"The Bayesian revolution in the sciences is fueled, not only by more and more cognitive scientists suddenly noticing that mental phenomena have Bayesian structure in them; not only by scientists in every field learning to judge their statistical methods by comparison with the Bayesian method; but also by the idea that science itself is a special case of Bayes' Theorem; experimental evidence is Bayesian evidence."

– Eliezer S. Yudkowsky.

University of Alberta

Budgeted Parameter Learning of Generative Bayesian Networks

by

Liuyang Li

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2009
Edmonton, Alberta

Examining Committee

Russ Greiner, Department of Computing Science

Csaba Szepesvari, Department of Computing Science

Peter Hooper, Department of Mathematical and Statistical Sciences

To My Parents and Elaine.

Abstract

This dissertation studies the parameter estimation problem of Bayesian networks in the budgeted learning setting. More precisely, we assume that the correct structure of the Bayesian network representing the underlying distribution is given together with a fixed positive budget, and each data attribute of the training set is associated with a cost. During the training phase, the learner is allowed to purchase value of an attribute of a certain data instance by deducting the corresponding cost from the budget. The goal of the learner is to make the purchases wisely so that when the budget is exhausted, the learned parameters from the purchased data are as close as possible to the underlying distribution that generates the data. The dissertation presents a theoretical framework for the problem, analyzes its hardness, and compares different algorithms and heuristics for solving the problem efficiently and economically.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Russ Greiner. Russ initiated, encouraged, and funded my research during my masters study. He introduced me to the Bayesian networks and collaborated with me on some of the results contained in the following pages. His creative ideas, wise advice, patient explanations and humor makes mathmatics and machine learning fun to me. I also wish to thank him for the enjoyable time that I had in the summer parties.

I am grateful to Dr. Sergey Krishner for his great insights during our discussions, and his generous suggestions to both my research and my career.

Table of Contents

List of Figures

List of Symbols

X, Y, Z	Random variables
$\|X\|$	Number of possible values for X
$\boldsymbol{X} = \{X_1, ..., X_n\}$	Set of random variables
N	Number of variables in a Bayesian network
M	Number of data instances
$\boldsymbol{U}_i$	Parents of node X_i
$\boldsymbol{u}_i$	An instantiation of parents of node X_i
$M_{x_i\|\boldsymbol{u}_i}$	Number of data instances where $X_i = x_i$ and $\boldsymbol{U}_i = \boldsymbol{u}_i$
K_i	Number of possible values for a variable X_i
x_i	Some instantiation of random variable X_i
x^k	Instantiation of random variable with the kth value
$P(X_1, ..., X_n)$	Probability distribution over random variables $X_1, ..., X_n$
$P(x_1, ..., x_n)$	Probability of random variables $X_1, ..., X_n$ taking values $x_1, ..., x_n$ shorthand for $P(X_1 = x_1, ..., X_n = x_n)$
$E[X]$	Expectation of X
$\boldsymbol{\theta}$	Bayesian network parameters
$\hat{\boldsymbol{\theta}}$	Point estimate of Bayesian network parameters
$\boldsymbol{\theta}^i$	Bayesian network parameters in the ith iteration of the EM algorithm
$\boldsymbol{\theta}_i$	Parameters of node X_i
$\theta_{x_i\|\boldsymbol{u}_i}$	Bayesian network parameter; shorthand for $P(x_i\|\boldsymbol{u}_i)$
$\mathcal{G}$	Bayesian network structure (DAG)
$\mathbb{R}_{\geq 0}$	Nonnegative real numbers
$\mathbb{Z}_{>0}$	Positive integers
$\mathcal{D} = \{\boldsymbol{d}^1, ..., \boldsymbol{d}^M\}$	Data set with M data instances
$\boldsymbol{d}^i = \{d_1^i, ..., d_N^i\}$	Data instance i with N attributes
$\alpha, \beta, \gamma, \delta \in \mathbb{R}_{>0}$	Dirichlet hyperparameter
$\ln(\cdot)$	Base e logarithm; shorthand for $\log_e(\cdot)$
$\Gamma(\cdot)$	Gamma function
$\Psi(\cdot)$	Digamma function $\Psi(\cdot) = \frac{\Gamma'(\cdot)}{\Gamma(\cdot)}$
$H(\cdot)$	When applied to a probability $P(x)$: Entropy function $H(P(x)) = -P(x) \ln P(x)$ When applied to a discrete distribution $P(X)$: Entropy $H(P(X)) = \sum_x -P(x) \ln P(x)$
$KL(\boldsymbol{\theta}\|\|\hat{\boldsymbol{\theta}})$	Kullback Leibler (KL)-divergence $\sum_x P_{\boldsymbol{\theta}}(x) \ln \frac{P_{\boldsymbol{\theta}}(x)}{P_{\hat{\boldsymbol{\theta}}}(x)}$
$\ell(\cdot)$	Log-likelihood computed by taking the base e logarithm of the likelihood; shorthand for $\ln P(\cdot)$
$\mathcal{B}$	Predetermined budget
$c(X_i)$	Cost of probing variable X_i
ε	an arbitrarily small positive quantity

