θ_N denote the parameter estimate obtained from a dataset with N examples, and also by $\theta_{N;i}$ to indicate the number i of positive examples. In the case of MAP parameter estimates (with a Beta prior, and meta-parameters $\alpha = \beta = 1$), the resulting parameter estimate is $\theta_{N;i} = \frac{i}{N}$.

9.3 A New SDP for Robust Learning

The SDP and E-SDP, of Equations 2.1 (from Chapter 2.2) and 5.2 (from Chapter 5.1), were originally proposed for quantifying the robustness for threshold-based decisions. In this section, we introduce a new form of the SDP which is based on quantifying the robustness of a parameter estimate in the face of additional data. We refer to this new SDP as the *same-parameter probability* (SPP).

Table 9.1: A list of all datasets $\mathcal{D}_M$ over $M{=}3$ instances, and the resulting marginal likelihoods and MAP estimates, for the case $N{=}7$ & $i{=}3$. Rows with estimates $\theta_{N+M} \in [\theta_N - 0.1, \theta_N + 0.1]$ in bold.

$\mathcal{D}_M$	j/M	$\theta_{N+M;i+j}$	$\mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i})$
x, x, x	3/3	$\frac{i+3}{N+3} = 60\%$	$\frac{i+1}{N+2}\frac{i+2}{N+3}\frac{i+3}{N+4} = 0.1200$
$x, x, \bar{x}$	2/3	$\frac{i+2}{N+3} = \mathbf{50\%}$	$\frac{i+1}{N+2}\frac{i+2}{N+3}\frac{N-i+1}{N+4} = \mathbf{0.1010}$
$x, \bar{x}, x$	2/3	$\frac{i+2}{N+3} = \mathbf{50\%}$	$\frac{i+1}{N+2}\frac{N-i+1}{N+3}\frac{i+2}{N+4} = \mathbf{0.1010}$
$x, \bar{x}, \bar{x}$	1/3	$\frac{i+1}{N+3} = \mathbf{40\%}$	$\frac{i+1}{N+2}\frac{N-i+1}{N+3}\frac{N-i+2}{N+4} = \mathbf{0.1212}$
$\bar{x}, x, x$	2/3	$\frac{i+2}{N+3} = \mathbf{50\%}$	$\frac{N-i+1}{N+2}\frac{i+1}{N+3}\frac{i+2}{N+4} = \mathbf{0.1010}$
$\bar{x}, x, \bar{x}$	1/3	$\frac{i+1}{N+3} = \mathbf{40\%}$	$\frac{N-i+1}{N+2}\frac{i+1}{N+3}\frac{N-i+2}{N+4} = \mathbf{0.1212}$
$\bar{x}, \bar{x}, x$	1/3	$\frac{i+1}{N+3} = \mathbf{40\%}$	$\frac{N-i+1}{N+2}\frac{N-i+2}{N+3}\frac{i+1}{N+4} = \mathbf{0.1212}$
$\bar{x}, \bar{x}, \bar{x}$	0/3	$\frac{i}{N+3} = 30\%$	$\frac{N-i+1}{N+2}\frac{N-i+2}{N+3}\frac{N-i+3}{N+4} = 0.2100$

Consider Table 9.1, where we assume an initial dataset $\mathcal{D}_{N;i}$ with $i{=}3$ positive instances out of $N{=}7$, leading to the estimate $\theta_N = \frac{3}{7} = 42.86\%$. Suppose that we consider this parameter stable, if after observing $M{=}3$ more instances, the resulting parameter estimate is within $\theta_N \pm 10\%$.

In each row of Table 9.1, we have enumerated each possible dataset $\mathcal{D}_M$ of size $M{=}3$ that we can potentially obtain, in addition to the initial dataset $\mathcal{D}_N$. Each dataset $\mathcal{D}_M$ has a marginal likelihood $\mathbb{P}(\mathcal{D}_M \mid \mathcal{D}_N)$, i.e., the probability of observing the dataset $\mathcal{D}_M$ given that we have already observed the dataset $\mathcal{D}_N$. Further, each dataset $\mathcal{D}_M$, when concatenated to the dataset $\mathcal{D}_N$, leads to a new parameter estimate, θ_{N+M}. There are 6 cases in Table 9.1 where estimates θ_N and θ_{N+M} remain within the threshold. By accumulating the corresponding marginal likelihoods, $0.1010 + 0.1010 + 0.1212 + 0.1010 + 0.1212 + 0.1212$, we get an SPP of 0.6667. Hence, there is a probability of 66.67% that our parameter estimates do

not exceed our threshold, given $M = 3$ additional data examples.

We are now ready to formalize the SPP. Let $\mathcal{D}_N$ denote a dataset of N instances that we have already observed, and let θ_N denote the parameter estimate that we obtain from it. Let $\mathcal{D}_M$ denote a dataset of M additional instances that we can potentially collect, and let θ_{N+M} denote the parameter estimate that we would obtain after concatenating the datasets $\mathcal{D}_N$ and $\mathcal{D}_M$. We want to know whether our original parameter θ_N is robust to the additional M data examples. More precisely, we want to know whether $\mathsf{D}(\theta_N, \theta_{N+M}) \leq \varepsilon$ for a given distance function D and a given threshold ε. This is given by the SPP as follows.

$$\begin{aligned} \mathsf{spp}(M, \mathcal{D}_N, \varepsilon) &= \mathbb{P}\Big(\mathsf{D}(\theta_N, \theta_{N+M}) \leq \varepsilon\Big) \\ &= \sum_{\mathcal{D}_M} [\mathsf{D}(\theta_N, \theta_{N+M}) \leq \varepsilon] \cdot \mathbb{P}(\mathcal{D}_M \mid \mathcal{D}_N) \end{aligned} \tag{9.1}$$

where we have the indicator function:

$$[\mathsf{D}(\theta_N, \theta_{N+M}) \leq \varepsilon] = \begin{cases} 1 & \text{if } \mathsf{D}(\theta_N, \theta_{N+M}) \leq \varepsilon \\ 0 & \text{otherwise.} \end{cases}$$

The indicator function is 1 if the difference between our parameters remains within our threshold, and 0 otherwise.

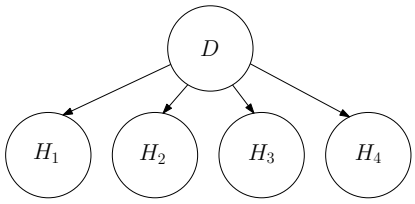
Later in the chapter, we assume a distance function D that is the absolute difference between the log-odds:

$$\mathsf{D}(\theta_N, \theta_{N+M}) = \left| \log \frac{\theta_N}{1 - \theta_N} - \log \frac{\theta_{N+M}}{1 - \theta_{N+M}} \right| \tag{9.2}$$

This corresponds to the CD-distance [CD05], which is known to have desirable properties.[1]

To summarize, the SPP enumerates all possible additional datasets $\mathcal{D}_M$, and accumulates their marginal likelihoods $\mathbb{P}(\mathcal{D}_M \mid \mathcal{D}_N)$ when the resulting parameter

[1]Bounds on the difference in the log-odds are desirable because they can bound the change in any query $Pr_\theta(\alpha \mid \beta)$ in a given Bayesian network, for arbitrary events α and β, for two different parameterizations θ [CD05, Dar09]. Hence, a bound on the difference in the log-odds of a local parameter is effectively a bound on the global distribution.



D	$Pr(D)$
$+$	0.60
$-$	0.40

D	H_i	$Pr(H_i\|D)$
$+$	$+$	0.80
$+$	$-$	0.20
$-$	$+$	0.20
$-$	$-$	0.80

Figure 9.2: A naive Bayes network.

estimate θ_{N+M} is within some distance from our original estimate, i.e., when $\mathsf{D}(\theta_N, \theta_{N+M}) \leq \varepsilon$. That is, the SPP is the *probability that our parameter estimate* θ_N *does not change too much*, given the M additional data examples, i.e., $\mathcal{D}_M$.

9.4 Computing the SPP

Equation 9.1 does not tell us how to compute the SPP efficiently, as there are an exponential number of possible datasets $\mathcal{D}_M$ that we have to sum over (exponential in the number of additional instances, M). We now show how the SPP can be computed in time linear in M.

We first consider a simpler case, the example naive Bayes network from Figure 9.2. As we shall see, this naive Bayes network exhibits additional structure that we can exploit to efficiently compute the SDP (and later the SPP). In particular, in the example of Figure 9.2, the feature variables H_i share the same CPT. Consider two instantiations $\mathbf{h}$ and $\mathbf{h}'$, where the the number of positive features is the same. In this case, the decision probabilites are the same, i.e., $Pr(d \mid \mathbf{h}) = Pr(d \mid \mathbf{h}')$. Further, the likelihoods of the features are the same, i.e., $Pr(\mathbf{h}) = Pr(\mathbf{h}')$. In fact, when all feature CPTs are identical, the probability of a decision, and the likelihood of the features observed, depend only on the number of positive features, not in the order that they were observed in. This is the type of structure that we can exploit, to compute the SDP more efficiently (as compared

to Equation 2.1).

Proposition 2. *Given a threshold $Pr(D{=}d \mid \mathbf{e}) \geq T$ in a naive Bayes network, with decision variable D, evidence $\mathbf{e}$, and threshold T, along with a set of unobserved variables $\mathbf{H}$ whose CPTs $\theta_{H|D}$ are identical, we have the following expression for the same-decision probability:*

$$\mathsf{SDP}(d, \mathbf{H}, \mathbf{e}, T) = \sum_{j=0}^{n} \binom{n}{j} \cdot [Pr(d \mid \mathbf{h}_j, \mathbf{e}) \geq T] \cdot Pr(\mathbf{h}_j \mid \mathbf{e})$$

where $\mathbf{h}_j$ denotes an instantiation where j features appear positively.