Chapter 1

Introduction

A Bayesian network is a compact and natural graphical representation of a multivariate probabilistic distribution, which encodes all the dependencies of the underlying distribution using a directed acyclic graph (DAG). There have been many successful applications of Bayesian networks to practical problems, *e.g.* a large expert system for the diagnosis of lymph-node pathology and troubleshooting systems for printing, photocopier feeders, automobiles, and gas turbines [16, 4]. Given the considerable cost and limited accuracy of the engineered Bayesian network by experts (it is particularly difficult for the expert to pinpoint the exact parameters of the Bayesian network [35]), inducing Bayesian network parameters and structure from given data has been studied extensively [17]. In this paper, we study the challenge of estimating Bayesian network parameters when the learner is initially given no data but instead allowed to purchase certain data attributes to help construct an accurate model. We assume that each data attribute takes a finite number of discrete values in the domain of the corresponding variable, and is associated with a cost; and a fixed budgeted is given. By deducting the cost of a data attribute for a certain instance from the budget, the learner is able to observe the value of the variable, which we call a probe. When the budget is exhausted, the leaner has to learn the parameters of the Bayesian network using the observed data. The goal of the budgeted learner is to make the probes wisely so that the learned generative model is as close to the underlying distribution as possible. We use the term *budgeted learning* to refer to this approach [23].

The budgeted learning task emerges naturally in many real life problems. Consider building a medical diagnosis Bayesian network starting with partial or no data. The network structure composed of symptom and disease diagnosis variables and their dependencies, together with the costs for probing different variables, is given by experts. A budget is provided for us to purchase data.

Our task is to make the selective purchases so that we can make a good estimate of the parameters of the given Bayesian network modeling the underlying joint distribution of the various symptoms and diseases using the purchased data, subject to the budget. Data can be costly in terms of different measurements, such as time, energy, and *etc*, but based on the utility theory [10], we assume that all these measurements can be converted to a single value for each variable in the distribution. We choose KL-divergence as our loss function, because it is efficient to compute the KL-divergence

between distributions over the same Bayesian network structure, and a small KL-divergence implies a small L_1 norm, which properly models the practical problems.

Similar problem exists in other domains, which involve making decision under uncertainty. This thesis investigates the budgeted parameter learning problem in Bayesian networks. The thesis statement of this dissertation is: well-defined allocation algorithms, though suboptimal, solves the budgeted parameter learning problem efficiently. We formally formulate the problem, discuss different loss functions, and study different policies for addressing the problem. We show that the budgeted learning problem under our framework is NP-hard even when all the variables are independent from each other. For the budgeted learning problem with uniform costs, we show theoretically that for a network with all independent nodes and uniform costs, an iterative greedy allocation algorithm is an optimal allocation algorithm, whereas for a complete network with certain priors constraints, Round-Robin might be preferred.

The rest of the dissertation is organized as follows: Chapter 2 presents the background knowledge for Bayesian network and its parameter learning; Chapter 3 studies the budgeted parameter learning problem in Bayesian networks with all independent nodes, gives the hardness result and optimality proof of the iterative greedy allocation algorithm, and presents results of our empirical studies. Chapter 4 discusses the budgeted parameter learning problem in general Bayesian networks. Chapter 5 gives a brief literature review of the related work and discusses their distinctions from our research. Chapter 6 concludes the dissertation.