A proof of Proposition 2 appears in Appendix B. Proposition 2 gives us a way to compute the SDP for a special class of naive Bayes networks, in time that is *linear* in the number of features. This is a significant savings compared to Equations 2.1 and 5.2, where both the SDP and E-SDP are NP-hard problems, even in naive Bayes networks, as we discussed in Chapters 4 and 6 and subsequently proved in Chapter 7.

Figure 9.1 depicts our parameter estimation problem, which shares a similar naive Bayes structure, and a similar dependence between the variable of interest, θ_x, and the observables, X_i. The following theorem shows us how to efficiently compute the SPP efficiently, as in Proposition 2.

Theorem 4. *Given a dataset $\mathcal{D}_{N;i}$ and the corresponding parameter estimate $\theta_{N;i}$, the SPP of the estimate θ_{N+M} remaining within a threshold ε, given M additional data examples, is:*

$$\mathsf{spp}(M, \mathcal{D}_{N;i}, \varepsilon) = \sum_{j=0}^{M} \binom{M}{j} \cdot [\mathsf{D}(\theta_N, \theta_{N+M}) \leq \varepsilon] \cdot \mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i})$$

where we have the marginal likelihoods $\mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i})$:

$$\mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i}) = \frac{\Gamma(N+2)}{\Gamma(N+M+2)} \frac{\Gamma(i+j+1)}{\Gamma(i+1)} \frac{\Gamma(N-i+M-j+1)}{\Gamma(N-i+1)} \tag{9.3}$$

for noninformative Beta priors.

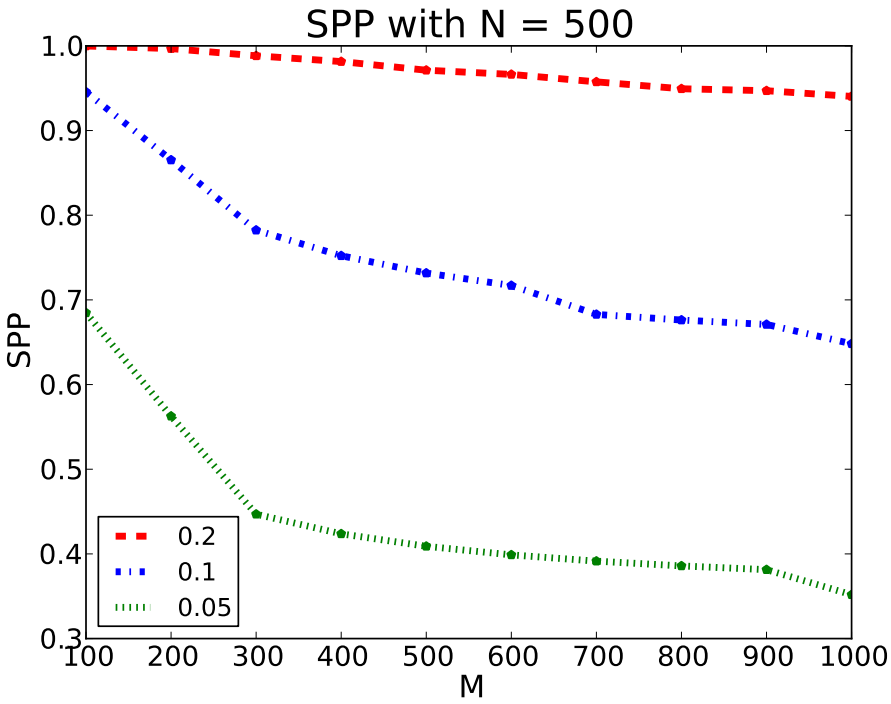


Figure 9.3: SPP: with a small initial dataset over varying ε

A proof of Theorem 4 is provided in Appendix B. Given that $\frac{\Gamma(s+t)}{\Gamma(s)} = s \cdot (s+1) \cdots (s+t-1)$, Theorem 4 gives us a way to compute the same-decision probability,[2] in time that is only linear in M.

9.5 Do We Have Enough Data?

We will now show how the SPP can be used to tell us whether we have collected enough data. In particular, we present several use cases for a decision maker that has estimated some parameters, and is considering whether or not to collect additional data. For these use cases, unless otherwise stated, we have assumed that the true parameter is θ_x=0.5, and accordingly, that our initial dataset $\mathcal{D}_N$ is split evenly between positive and negative instances, i.e., $i = \frac{N}{2}$. Further, we assumed thresholds ε of $0.05, 0.1$ and 0.2, on the log-odds difference between the

[2]To help avoid numerical overflow and underflow, we use the log-sum-exp trick; see, e.g., [Mur12].

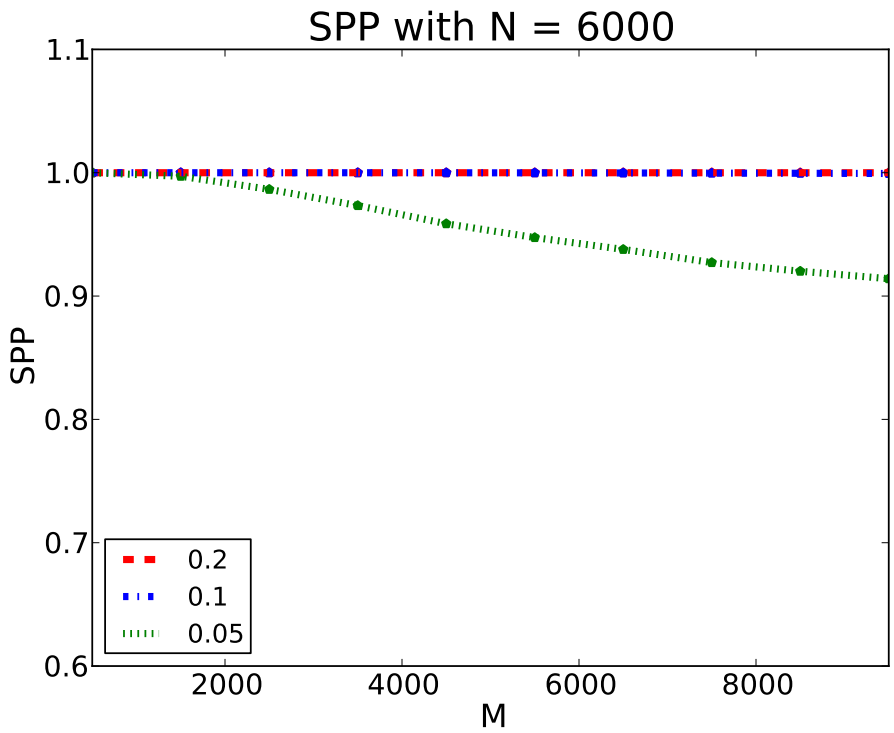


Figure 9.4: SPP: with a large initial dataset over varying ε

parameter estimates, as in Equation 9.2 (we assume logarithms with base 2). For a log-odds threshold of 0.1, and an initial parameter estimate θ_N of 0.5, the new estimate θ_{N+M} must have a probability from 48.27% to 51.73%, to remain within our threshold.

Size of initial dataset. Suppose that we have estimated a parameter θ_N based on an initial dataset of size $N{=}500$. We are now offered an additional $M = 100$ examples at some cost, and wish to assess the potential impact of these new examples on the estimate. The SPP can be used for this purpose, which comes out to be 94.55% using Theorem 4 (assuming $\varepsilon = .1$). This probability can then form the basis of a decision as to whether we should acquire the additional data or not.

One would expect the SPP to be lower if the additional data was larger, say, containing $M = 500$ examples. The question though is how much lower? Again, using Theorem 4, we calculate this to be 73.16%. More generally, Figure 9.3 plots

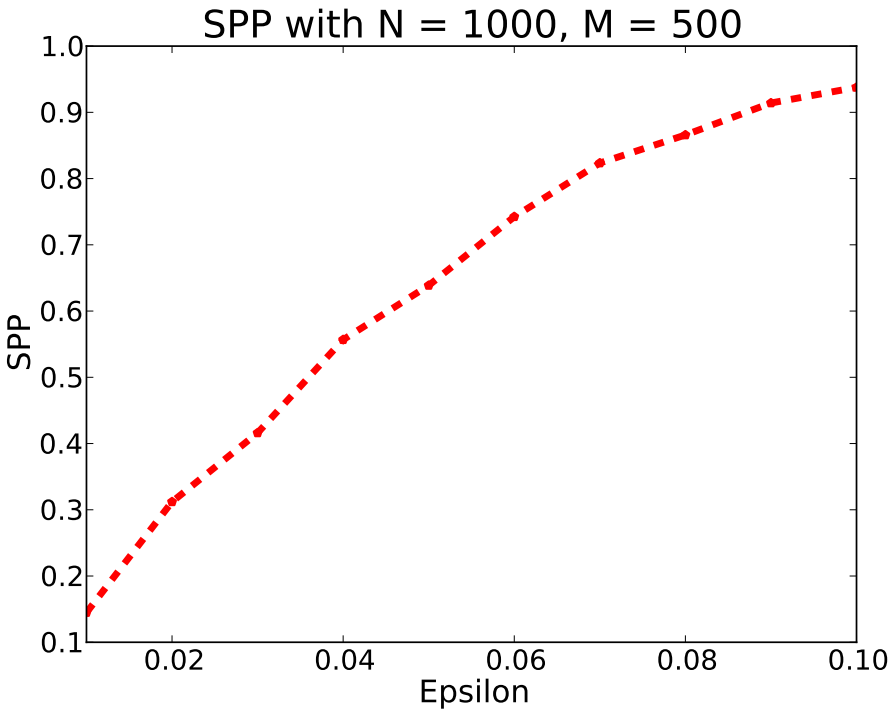


Figure 9.5: SPP: impact of the threshold ε

the SPP for different sizes M of additional data, assuming an initial dataset of size $N = 500$.