Chapter 2

Background

2.1 Bayesian Networks

2.1.1 Introduction

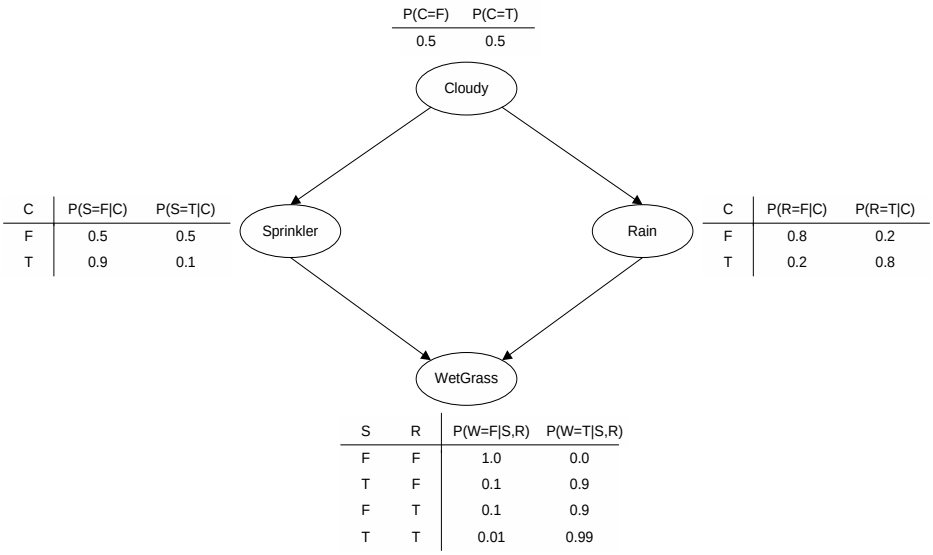


Figure 2.1: *Sprinkler* Bayesian network [31] modeling the "Cloudy-WetGrass" domain. Variables are denoted by ellipses with their names at the centers. CPTs are denoted by tables. Dependencies are indicated by directed arcs.

A Bayesian network is a graphical probabilistic representation of the domain of interest. It has gained significant popularity in recent decades due to its conciseness and naturalness. It is concise because it models dependencies of the underlying joint distribution and this modeling scheme avoids the construction of the whole joint distribution table, which is exponential in the number of variables. Moreover, since all the dependencies and causal relationships of the underlying distribution are explicitly modeled by the Bayesian network, it is also a natural representation. In this chapter, we review some of the fundamentals of Bayesian networks including its representation, and standard

parameter learning.

A Bayesian network is composed of a DAG and local probability models for each node in the graph structure. Figure 2.1 gives a sample Sprinkler Bayesian network from Russell and Norvig [31]. "Cloudy" (C) denotes whether the weather is cloudy; "Sprinkler" (S) denotes whether the sprinkler in the yard is on; "Rain" (R) denotes whether it rained; "WetGrass" (W) denotes whether grass in the yard is wet. For simplicity, all the variables in the network are binary. The structure encodes the dependencies of the underlying distribution. Each node in the Bayesian network is associated with a local conditional distribution given its parents. For instance, in the Sprinkler network, the probability of WetGrass depends on the values of Sprinkler and Rain. In general, the conditional probability distribution (CPD) of the child given its parents is encoded by a conditional probability table (CPT). The DAG, together with all the local conditional distributions, models the underlying joint distribution compactly and naturally. If we have to represent the full joint probability distribution of N [1] binary variables, $O(2^N)$ space is required. However, the factored Bayesian network representation requires only $O(N2^m)$ space, where m is maximum number of parents for a node. Also, the structure of the network properly encodes the fact that each node is independent from its non-descendants knowing the values of its parents. For example, through investigating Figure 2.1, it is easy to retrieve and express the probability of a rain happening given that it is a cloudy day. Finally, by multiplying the relevant entries of each CPT, it is not hard to reconstruct the full joint probability table.

For example, the full joint distribution $P(C, S, R, W)$ can *always* be represented as the following using the chain rule of probability:

$$P(C, S, R, W) = P(C)P(S|C)P(R|S, C)P(W|C, S, R). \tag{2.1}$$

By applying the conditional independencies encoded in the Bayesian network, Equation 2.1 can be simplified to:

$$P(C, S, R, W) = P(C)P(S|C)P(R|C)P(W|S, R). \tag{2.2}$$

In general,

$$P(X_1, ..., X_n) = \prod_i P(X_i|\boldsymbol{U}_i), \tag{2.3}$$

where $P(X_1, ..., X_n)$ denotes the probability distribution of random variables $X_1, ..., X_n$, and $\boldsymbol{U}_i$ denotes the set of parents of variable X_i.