Suppose now that we start with a much larger initial dataset of size $N{=}6,000$. In Figure 9.4, we can see that even if $M{=}10,000$ additional examples are made available, our parameter estimates are very likely to remain within a threshold of $\varepsilon{=}0.1$—the probability is almost 1.0. We can see that with a large enough initial dataset, observing any more examples is very unlikely to change our parameterization. Going further, we obtain an SPP of 99.85% when offered $M{=}15,000$ additional examples, an SPP of 99.55% for $M{=}50,000$ additional examples, and an SPP of 99.43% for $M{=}100,000$ additional examples. These examples illustrate how we can use the SPP, to determine when we have acquired enough data.

Tight vs. loose confidence intervals. Consider the following two scenarios: 1) we are fine with a parameterization that is more noisy, and 2) we want a very tight parameterization. How much data do we need in each case? We clearly need

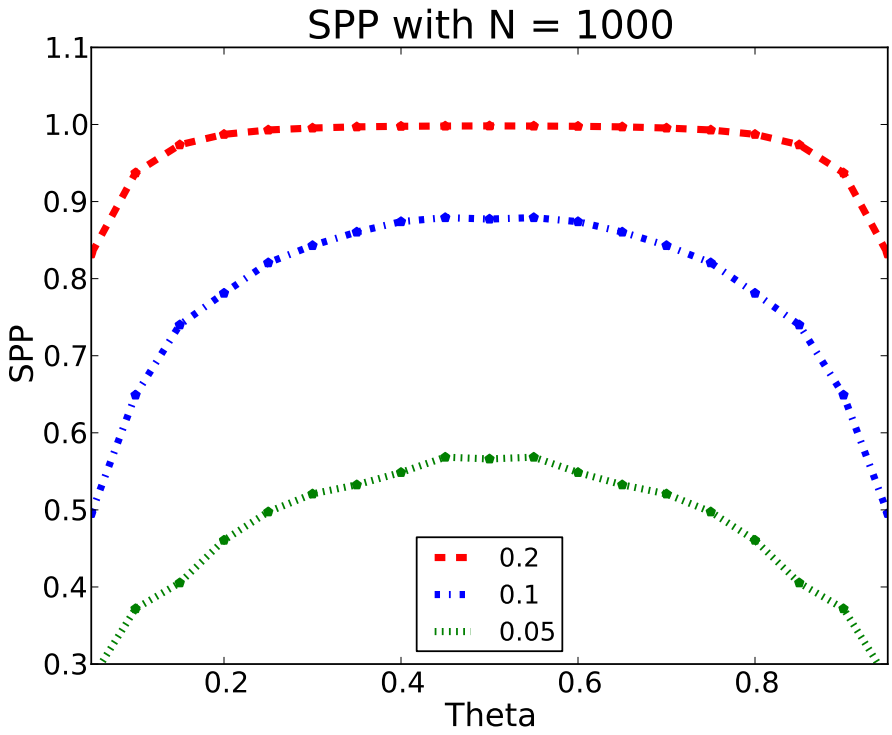


Figure 9.6: SPP: impact of the true parameter θ_x over varying threshold ε

more data for Scenario 2 — but can we quantify how much more we need? In Figure 9.5, we have fixed $N{=}1,000$, and have made available $M{=}500$ additional examples. From *right-to-left* on the x-axis, we tighten the threshold from $\varepsilon{=}0.10$ to $\varepsilon{=}0.01$. Note that for a loose threshold of 0.10,[3] the SPP is relatively large: 0.9377, whereas for a tight threshold of 0.01,[4] the SPP is 0.1448, indicating that if for Scenario 1 we had the option to purchase 500 additional data examples for observation, we shouldn't, whereas for Scenario 2, those 500 additional data examples are crucial.

Extreme data parameters. What if we had seen a dataset that caused us to learn a more extreme parameter θ_x? How does our need to observe additional data points differ than if we had learned a relatively uniform θ_x? Here, we vary the underlying parameter θ_x, and have fixed the initial dataset $\mathcal{D}_N$ to $N{=}1,000$

[3]For a log-odds threshold of 0.10, the new estimate θ_{N+M} must range from the probabilities 48.27% to 51.73%, to remain within our threshold.

[4]For a log-odds threshold of 0.01, the new estimate must range from 49.83% to 50.17%.

instances, where the number of positive instances matches the underlying parameter, i.e., $i = N \cdot \theta_x$. Further, we make M=1000 additional instances available.

In Figure 9.6, we see that the parameter estimate θ_N is most robust when the underlying parameter θ_x is 0.5 — for $\varepsilon = 0.1$, the SPP is 0.9377. However, as the parameter becomes more extreme (towards 0 or towards 1), then the parameter estimate θ_N is less robust — for example, when the underlying parameter θ_x is set to 0.9, the SPP drops to 0.7535 (for $\varepsilon = 0.1$), suggesting that as our learned parameter θ_x becomes more extreme, we need additional data points in order to validate it. The intuition behind this occurrence is that when we have more extreme parameters, those parameters are generally less robust as it is easier for additional data instances to cause a greater shift in the distance between the initial parameter estimate and the complete parameter estimate.

9.6 How Much More Data Do We Need?

Suppose that we obtained a parameter estimate θ_N based on a dataset $\mathcal{D}_N$, then computed the corresponding SPP and found it not high enough. The next question we ask is: how much more data do we need, before we *are* confident about our estimate?

We formalize this question intuitively as follows. We have the option of obtaining M additional examples, but we want to know if there is an $L < M$ that renders the additional examples $M - L$ somewhat redundant as far as our confidence is concerned? In other words, do we *expect* the estimate θ_{N+L} to be robust enough against the additional $M - L$ examples? This is an expectation as we do not know what the L examples will be; all we know is their count. This leads to

the *expected* SPP (E-SPP), which we define as follows.

$$\begin{aligned}&\mathsf{e\text{-}spp}(L, M, \mathcal{D}_N, \varepsilon) = \mathbb{E}[\mathsf{sdp}(M - L, \mathcal{D}_{N+L}, \varepsilon)] \\ &= \sum_{\mathcal{D}_L} \mathsf{spp}(M - L, \mathcal{D}_{N+L}, \varepsilon) \cdot \mathbb{P}(\mathcal{D}_L \mid \mathcal{D}_N) \end{aligned} \tag{9.4}$$

As with the SPP, we enumerate all possible datasets $\mathcal{D}_L$, and concatenate each with our existing dataset $\mathcal{D}_N$. We then take the weighted average of $\mathsf{spp}(M - L, \mathcal{D}_{N+L}, \varepsilon)$. The expected SPP can be computed efficiently as well.

Theorem 5. *Given a dataset $\mathcal{D}_{N;i}$ and a threshold ε, the expected SPP after observing L more instances, with respect to a total of additional M instances, is:*

$$\begin{aligned}&\mathsf{e\text{-}spp}(L, M, \mathcal{D}_{N;i}, \varepsilon) \\ &= \sum_{k=0}^{L} \binom{L}{k} \cdot \mathsf{spp}(M - L, \mathcal{D}_{N+L;k+i}, \varepsilon) \cdot \mathbb{P}(\mathcal{D}_{L;k} | \mathcal{D}_{N;i})\end{aligned}$$

The marginal likelihoods $\mathbb{P}(\mathcal{D}_{L;k} \mid \mathcal{D}_{N;i})$ can be computed, as in Equation 9.3. The expected-SPP can then be computed in time linear in L and M.

This brings the question of whether we can *optimize* the number of examples L. That is, can we find a smallest number of additional instances L, so that the resulting estimates are expected to be robust? For example, can we find a smallest L so that the expected-SPP is at least 95%?

Since the expected-SPP is efficient to compute ($O(L \cdot M)$), we can simply (1) upper bound the optimal L, and then (2) perform binary search. We can upper bound the optimal L, by simply starting with an initial guess (say L=1), and then doubling our guess, until our expected-SPP exceeds our threshold (say 95%); this takes $O(\log L^\star)$ expected-SPP queries, where $L^\star$ is the optimal value of L. Once we have this upper-bound, we can then perform binary search to find the optimal value of L, which again takes $O(\log L^\star)$ expected-SPP queries. Hence, finding the optimal number of additional examples L has a polynomial time complexity, $O(L^\star \cdot M \cdot \log L^\star)$.