2.2 Parameter Estimation

In this section, we discuss the standard ways to estimate parameters of Bayesian networks. For *standard parameter estimation*, we are given a Bayesian network structure $\mathcal{G}$ that contains N nodes,

[1] N denotes number of variables, M denotes number of data instances, and $K_i = |X_i|$ denotes number of possible values of the variable X_i.

together with a data set $\mathcal{D} = \{\boldsymbol{d}^1, ..., \boldsymbol{d}^M\}$ of M data instances, where each data instance, also called a data tuple, is characterized by N attributes $\boldsymbol{d}^j = \{d_1^j, ..., d_N^j\}$, which correspond to values of the N variables. If the value of the ith attribute of the jth instance is missing, d_i^j is denoted as a question mark "?"; otherwise, d_i^j takes a value from the domain of variable X_i. In the paper, we also assume that all the variables are discrete. Recall that parameters refer to the function potentials associated with each node in the graph structure. The goal of standard Bayesian network parameter estimation algorithms is to learn the parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_N\}$ [2] given $\mathcal{G}$ from $\mathcal{D}$.

Bayesian network parameters do not have to be learned in a Bayesian way; any method, if applicable, can be used to deal with the parameter learning task. The *Maximum likelihood estimation* (MLE) is the frequentist approach to learning the parameters of a Bayesian network, while the *maximum a posterior* (MAP) is a Bayesian counterpart. Therefore, the name *Bayesian* network is slightly incongruous. In this dissertation, because we shall take the Bayesian approach, we focus our attention on the MAP learning.

2.2.1 The Beta Distribution

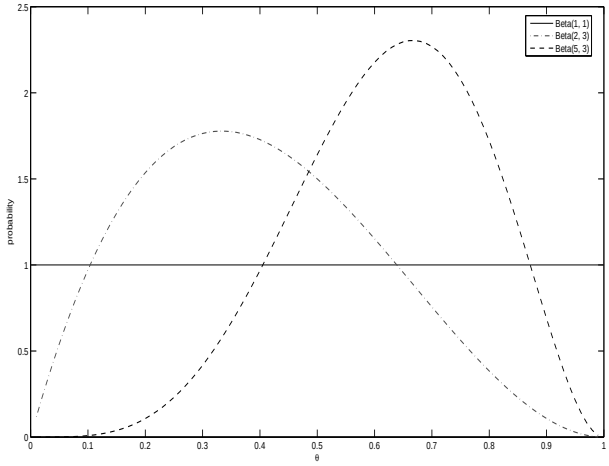


Figure 2.2: Probability density function of beta distributions: Beta(1, 1), Beta(2, 3), and Beta(5, 3).

For the MAP learning, the beta distribution is commonly chosen for modeling parameters of Bayesian networks with discrete variables. It is popular because: (1) it is a continuous distribution defined on the interval $[0, 1]$, which makes it a good choice for modeling probabilities; and (2) it is conjugate for binomial signals, which makes the computation of the posterior efficient. The beta distribution is a two parameter distribution. The probability density function of the beta distribution

[2] We use bold $\boldsymbol{\theta}$s to denote sets of parameters. For example, here $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1 ... \boldsymbol{\theta}_N\}$ denotes all the parameters for all the variables in a Bayesian network, where $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{X_i}$ denotes the set of parameters for variable X_i.

is in the form of

$$p(\boldsymbol{\theta}) = Beta(\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \tag{2.4}$$

where $Beta$ is a distribution of $\boldsymbol{\theta}$, α and β are the two parameters for $Beta$ and can take any positive real values, and the Gamma function Γ is defined as:

$$\Gamma(z) = \int_0^{+\infty} t^{z-1}e^{-t}dt. \tag{2.5}$$

for all positive real numbers $z \in \mathbb{R}_{>0}$. Figure 2.2 plots the probability density function of 3 beta distributions.

The beta distribution is well defined for positive real numbers, but in this paper, the two parameters are positive integers ($\alpha, \beta \in \mathbb{Z}_{>0}$) most of the time. For positive integer $\alpha \in \mathbb{Z}_{>0}$, the Gamma function can be simplified to:

$$\Gamma(\alpha) = (\alpha-1)!. \tag{2.6}$$

One property of the Gamma function that we use extensively in our derivations is,

$$\Gamma(\alpha+1) = \alpha\Gamma(\alpha). \tag{2.7}$$

The multivariate generalization of the beta distribution is the Dirichlet distribution. The probability density function is given by

$$p(\theta_{x^1}, ...\theta_{x^K}) = Dirichlet(\alpha_{x^1}, ...\alpha_{x^K}) = \frac{\Gamma(\sum_{k=1}^K \alpha_{x^k})}{\prod_{k=1}^K \Gamma(\alpha_{x^k})}\prod_{k=1}^K \theta_{x^k}^{\alpha_{x^k}-1} \tag{2.8}$$

Each of the $\alpha_{x^k} \in \mathbb{R}_{>0}$ is a hyper-parameter for the variable X with the kth value; Γ is the gamma function; $\theta_{x^k} > 0$, and $\sum \theta_{x^k} = 1$.

In most of the examples and discussions, we will use beta distributions to illustrate our ideas, which can be easily extended to multivariate Dirichlet distributions.

2.2.2 Complete Data

The MAP learning starts with a prior knowledge of the model, which is updated as new data are collected. Because we assume that the correct structure is given, we only need to have prior distribution for the parameters in the target network. The goal of the standard MAP learning is to compute the posterior probability distribution of the parameters.

For a given structure $\mathcal{G}$ and data set $\mathcal{D}$, the posterior distribution of the parameter can be computed as:

$$P(\boldsymbol{\theta}|\mathcal{G}, \mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta}, \mathcal{G})P(\boldsymbol{\theta}|\mathcal{G})}{P(\mathcal{D}|\mathcal{G})} \tag{2.9}$$

The computation is also called an *update* to the prior using the data.

With the parameter independence assumption [17], $p(\boldsymbol{\theta}) = \prod_i \prod_{\boldsymbol{u}_i} p(\boldsymbol{\theta}_{X_i|\boldsymbol{u}_i})$, we can decompose the prior $p(\boldsymbol{\theta})$ into independent Dirichlet distributions [3]. Standard results [19] show that it is easy to update conjugate Dirichlet priors:

[3] $\boldsymbol{u}_i$ denotes an instantiation of the parents of X_i.

Theorem 2.2.1. *If* $p(\theta_{x^1}, ...\theta_{x^K}) = Dirichlet(\alpha_{x^1}, ...\alpha_{x^K})$ *then* $p(\theta_{x^1}, ...\theta_{x^K}|\mathcal{D}) = Dirichlet(\alpha_{x^1} + M_{x^1}, ...\alpha_{x^K} + M_{x^K})$*, where* M_{x^k} *is the number of occurrences of* x^k *in the dataset* $\mathcal{D}$ [4].

To make predictions or inferences, we employ the *prediction distribution* of the model. Based on our data and model, suppose we wish to predict the next observation, d^{M+1}. This computation is achievable by integrating out the uncertainty in the model parameters:

$$P(d^{M+1}|\mathcal{G}, \mathcal{D}) = \int_{\boldsymbol{\theta}} P(d^{M+1}|\boldsymbol{\theta}, \mathcal{G}, \mathcal{D}) P(\boldsymbol{\theta}|\mathcal{G}, \mathcal{D}) d\boldsymbol{\theta} \tag{2.10}$$

This is called the Bayesian one-step prediction. The bad news is that in general, it is hard to compute the integration exactly and efficiently, although various approximation techniques exist. Luckily, Dirichlet distributed priors over parameters for discrete variables is one exception. In the rest of this section, we shall discuss how we can incorporate Dirichlet distributions in our parameter estimation task. It is also not hard to compute the Bayesian one-step prediction, which is often used as the point estimate of the distribution when doing inferences:

$$P(d^1 = x^k) = E[\theta_{x^k}] = \frac{\alpha_{x^k}}{\sum_{j=1}^{K} \alpha_{x^j}}, \tag{2.11}$$

and

$$P(d^{M+1} = x^k|\mathcal{D}) = E[\theta_{x^k}|\mathcal{D}] = \frac{\alpha'_{x^k}}{\sum_{k=j}^{K} \alpha'_{x^j}} = \frac{M_{x^k} + \alpha_{x^k}}{\sum_{k=j}^{K} (M_{x^j} + \alpha_{x^j})}. \tag{2.12}$$

We call the summations of all hyper-parameters of each Dirichlet distribution $\sum_{j=1}^{K} \alpha_{x^j}$ and $\sum_{j=1}^{K} \alpha'_{x^j}$ the effective sample sizes, and the counts M_{x^k}s, the sufficient statistics for the sample. Intuitively, the hyper-parameter α_{x^k} is the imaginary counts (pseudo-counts) and M_{x^k} is the number of the instances observed of the kth value of the variable.