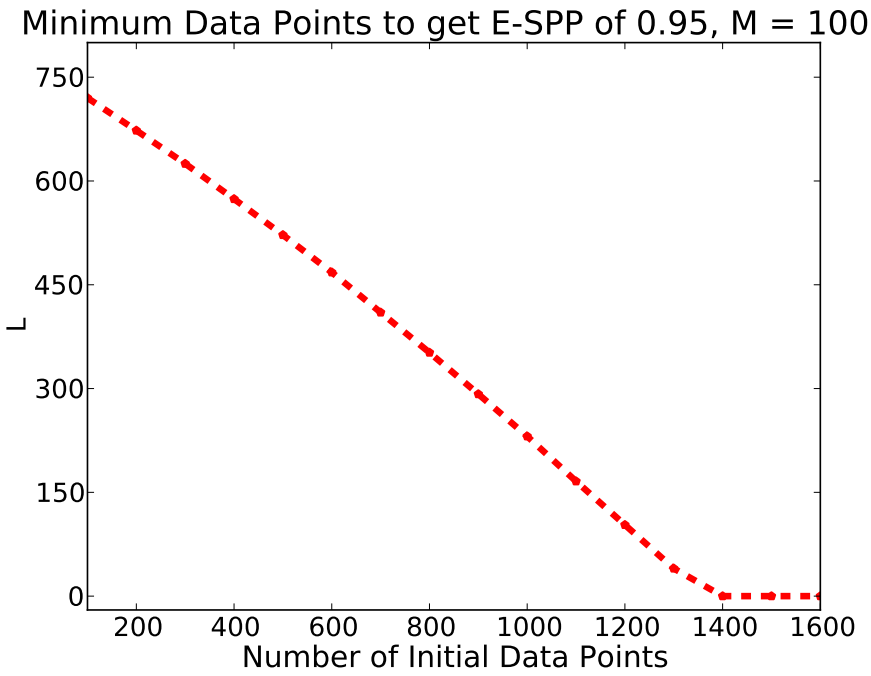


Figure 9.7: E-SPP: with underlying parameter θ_x=0.5 and a fixed $M = 1000$ and $\varepsilon = 0.1$.

We now illustrate the results of this section using a concrete example. In Figure 9.7, from left-to-right on the x-axis, we increase the size N of our initial dataset $\mathcal{D}_N$. On the y-axis, we plot the minimum number of additional instances L that we would need to observe, to obtain an expected SPP of at least 95% (with respect to a total of M=1,000 additional examples). According to this plot, with a very small number of initial examples, say N=100, we need many additional examples, almost L=750, to obtain parameter estimates that are robust against additional data. As we increase the amount of initial data, the amount of data needed drops steadily, until we have an initial dataset of N=1,400 examples. At this point, our parameter estimates are robust, and we need no additional data examples.

We finally remark that the expected-SPP is linear, up to the point where we have enough initial data for our initial estimates to be confident. We believe that 1) this tells us that it does not matter so much *what* we've seen, compared to *how*

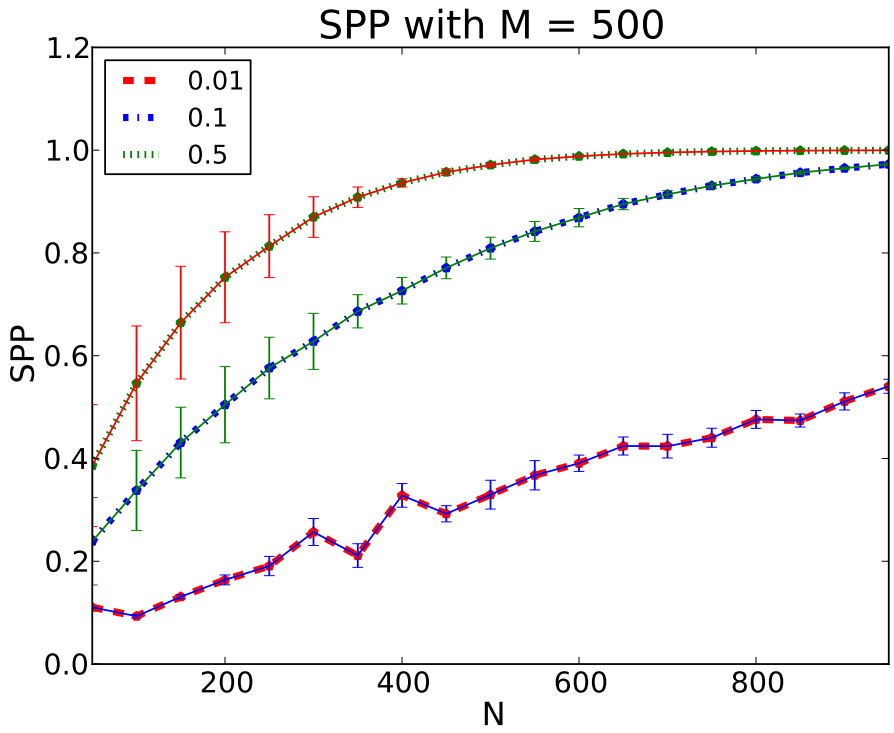


Figure 9.8: SPP: Impact of increasing N with a fixed M. Variances are depicted using error bars and ε is set to 0.1.

much we've seen and that 2) the data points we are about to acquire are equally as important as the data points we have already acquired.

9.7 Simulation Results

We wish to validate our approach for computing the SPP to make sure it is practically useful — if given some generated data we learn some θ_N and then compute the SPP, how often will the fully learned parameter match with our initial θ_N? We hypothesize that the percentage of the time that a completely learned parameter θ_{N+M} matches our initial θ_N will be close to the initially computed SPP. We empirically evaluate the same-parameter probability on simulated datasets. Here, we varied the size N of the initial dataset $\mathcal{D}_N$, which we simulated from

several fixed parameters, $\theta_x = \{0.5, 0.1, 0.01\}$,[5] which we want to estimate from the data. For each size N, we generated 50 samples, in order to compute an *empirical SPP*. That is, for each sample, we simulated $5,000$ pairs of datasets, an initial dataset $\mathcal{D}_N$ of size N and a second dataset $\mathcal{D}_M$ of size $M{=}500$. For each of the $5,000$ pairs $(\mathcal{D}_N, \mathcal{D}_M)$ in a sample, we recorded whether the resulting estimate $\mathcal{D}_{N+M}$ resulted in the same-decision, i.e., whether it was within our threshold of $\varepsilon{=}0.1$ on the log-odds difference. For each of our 50 samples, we recorded the resulting sample mean (i.e., the empirical SPP), and computed the mean and variance of the sample means, which we report in Figure 9.8. We further plotted the average SDPs, as predicted by Theorem 4.

As expected, the sample means (empirical SPPs) closely match the predicted SPP. Further, as we increase the size of the initial dataset $\mathcal{D}_N$, we see that (1) the SPP and empirical SPP grow, indicating an increasing robustness of our initial parameter estimates, and (2) the variances of the sample means grow smaller, indicating an increasing robustness in the same-parameter probability itself.

Additionally, we validate our approach for computing the E-SPP as well. If given some generated data we learn some θ_N and then select some L to compute the E-SPP, how often will the half learned parameter θ_{N+L} match with the final θ_{N+M+L}? we provide an empirical evaluation similar to the one provided in Section 9.7. Again, we simulate datasets from several fixed parameters, $\theta_x = \{0.5, 0.3, 0.1\}$,[6] which we want to estimate. We fix the size of the initial dataset $\mathcal{D}_N$ to $N{=}200$, and vary the size L of the additional dataset that we provide. The resulting robustness of the estimate θ_{N+L} is computed with respect to an additional $M{=}500$ instances. Again, we generate 50 samples, where we simulated $5,000$ triples of datasets $(\mathcal{D}_N, \mathcal{D}_L, \mathcal{D}_M)$. For each triple, we recorded

[5] We found that using a parameter of $\theta_x = 0.1$ and $\theta_x = 0.01$ had nearly identical results to respectively using parameters $\theta_x = 0.9$ and $\theta_x = 0.99$.

[6] We found that using a parameter of $\theta_x = 0.3$ and $\theta_x = 0.1$ had nearly identical results to respectively using parameters $\theta_x = 0.7$ and $\theta_x = 0.9$.

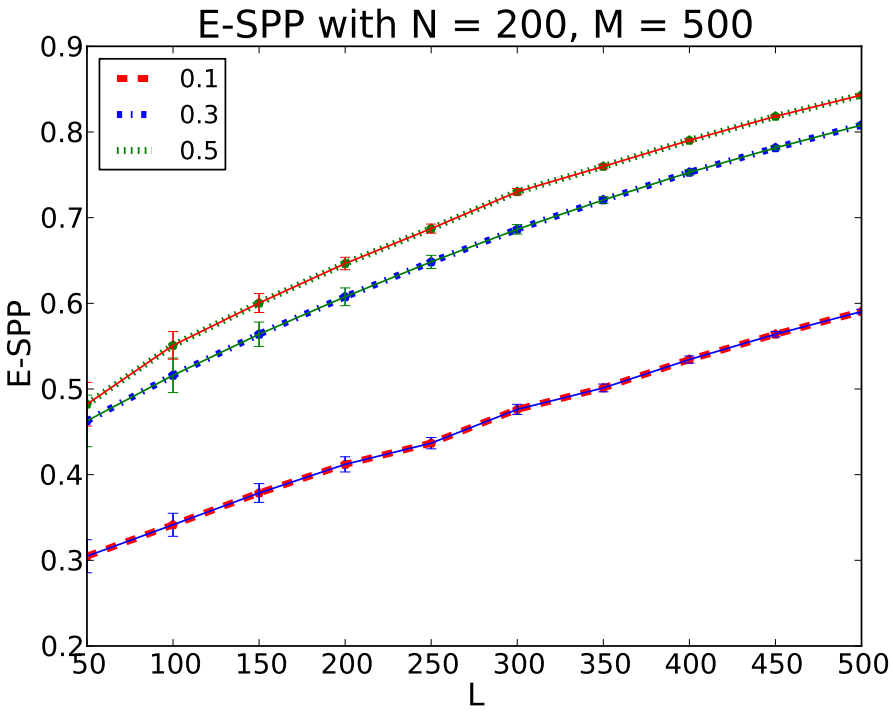


Figure 9.9: E-SPP: Impact of increasing L with a fixed N and M. Variances are depicted using error bars and ε is set to 0.1.

whether the resulting estimate θ_{N+L+M} resulted in the same-decision, i.e., whether $\mathcal{D}(\theta_{N+L}, \theta_{N+L+M}) \leq \varepsilon$. For each of our 50 samples, we recorded the sample mean (i.e., the empirical expected-SPP), and computed the mean and variance of the sample means, which we report in Figure 9.9. We further plotted the average expected-SPPs, as predicted by Theorem 5.