Figure 2.3 shows an example of the whole MAP parameter learning procedure. We collect the data, use them to update our prior knowledge to produce the posterior, then use the point estimate for future decision making tasks. We may want to keep the posterior and take it as the prior for further learning tasks.

To set the priors for the parameters, we need to choose the values of all the hyper-parameters of the Dirichlets. Uniform and BDe (Bayesian Dirichlet likelihood equivalent) [17] are the two popular assignment for the Dirichlet priors over parameters. In the example, we use uniform Dirichlet priors, *e.g.* $P(C) \sim Beta(1, 1)$. It can be seen from the flat line in Figure 2.2 that uniform Dirichlet priors are uniform distributions of parameter values in the range of $[0, 1]$. This means all values are equally likely, which is why technically it is also called non-informative priors. For our problem, the BDe constraint basically says that effective sample sizes of the priors for all variables should be the same. To set BDe priors for a Bayesian network, we start with an effective sample size M_0 and a prior joint distribution J_0, and set each $\alpha_{x_i|\boldsymbol{u}_i} = M_0 \times P(x_i, \boldsymbol{u}_i|J_0)$.

[4] x^k denotes the instantiation of the random variable X with the kth value

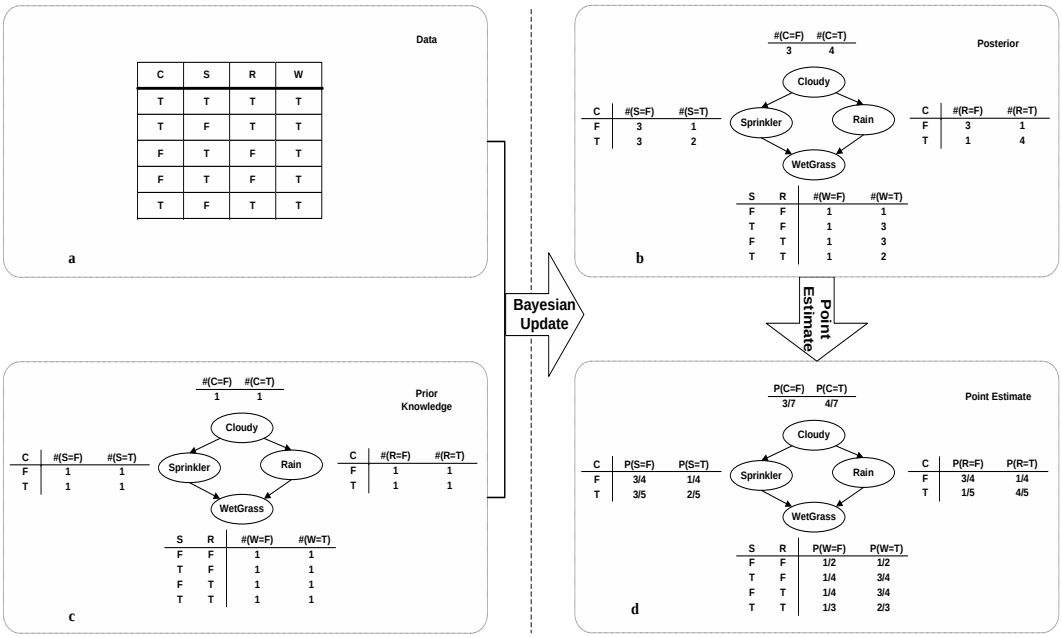


Figure 2.3: MAP parameter learning without missing data. In (a), data generated from the underlying distribution (the true Sprinkler network) is shown in a table. Each row of the table corresponds to a data instance d^j, while each column corresponds to the values of a data attribute d_i. All the variables are binary, with T denoting true and F denoting false. Below the data is our prior knowledge (b), which is composed of the correct structure and uniform priors. We use $\#(\cdot)$ to denote the fictitious counts (hyper-parameters) *e.g.*, $\#(C = T) = 1$ for the prior. The data, together with the prior, give the posterior estimate (c) of the network parameters after the standard Bayesian update. By taking the mean values of the posterior distributions, we get the point estimate (d).