Again, the sample means (i.e., the empirical expected-SPPs) closely match the predicted expected-SPP. Moreover, as the size of the additional dataset $\mathcal{D}_L$ increases, the expected-SPP increases as well, as expected. Compared to the empirical SPPs, the empirical expected-SPPs have much smaller variance, suggesting that the expected-SPP is robust itself.

9.8 Related Work

In [Dar09], there is discussion on how to find the minimum sample size in order to guarantee some confidence in a parameter estimate. This work is solving a different problem. That is, given a margin of error, it asks for the sample size that guarantees a certain confidence level (independently of a current estimate). In our work, we are given (1) a "current estimate" based on (2) a "current dataset", and we answer two key questions on how this estimate is likely to "change" based on additional data. Both questions (SPP and E-SPP) take more input (1 & 2), are sensitive to this input, and yield answers that are beyond the scope of statistical techniques for estimating a proportion.

Another simple measure of the confidence in a parameter estimate is the variance of the posterior distribution $\rho(\theta \mid \mathcal{D})$. If we assume a Beta prior $\rho(\theta)$, then given complete data, the posterior is a Beta distribution also. A Beta distribution with parameters α and β has a variance $\frac{\alpha\beta}{(\alpha+\beta)^2+(\alpha+\beta+1)}$, with smaller variance entailing higher confidence. Another, more related, measure of confidence is a credible interval (which is a Bayesian analogue of the frequentist confidence interval); see, e.g., [Was11, Mur12]. In particular, if we have an interval (a, b), then it is a 95% credible interval, given an estimate θ and dataset $\mathcal{D}$, if:

$$Pr(\theta \in (a, b) \mid \mathcal{D}) = 95\%.$$

For a Beta distribution, a credible interval can be determined by evaluating the inverse of the Beta cumulative density function (which in turn can be computed using, e.g., Newton's method). The tighter our credible interval is, the more confident we are in our estimate θ. In contrast, we view the SPP as a more refined measure of confidence, which takes into account the explicit availability of additional data examples. Moreover, the SPP can be extended, via the expected SPP, to suggest a specific number of additional examples that would be needed to obtain a certain level of confidence.

There are also statistical techniques to determine a sample size that is large enough to guarantee some level of confidence in a parameter estimate; see, e.g., [Dar09]. That is, given a margin of error, these techniques provide a sample size that guarantees a certain confidence level (which is generally independent of the current estimate). In contrast, with the SPP and E-SPP, we are given (1) a *current estimate* based on (2) a *current dataset*, and we want to answer two questions (i.e., the SPP/E-SPP) on how this estimate is likely to change based on additional data. This leads to a measure of robustness that is more precise, and more sensitive to the input, compared to the more statistical techniques mentioned above.

The types of symmetries that we exploited in Section 9.4, are among those studied in *lifted probabilistic inference*. This field is concerned with the systematic exploitation of symmetries and regularities for efficient inference in large and complex models, such as those found in statistical relational learning [Poo03, Ker12, NV14].

9.9 Conclusion

In this chapter, we provided a new instance of the SDP, called the same-parameter probability (SPP), and used it to quantify the robustness of parameter estimates in Bayesian networks. We showed how the SPP can be used to decide whether enough data has been collected, in order to learn the parameters of a Bayesian network robustly (in the face of new data). We further showed how the expected SPP can be used to decide how much more data needs to be collected, if one's parameter estimates are not yet robust. In contrast to the SDP for robust decision-making, which is computationally hard, we find that the SPP for robust learning admits simple and efficient reasoning algorithms.

CHAPTER 10

Conclusion and Future Work

In this thesis, we have shown how the *Same-Decision Probability* (SDP), the probability that we would make the same decision had we known what we currently do not know, can be used as an information gathering tool to make robust decisions. We have done so by comparing it against standard information gathering criteria and demonstrating that the SDP can act as an information gathering criterion as well — we showed multiple concrete examples of its usefulness as both a stopping criterion and a selection criterion. As a stopping criterion, we have demonstrated that the SDP allows us to determine when no further observations are necessary. As a selection criterion, usage of the SDP allows us to select observations that lead to robust decisions. We showed that the SDP can give us additional insight on 1) whether additional information should be gathered and 2) which pieces of additional information should be gathered.

Since we have justified the usage of the SDP as both a stopping criterion and selection criterion, we also proposed an exact algorithm for its use as a stopping criterion in general Bayesian networks and an exact algorithm for its use as a feature selection algorithm (selection criterion) in Naive Bayes networks. Experimental results showed that the stopping criterion algorithm has comparable running time to the previous approximate algorithm and is also much faster than the naive brute-force algorithm. In addition, we showed empirically that in the context of classification, our feature selection algorithm can be used to significantly reduce the cost of observing features while minimally affecting the classification

accuracy. Our final empirical result showed that this feature selection algorithm is more suitable than traditional feature selection algorithms when the goal is to make stable decisions given a limited budget.

In addition, we presented several new complexity results that indicate the hardness of computing the SDP in naive Bayes networks, as well as the general hardness of computing the value of information in general Bayesian networks. Finally, we showed how we can apply the SDP as an information gathering criterion in practical problems. First, we applied the SDP to an interactive tutoring system to reduce the number of questions that needed to be asked. Subsequently, we presented closed-form solutions for computing the SDP in certain settings and showed how to use these solutions to allow us to decide how much additional data is necessary in order to train a robust Bayesian network.

For future work, we believe that there is much to be done algorithmically for computing the SDP. For highly connected networks, our algorithm will still perform poorly due to a shallow search tree and less opportunities for search space pruning. We are interested in finding other ways to disconnect the network — trading time for space perhaps with recursive conditioning. For maximizing the expected SDP, the work of extending the naive Bayes algorithm to general Bayesian networks still remains to be done. Additionally, there are certain settings where the SDP can be computed with a closed-form solution — there would be much benefit to discovering these domains and the corresponding closed-form solutions. For applications, we feel that the SDP can be applicable to several domains, ranging from medical diagnosis to product troubleshooting, in order to determine when additional tests are needed and which additional tests should be ordered.

We believe that continued study and utilization of the SDP will provide a fresh and unique insight for making robust decisions.

APPENDIX A

Proofs of Theorems in Chapter 7

We first provide the proof for Theorem 1:

Proof of Theorem 1. We reduce the number partition problem defined by [Kar72] to computing the SDP in a Naive Bayes model. Suppose we are given a set of positive integers $c_1, \ldots, c_n$, and we wish to determine whether there exists $I \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in I} c_i = \sum_{j \notin I} c_j$. We can solve this by considering a Naive Bayes network with a binary class variable D having uniform probability, and binary attributes $H_1, \ldots, H_n$ having CPTs leading to weights of evidence $w_{H_i=T} = c_i$ and $w_{H_i=F} = -c_i$. The construction of these CPTs can be done by solving the following system of equations (see Exercise 3.27 in [Dar09]):

$$
\begin{aligned}
c_i &= \log \frac{Pr(H_i = T \mid D = T)}{Pr(H_i = T \mid D = F)} \\
-c_i &= \log \frac{Pr(H_i = F \mid D = T)}{Pr(H_i = F \mid D = F)} \\
1 &= Pr(H_i = T \mid D = T) + Pr(H_i = F \mid D = T) \\
1 &= Pr(H_i = T \mid D = F) + Pr(H_i = F \mid D = F)
\end{aligned}
$$

This leads to

$$
\begin{aligned}
Pr(H_i = T \mid D = F) = Pr(H_i = F \mid D = T) &= \frac{1}{2^{c_i} + 1} \\
Pr(H_i = T \mid D = T) = Pr(H_i = F \mid D = F) &= 1 - \frac{1}{2^{c_i} + 1}
\end{aligned}
$$

Since these CPTs have been defined such that $w_{H_i=T} = c_i$ and $w_{H_i=F} = -c_i$, the set of integers can be partitioned if there is an instantiation $\mathbf{h} = \{h_1, \ldots, h_n\}$ with $\sum_{i=1}^{n} w_{h_i} = 0$ since I would then include all indices i with $h_i = T$ in this case.

The Naive Bayes network satisfies a number of properties that we shall use next. First, $\sum_{i=1}^{n} w_{h_i}$ is either 0, ≥ 1, or ≤ -1 since all weights w_{h_i} are integers. Next, if $\sum_{i=1}^{n} w_{h_i} = c$, then $\sum_{i=1}^{n} w_{h'_i} = -c$ where $h'_i \neq h_i$. Finally, $Pr(h_1, \ldots, h_n) = Pr(h'_1, \ldots, h'_n)$ when $h'_i \neq h_i$, as D has a uniform probability distribution and each leaf H_i has been defined with a symmetric CPT.