The BDe constraint is not satisfied in our example because in the CPT of Cloudy $\boldsymbol{\theta}_{Cloudy} \sim Beta(1,1)$ corresponds to the claim that Cloudy has been true and false one time each, whereas in the CPT of Sprinkler $\boldsymbol{\theta}_{Sprinkler|Cloudy=T} \sim Beta(1,1)$ corresponding to Cloudy being true twice and $\boldsymbol{\theta}_{Sprinkler|Cloudy=F} \sim Beta(1,1)$ corresponding to Cloudy being false twice. To set a BDe prior for these three nodes, we need to change the prior of Cloudy to $\boldsymbol{\theta}_{Cloudy} \sim Beta(2,2)$ [5]. Here the prior for Cloudy is not uniform anymore, which results in an informative prior. Generally speaking, uniform priors are not BDe priors, but the more data we get, the less pronounced the difference of the point estimate of the posterior is.

2.2.3 With Missing Data

We have to use more sophisticated algorithms to compute the posteriors with the missing data. This is because the posterior distribution of the parameters are not independent from each other any more. In fact, no simple update rule is available. See Section 5.3 for a brief review of popular algorithms to learn parameters of Bayesian networks with missing data, and Section 4.1.3 for a discussion of the (in)applicability of the Expectation-Maximization (EM) and other imputation algorithms to the

[5] Note that the other parts of the network still do not satisfy the BDe constraint.

budgeted parameter learning problem.

2.3 KL-Divergence and Expected KL-Divergence

This section defines the loss function that we use to measure the learner's final performance, and the risk function for posterior distributions.

2.3.1 KL-Divergence as the Loss Function

To evaluate the parameter estimate, we choose the *Kullback-Leibler (KL)-divergence* [7] as our loss function. The KL-divergence is defined as

$$\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}) = \sum_x P_{\boldsymbol{\theta}}(x) \ln \frac{P_{\boldsymbol{\theta}}(x)}{P_{\hat{\boldsymbol{\theta}}}(x)}, \tag{2.13}$$

where $\boldsymbol{\theta}$ is the set of true parameters, and $\hat{\boldsymbol{\theta}}$ is our estimate [6].

The KL-divergence measures the difference between $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$. It can be interpreted as the additional bits used when encoding events from $\boldsymbol{\theta}$ with a code based on $\hat{\boldsymbol{\theta}}$. Actually, it is not hard to show that $\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}) \geq 0$, and $\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}) = 0$ if and only if $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. As the KL-divergence is not commutative and does not satisfy the triangle equality, it is not a metric.

Another common used loss function, *log-likelihood* $\ell(\boldsymbol{\theta}) = \sum_x P_{\boldsymbol{\theta}}(x) \ln P_{\hat{\boldsymbol{\theta}}}(x)$, is highly related to the KL-divergence.

$$\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}) = -\sum_x P_{\boldsymbol{\theta}}(x) \ln P_{\hat{\boldsymbol{\theta}}}(x) - \left(-\sum_x P_{\boldsymbol{\theta}}(x) \ln P_{\boldsymbol{\theta}}(x) \right). \tag{2.14}$$

We see that maximizing log-likelihood of $\hat{\boldsymbol{\theta}}$ corresponds to minimizing the KL-divergence, as $P_{\boldsymbol{\theta}}(x)$ does not depend on $\hat{\boldsymbol{\theta}}$.

The KL-divergence decomposes over Bayesian networks with the parameter independence assumption, which makes the computation efficient. It is also a meaningful loss function to choose because a small KL-divergence implies a small L_1-norm [7].

The mean value of the posterior distribution, which we used as the point estimate in the previous section, can be easily proven to be the one that minimizes the expected KL-divergence for the posterior distribution [34]:

$$\hat{\boldsymbol{\theta}} = E[\boldsymbol{\theta}] = \arg\min_{\tilde{\boldsymbol{\theta}}} \int_{\boldsymbol{\theta}} \mathrm{KL}(\boldsymbol{\theta}||\tilde{\boldsymbol{\theta}}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{2.16}$$

[6] Here, $0 \ln \frac{0}{P_{\hat{\boldsymbol{\theta}}}(X)} = 0$, and $P_{\boldsymbol{\theta}}(X) \ln \frac{P_{\boldsymbol{\theta}}(X)}{0} = \infty$. Because we take a Bayesian approach in this paper, the priors serve as a smoother, and we will never have zero probabilities in our estimate. Thus, the second case does not happen.