Consider now the following SDP (the last step below is based on the above properties):

$$
\begin{aligned}
&\mathsf{sdp}(D = T, \{H_1, \ldots, H_n\}, \{\}, 2/3) \\
&= \sum_{h_1,\ldots,h_n} [Pr(D = T \mid h_1, \ldots, h_n) \geq 2/3] Pr(h_1, \ldots, h_n) \\
&= \sum_{h_1,\ldots,h_n} [\log O(D = T \mid h_1, \ldots, h_n) \geq 1] Pr(h_1, \ldots, h_n) \\
&= \sum_{h_1,\ldots,h_n} \left[\sum_{i=1}^{n} w_{h_i} \geq 1 \right] Pr(h_1, \ldots, h_n) \\
&= \frac{1}{2} \sum_{h_1,\ldots,h_n} \left[\sum_{i=1}^{n} w_{h_i} \neq 0 \right] Pr(h_1, \ldots, h_n)
\end{aligned}
$$

We then have $\sum_{i=1}^{n} w_{h_i} = 0$ for some instantiation $h_1, \ldots, h_n$ iff

$$\sum_{h_1,\ldots,h_n} \left[\sum_{i=1}^{n} w_{h_i} \neq 0 \right] Pr(h_1, \ldots, h_n) < 1$$

Hence, the partitioning problem can be solved iff

$$\mathsf{sdp}(D = T, \{H_1, \ldots, H_n\}, \{\}, 2/3) < 1/2$$

□

We next provide proofs for Theorems 2 and 3.

Proof of Theorem 2. We show **D-EPT** is PP^{PP}-hard by reduction from the following decision problem **D-SDP**, which corresponds to the originally proposed notion of same-decision probability for threshold-based decisions [DC10].

D-SDP: Given a decision based on probability $Pr(d \mid \mathbf{e})$ surpassing a threshold T, a set of unobserved variables $\mathbf{H}$, and a probability p, is the same-decision probability:

$$\sum_{\mathbf{h}} [Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T] Pr(\mathbf{h} \mid \mathbf{e}) \tag{A.1}$$

greater than p?

Here, [.] denotes an indicator function which evaluates to 1 if the enclosed expression is satisfied, and 0 otherwise. **D-SDP** was shown to be PP^{PP}-complete by [CXD12].

This same-decision probability corresponds to an expectation with respect to the distribution $Pr(\mathbf{H} \mid \mathbf{e})$, using the reward function:

$$R(Pr(D \mid \mathbf{h}, \mathbf{e})) = \begin{cases} 1 & \text{if } Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T \\ 0 & \text{otherwise.} \end{cases}$$

Thus the same-decision probability is $\geq T$ iff this expectation is $\geq T$. □

Proof of Theorem 3. To show that **D-EPT** is in PP^{PP}, we provide a probabilistic polynomial-time algorithm, with access to a PP oracle, that answers the decision problem **D-EPT** correctly with probability greater than $\frac{1}{2}$. This proof generalizes and simplifies the proof given by [CXD12] for **D-SDP**.

Consider the following probabilistic algorithm that determines if $\mathbb{E} > N$:

1. Sample a complete instantiation $\mathbf{x}$ from the Bayesian network, with probability $Pr(\mathbf{x})$. We can do this in linear time, using forward sampling [Hen86].

2. If $\mathbf{x}$ is compatible with $\mathbf{e}$, we can use a PP-oracle to compute $t = R(Pr(D \mid \mathbf{h}, \mathbf{e}))$. First, the reward function R can be computed in polynomial time, by definition. Second, $Pr(D \mid \mathbf{h}, \mathbf{e})$ can be computed using a PP-oracle, since the inference is #P-complete [Rot96], and since $\text{P}^{\text{PP}} = \text{P}^{\#\text{P}}$.[1]

[1]Since #P is shown to be equally as powerful of an oracle as PP, therefore PP^{PP} is equal to

3. Define a function $a(t) = \frac{1}{2} + \frac{1}{2}\frac{t-N}{u-l}$, which defines a probability used by our probabilistic algorithm to guess whether $\mathbb{E} > N$ (see Lemma 1).

4. Declare that $\mathbb{E} > N$ with probability:

   - $a(t)$ if $\mathbf{x}$ is compatible with $\mathbf{e}$;
   - $\frac{1}{2}$ if $\mathbf{x}$ is not compatible with $\mathbf{e}$.

The probability of declaring $\mathbb{E} > N$ is:

$$r = \sum_{\mathbf{h}} a(t) Pr(\mathbf{h}, \mathbf{e}) + \frac{1}{2}(1 - Pr(\mathbf{e})) \tag{A.2}$$

which is greater than $\frac{1}{2}$ iff the following set of equivalent statements hold:

$$\begin{aligned}
\sum_{\mathbf{h}} a(t) Pr(\mathbf{h}, \mathbf{e}) &> \frac{Pr(\mathbf{e})}{2} \\
\sum_{\mathbf{h}} a(t) Pr(\mathbf{h} \mid \mathbf{e}) &> \frac{1}{2} \\
\sum_{\mathbf{h}} \left(\frac{1}{2} + \frac{1}{2}\frac{t-N}{u-l} \right) Pr(\mathbf{h} \mid \mathbf{e}) &> \frac{1}{2} \\
\sum_{\mathbf{h}} \left(\frac{1}{2}\frac{t-N}{u-l} \right) Pr(\mathbf{h} \mid \mathbf{e}) &> 0 \\
\sum_{\mathbf{h}} (t-N) Pr(\mathbf{h} \mid \mathbf{e}) &> 0 \\
\sum_{\mathbf{h}} R(Pr(D \mid \mathbf{h}, \mathbf{e})) Pr(\mathbf{h} \mid \mathbf{e}) &> N.
\end{aligned}$$

Thus $r > \frac{1}{2}$ iff $\mathbb{E} > N$. □

$$a(t) = \frac{1}{2} + \frac{t-N}{2(u-l)} \tag{A.3}$$

can thus transform any sampled reward into a probability.

$\mathrm{PP}^{\#\mathrm{P}}$ [Kwi09]. We use a #P oracle in place of a PP oracle, as [Rot96] proves that the problem **MAR**, computing the conditional probability of a node given evidence, is #P-complete. We can thus use the **MAR** oracle to calculate $Pr(D \mid \mathbf{e})$.

Lemma 1. *The function* $a(t) = \frac{1}{2} + \frac{1}{2}\frac{t-N}{u-l}$ *maps a reward* t *to a probability in* $[0, 1]$.

Proof. Values u and l are given, and denote upper and lower bounds on the reward t, but also the threshold N. Thus $\frac{t-N}{u-l}$ is in $[-1, 1]$. □

Note that $a(t)$ denotes a probability used by our algorithm to declare whether $\mathbb{E} > N$, which is higher or lower depending on the value of the reward $t = R(Pr(D \mid \mathbf{h}, \mathbf{e}))$.

APPENDIX B

Proofs of Theorems in Chapter 9

Proof of Proposition 2. Consider the same-decision probability of Equation 2.1:

$$SDP(d, \mathbf{H}, \mathbf{e}, T) = \sum_{\mathbf{h}} [Pr(d \mid \mathbf{h}, \mathbf{e}) \geq T] Pr(\mathbf{h} \mid \mathbf{e})$$

Let $\mathbf{h}_j$ represent an instantiation where features appear positively j times, and $\mathbf{h}'_j$ represent a different instantiation where features appear positively j times. Since both instantiations have the same number of positive features, $Pr(d \mid \mathbf{h}_j) = Pr(d \mid \mathbf{h}'_j)$ and $Pr(\mathbf{h}_j) = Pr(\mathbf{h}'_j)$, we can compute the sum of all such features, collectively. That is, we can compute $Pr(d \mid \mathbf{h}_j)$ and $Pr(\mathbf{h}_j)$ once for all instantiations $\mathbf{h}'_j$ that have the same number of positive features. If there are n features, then there are $\binom{n}{j}$ ways that we can see j positive features as an instantiation $\mathbf{h}_j$. Hence, we just need to consider the different positive counts where j features can appear positively, from 0 to n. This leads to the following expression for the same-decision probability:

$$\begin{aligned} & SDP(d, \mathbf{H}, \mathbf{e}, T) \\ & = \sum_{j=0}^{n} \binom{n}{j} \cdot [Pr(d \mid \mathbf{h}_j, \mathbf{e}) \geq T] \cdot Pr(\mathbf{h}_j \mid \mathbf{e}) \end{aligned}$$

which can be computed in time linear in n. □

Proof of Theorem 4. Consider the same-parameter probability of Equation 9.1,

which we repeat here.

$$SPP(M, \mathcal{D}_N, \varepsilon)$$
$$= \sum_{\mathcal{D}_{M;j}} [\mathsf{D}(\theta_{N;i}, \theta_{N+M;i+j}) \leq \varepsilon] \cdot \mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i})$$

Given an i.i.d. dataset $\mathcal{D}_M$, the estimate θ_{N+M} that one obtains is the same, whenever the number of times that variable X appears positively is the same. This is because the estimate depends only on the number of times that X appears in the dataset, not in the order that they appear in (which is a well-known property that follows from the i.i.d. assumption). For example, in Table 9.1, any row where the proportion of positive instances, j/M, is the same, we have that the marginal likelihood $\mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i})$ is the same, and further, the parameter estimate $\theta_{N+M;i+j}$ is the same. Note that there are $\binom{M}{j}$ ways that X can appear positively j times, in a dataset of size M. Hence, we have:

$$SPP(M, \mathcal{D}_N, \varepsilon)$$
$$= \sum_{j=0}^{M} \binom{M}{j} [\mathsf{D}(\theta_{N;i}, \theta_{N+M;i+j}) \leq \varepsilon] \cdot \mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i})$$

which can be computed in time linear in M.

Consider now the marginal likelihood $\mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i})$, with non-informative Beta priors ($\alpha = \beta = 1$). First, the marginal likelihood of dataset $\mathcal{D}_{N;i}$ is

$$\mathbb{P}(\mathcal{D}_{N;i}) = \frac{\Gamma(2)}{\Gamma(N+2)} \frac{\Gamma(i+1)}{\Gamma(1)} \frac{\Gamma(N-i+1)}{\Gamma(1)}$$

The marginal likelihood of the joint dataset $\mathcal{D}_{N;i}, \mathcal{D}_{M;j}$ is

$$\mathbb{P}(\mathcal{D}_{N;i}, \mathcal{D}_{M;j}) = \mathbb{P}(\mathcal{D}_{N+M;i+j})$$
$$= \frac{\Gamma(2)}{\Gamma(N+M+2)} \frac{\Gamma(i+j+1)}{\Gamma(1)} \frac{\Gamma(N-i+M-j+1)}{\Gamma(1)}$$

Hence,

$$\mathbb{P}(\mathcal{D}_{M;j} \mid \mathcal{D}_{N;i})$$
$$= \frac{\Gamma(N+2)}{\Gamma(N+M+2)} \frac{\Gamma(i+j+1)}{\Gamma(i+1)} \frac{\Gamma(N-i+M-j+1)}{\Gamma(N-i+1)}$$

as desired. □

Proof of Theorem 5. Analagous to the proof of Thm. 4. □

References

[ADM07] Russell G Almond, Louis V DiBello, Brad Moulder, and Juan-Diego Zapata-Rivera. "Modeling diagnostic assessments with Bayesian networks." *Journal of Educational Measurement*, **44**(4):341–359, 2007.

[AW05] Ivon Arroyo and B Woolf. "Inferring learning and attitudes from a Bayesian Network of log file data." In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, pp. 33–40, 2005.

[AY13] Sheeraz Ahmad and Angela J. Yu. "Active Sensing as Bayes-Optimal Sequential Decision Making." In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI-13)*, pp. 12–21, 2013.

[BAC10] Carole R Beal, I Arroyo, Paul R Cohen, Beverly P Woolf, and Carole R Beal. "Evaluation of AnimalWatch: An Intelligent Tutoring System for arithmetic and fractions." *Journal of Interactive Online Learning*, **9**(1):64–77, 2010.

[Bar12] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[BG11] Mustafa Bilgic and Lise Getoor. "Value of Information Lattice: Exploiting Probabilistic Independence for Effective Feature Subset Acquisition." *Journal of Artificial Intelligence Research (JAIR)*, **41**:69–95, 2011.

[BHM04] Cory J. Butz, Shan Hua, and R. Brien Maguire. "A Web-Based Intelligent Tutoring System for Computer Programming." In *Web Intelligence*, pp. 159–165. IEEE Computer Society, 2004.

[BL13] K. Bache and M. Lichman. "UCI Machine Learning Repository.", 2013.

[BM07] Peter Brusilovsky and Eva Millán. "User models for adaptive hypermedia and adaptive educational systems." In *The adaptive web*, pp. 3–53. Springer-Verlag, 2007.

[BSB13] Gowtham Bellala, Jason Stanley, Suresh K. Bhavnani, and Clayton Scott. "A Rank-Based Approach to Active Diagnosis." *IEEE Trans. Pattern Anal. Mach. Intell.*, **35**(9):2078–2090, 2013.

[CCD12] Suming Chen, Arthur Choi, and Adnan Darwiche. "The Same-Decision Probability: A New Tool for Decision Making." In *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models*, pp. 51–58, 2012.

[CCD13] Suming Chen, Arthur Choi, and Adnan Darwiche. "An Exact Algorithm for Computing the Same-Decision Probability." In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 2525–2531, 2013.

[CCD14] Suming Chen, Arthur Choi, and Adnan Darwiche. "Algorithms and Applications for the Same-Decision Probability." *Journal of Artificial Intelligence Research (JAIR)*, **49**:601–633, 2014.

[CCD15] Suming Chen, Arthur Choi, and Adnan Darwiche. "Value of Information Based on Decision Robustness." In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*, 2015.

[CCDew] Suming Chen, Arthur Choi, and Adnan Darwiche. "How Much Data Do We Need?" *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI)*, under review.

[CD03] Hei Chan and Adnan Darwiche. "Reasoning about Bayesian Network Classifiers." In *Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence*, pp. 107–115, 2003.

[CD05] Hei Chan and Adnan Darwiche. "A Distance Measure for Bounding Probabilistic Belief Change." *International Journal of Approximate Reasoning*, **38**:149–174, 2005.

[CGV02] Cristina Conati, Abigail Gertner, and Kurt VanLehn. "Using Bayesian networks to manage uncertainty in student modeling." *User Modeling and User-Adapted Interaction*, **12**(4):371–417, 2002.

[CLT12] Jie Chen, Kian Hsiang Low, Colin Keng-Yan Tan, Ali Oran, Patrick Jaillet, John Dolan, and Gaurav Sukhatme. "Decentralized Data Fusion and Active Sensing with Mobile Sensors for Modeling and Predicting Spatiotemporal Traffic Phenomena." In *Proceedings of the Twenty-Eighth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pp. 163–173, Corvallis, Oregon, 2012. AUAI Press.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[CV13] Konstantina Chrysafiadi and Maria Virvou. "PeRSIVA: An empirical evaluation method of a student model of an intelligent e-learning environment for computer programming." *Computers & Education*, **68**:322–333, 2013.

[CV14] Konstantina Chrysafiadi and Maria Virvou. "KEM Cs: A set of student's characteristics for modeling in adaptive programming tutoring systems." In *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on*, pp. 106–110. IEEE, 2014.

[CWM14] Brandi L Cantarel, Daniel Weaver, Nathan McNeill, Jianhua Zhang, Aaron J Mackey, and Justin Reese. "BAYSIC: a Bayesian method for combining sets of genome variants with improved specificity and sensitivity." *BMC Bioinformatics*, **15**(1):104, 2014.

[CXD12] Arthur Choi, Yexiang Xue, and Adnan Darwiche. "Same-Decision Probability: A Confidence Measure for Threshold-Based Decisions." *International Journal of Approximate Reasoning (IJAR)*, **2**:1415–1428, 2012.

[Dar09] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 1st edition, 2009.

[DC10] Adnan Darwiche and Arthur Choi. "Same-Decision Probability: A Confidence Measure for Threshold-Based Decisions under Noisy Sensors." In *Proceedings of the Fifth European Workshop on Probabilistic Graphical Models*, pp. 113–120, 2010.

[De 11] Cassio P De Campos. "New complexity results for MAP in Bayesian networks." In *Proceedings of the Twenty-Second IJCAI*, pp. 2100–2106. AAAI Press, 2011.

[Dec99] Rina Dechter. "Bucket elimination: A unifying framework for reasoning." *Artificial Intelligence*, **113**(1):41–85, 1999.

[DJ97] Soren Dittmer and Finn Jensen. "Myopic Value of Information in Influence Diagrams." In *Proceedings of the Thirteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 142–149, 1997.

[DP03] Chris Ding and Hanchuan Peng. "Minimum Redundancy Feature Selection from Microarray Gene Expression Data." In *Proceedings of the IEEE Computer Society Conference on Bioinformatics*, CSB '03, pp. 523–, Washington, DC, USA, 2003. IEEE Computer Society.

[FGG97] Nir Friedman, Dan Geiger, and Moises Goldszmidt. "Bayesian network classifiers." *Machine Learning*, **29**(2-3):131–163, 1997.

[GAS07] Patricio García, Analía Amandi, Silvia Schiaffino, and Marcelo Campo. "Evaluating Bayesian networks precision for detecting students learning styles." *Computers & Education*, **49**(3):794–808, 2007.

[GB11] Linda C. van der Gaag and Hans L. Bodlaender. "On Stopping Evidence Gathering for Diagnostic Bayesian Networks." In *ECSQARU*, pp. 170–181, 2011.

[GC99] Linda C. van der Gaag and Veerle M. H. Coupé. "Sensitivity Analysis for Threshold Decision Making with Bayesian Belief Networks." In *AI*IA*, pp. 37–48, 1999.

[GCV98] Abigail S Gertner, Cristina Conati, and Kurt VanLehn. "Procedural help in Andes: Generating hints using a Bayesian network student model." In *Proceedings of the 13th National Conference on Artificial intelligence*, pp. 106–111, 1998.

[GGR02] Russell Greiner, Adam J. Grove, and Dan Roth. "Learning cost-sensitive active classifiers." *Artificial Intelligence*, **139**(2):137–174, 2002.

[GH89] Paul Glasziou and Jrgen Hilden. "Test Selection Measures." *Medical Decision Making*, **9**(2):133–141, 1989.

[GK11a] T. Gao and D. Koller. "Active Classification based on Value of Classifier." In *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011.

[GK11b] Daniel Golovin and Andreas Krause. "Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization." *Journal of Artificial Intelligence Research (JAIR)*, **42**:427–486, 2011.

[GS12] Mayura V Gujarathi and Manojkumar S Sonawane. "Intelligent Tutoring System: A Case Study of Mobile Mentoring for Diabetes." *International Journal of Applied Information Systems (IJAIS)*, **3**(8):41–43, 2012.

[HBB10] Matthew D. Hoffman, David M. Blei, and Francis R. Bach. "Online Learning for Latent Dirichlet Allocation." In *Advances in Neural Information Processing Systems 23 (NIPS)*, pp. 856–864, 2010.

[HBR95] David Heckerman, John S. Breese, and Koos Rommelse. "Decision-theoretic troubleshooting." *Communications of the ACM*, **38**(3):49–57, March 1995.

[HCK92] Walter Hamscher, Luca Console, and Johan de Kleer, editors. *Readings in Model-Based Diagnosis*. Morgan Kaufmann Publishers Inc., 1992.

[Hen86] Max Henrion. "Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling." In *Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, pp. 149–163, 1986.

[HHM93] David Heckerman, Eric Horvitz, and Blackford Middleton. "An Approximate Nonmyopic Computation for Value of Information." *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, **15**(3):292–298, 1993.

[HM84] Ronald A. Howard and James E. Matheson, editors. *Readings on the Principles and Applications of Decision Analysis*. Strategic Decision Group, 1984.

[How66] Ronald A. Howard. "Information Value Theory." *IEEE Transactions on Systems Science and Cybernetics*, **2**(1):22–26, 1966.

[ICR04] Jaime S. Ide, Fabio G. Cozman, and Fabio T. Ramos. "Generating random Bayesian networks with constraints on induced width." In *Proceedings of the 16th European Conference on Artificial Intelligence*, pp. 323–327, 2004.

[Jor02] A Jordan. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes." *Advances in Neural Information Processing Systems*, **14**:841, 2002.

[Kar72] Richard M Karp. "Reducibility among combinatorial problems." In *Complexity of Computer Computations*. Springer, 1972.

[Ker12] Kristian Kersting. "Lifted Probabilistic Inference." In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, pp. 33–38, 2012.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[KG05] Andreas Krause and Carlos Guestrin. "Near-optimal Nonmyopic Value of Information in Graphical Models." In *21st Conference on Uncertainty in Artificial Intelligence*, pp. 324–331, 2005.

[KG09] Andreas Krause and Carlos Guestrin. "Optimal Value of Information in Graphical Models." *Journal of Artificial Intelligence Research (JAIR)*, **35**:557–591, 2009.

[KL97] Elizabeth Adele Klonoff and Hope Landrine. *Preventing Misdiagnosis of Women: A Guide to Physical Disorders That Have Psychiatric Symptoms*. SAGE Publications, Inc., 1997.

[KM08] Uffe B. Kjærulff and Anders L. Madsen. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer, 2008.

[KMR03] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. "Bayesian Event Classification for Intrusion Detection." In

Proceedings of the Annual Computer Security Applications Conference (ACSAC), 12 2003.

[Kor09] Richard Korf. "Multi-Way Number Partitioning." In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pp. 538–543, 2009.

[KPP04] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[KRS97] C E Kahn, L M Roberts, K A Shaffer, and P Haddawy. "Construction of a Bayesian network for mammographic diagnosis of breast cancer." *Computers in Biology and Medicine*, **27**(1):19–29, 1997.

[Kwi09] Johan Kwisthout. *The Computational Complexity of Probabilistic Networks*. PhD thesis, University of Utrecht, 2009.

[Lin56] Dennis V. Lindley. "On a measure of the information provided by an experiment." *Annals of Mathematical Statistics*, **27**(4):986–1005, 1956.

[LJ08] Wenhui Liao and Qiang Ji. "Efficient non-myopic value-of-information computation for influence diagrams." *International Journal of Approximate Reasoning*, **49**(2):436–450, 2008.

[LP06] Tsai-Ching Lu and K. Wojtek Przytula. "Focusing Strategies for Multiple Fault Diagnosis." In *Proceedings of the 19th International FLAIRS Conference*, pp. 842–847, 2006.

[MBL08] Gaspar Modelo-Howard, Saurabh Bagchi, and Guy Lebanon. "Determining Placement of Intrusion Detectors for a Distributed Application through Bayesian Network Modeling." In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, pp. 271–290, 2008.

[MDC13] Eva Millán, Luís Descalco, Gladys Castillo, Paula Oliveira, and Sandra Diogo. "Using Bayesian networks to improve knowledge assessment." *Computers & Education*, **60**(1):436–447, 2013.

[MP02] Eva Millán and José Luis Pérez-De-La-Cruz. "A Bayesian diagnostic algorithm for student modeling and its evaluation." *User Modeling and User-Adapted Interaction*, **12**(2-3):281–330, 2002.

[MS08] Michael Munie and Yoav Shoham. "Optimal testing of structured knowledge." In *Proceedings of the 23rd National Conference on Artificial intelligence*, pp. 1069–1074, 2008.

[Mur12] Kevin Patrick Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[NV14] Mathias Niepert and Guy Van den Broeck. "Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference." In *Proceedings of the 28th Conference on Artificial Intelligence (AAAI)*, pp. 2467–2475, 2014.

[OD13] Dimitri Ognibene and Yiannis Demiris. "Towards Active Event Recognition." In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 2495–2501, 2013.

[PD04] James D. Park and Adnan Darwiche. "Complexity Results and Approximation Strategies for MAP Explanations." *Journal of Artificial Intelligence Research (JAIR)*, **21**:101–133, 2004.

[Pis95] David Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, University of Copenhagen, 1995.

[PK80] Steven G. Pauker and Jerome P. Kassirer. "The threshold approach to clinical decision making." *The New England Journal of Medicine*, **302**(20):1109–17, 1980.

[Poo03] David Poole. "First-order probabilistic inference." In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 985–991, 2003.

[Rai68] H. Raiffa. *Decision Analysis – Introductory Lectures on Choices under Uncertainty*. Addison-Wesley, 1968.

[RIM13] R. Rajendran, S. Iyer, S. Murthy, C. Wilson, and J. Sheard. "A Theory-Driven Approach to Predict Frustration in an ITS." *Learning Technologies, IEEE Transactions on*, **6**(4):378–388, Oct 2013.

[Rot96] Dan Roth. "On the hardness of approximate reasoning." *Artificial Intelligence*, **82**:273 – 302, 1996.

[RS01] Marco Ramoni and Paola Sebastiani. "Robust Bayes classifiers." *Artificial Intelligence*, **125**(1-2):209–226, 2001.

[SH06] Siriwan Suebnukarn and Peter Haddawy. "A Bayesian approach to generating tutorial hints in a collaborative medical problem-based learning system." *Artificial intelligence in Medicine*, **38**(1):5–24, 2006.

[Shi94] Solomon Eyal Shimony. "Finding MAPs for Belief Networks is NP-Hard." *Artificial Intelligence*, **68**(2):399–410, 1994.

[Sin06] Sandip Sinharay. "Model diagnostics for Bayesian networks." *Journal of Educational and Behavioral Statistics*, **31**(1):1–33, 2006.

[SS13] Mike Shann and Sven Seuken. "An Active Learning Approach to Home Heating in the Smart Grid." In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 2892–2899, 2013.

[Str65] Ruslan Stratonovich. "On value of information." *Izvestiya of USSR Academy of Sciences, Technical Cybernetics*, **5**:3–12, 1965.

[TK00] Simon Tong and Daphne Koller. "Active Learning for Parameter Estimation in Bayesian Networks." In *Advances in Neural Information Processing Systems 13 (NIPS)*, pp. 647–653, 2000.

[TK01] Simon Tong and Daphne Koller. "Active Learning for Structure in Bayesian Networks." In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 863–869, 2001.

[VC14] Mieke Vandewaetere and Geraldine Clarebout. "Advanced Technologies for Personalized Learning, Instruction, and Performance." In *Handbook of Research on Educational Communications and Technology*, pp. 425–437. Springer, 2014.

[VN01] Kurt VanLehn and Zhendong Niu. "Bayesian student modeling, user interfaces and feedback: A sensitivity analysis." *International Journal of Artificial Intelligence in Education*, **12**(2):154–184, 2001.

[Vom04] Jiri Vomlel. "Bayesian networks in educational testing." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **12**(supp01):83–100, 2004.

[Was11] Larry Wasserman. *All of Statistics*. Springer Science & Business Media, 2011.

[WRM09] Patti West, DW Rutstein, Robert J Mislevy, Junhui Liu, Roy Levy, KE DiCerbo, A Crawford, Y Choi, and J Behrens. "A Bayes net approach to modeling learning progressions and task performances." In *Learning Progressions in Science conference, Iowa City, IA*, 2009.

[Xen04] Michalis Xenos. "Prediction and assessment of student behaviour in open and distance education in computers using Bayesian networks." *Computers & Education*, **43**(4):345–359, 2004.

[YKR09] Shipeng Yu, Balaji Krishnapuram, Romer Rosales, and R Bharat Rao. "Active sensing." In *International Conference on Artificial Intelligence and Statistics*, pp. 639–646, 2009.

[ZBB10] Günther Zäpfel, Roland Braune, and Michael Bögl. *Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics*. Springer-Verlag Berlin Heidelberg, 2010.

[Zha98] Nevin L. Zhang. "Probabilistic Inference in Influence Diagrams." In *Computational Intelligence*, pp. 514–522, 1998.

[ZJ10] Yongmian Zhang and Qiang Ji. "Efficient sensor selection for active information fusion." *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, **40**(3):719–728, June 2010.