[7] For example, one can show that [1]

$$||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||_1 = \int_X |\theta_x - \hat{\theta}_x| dx \leq 2\sqrt{1 - \exp(-\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}))} \leq \sqrt{2\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}})}. \tag{2.15}$$

Thus, from now on we will use the $\hat{\boldsymbol{\theta}}$ to denote the mean of a Dirichlet distribution. That is for $\boldsymbol{\theta}_X \sim Dirichlet(\alpha_{x^1}, ...\alpha_{x^K})$:

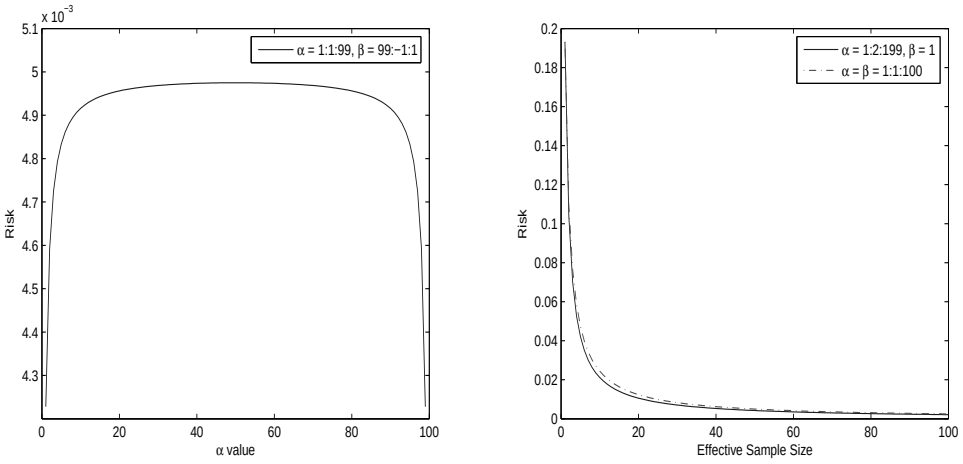
$$\hat{\theta}_{x^k} = E[\theta_{x^k}] = \frac{\alpha_{x^k}}{\sum_j \alpha_{x^j}}. \tag{2.17}$$

2.3.2 Expected KL-divergence as the Risk Function

As the KL-divergence is chosen as the loss function, we use the expected KL-divergence to the mean value (point estimate) of the distribution as the risk. $p_{\boldsymbol{\theta}}$ is a distribution over a set of parameter $\boldsymbol{\theta}$, but because the risk function does not depend on $\boldsymbol{\theta}$, we write it as $p_{\boldsymbol{\theta}}$ instead of $p(\boldsymbol{\theta})$.

$$Risk(p_{\boldsymbol{\theta}}) = E[\mathrm{KL}(p_{\boldsymbol{\theta}})] = \int_{\boldsymbol{\theta}} \mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}, \tag{2.18}$$

where $\hat{\boldsymbol{\theta}}$ is the mean value of the parameter distribution $p(\boldsymbol{\theta})$.



(a) $Risk(Beta(\alpha, \beta))$, where $\alpha + \beta = 100$.

(b) $Risk(Beta(\alpha, \beta))$, where $\alpha + \beta$ ranges from 2 to 200.

Figure 2.4: Properties of the risk function defined as expected KL-divergence to the mean value.

For a beta distribution, the risk is a function of the hyper-parameters α and β. The full formula of $Risk(Beta(\alpha, \beta))$ is given in the next chapter. Figure 2.4(a) plots the risks of beta distributions $Beta(\alpha, \beta)$ that have the same effective sample size $\alpha + \beta = 100$, where α ranges from 1 to 99, and β ranges from 99 to 1; Figure 2.4(b) plots the risks of beta distributions that have different effective sample sizes ranging from 2 to 200 with a step of 2, where for the dashed line $\alpha = \beta$ and for the solid line $\beta = 1$ and α ranges from 1 to 199 with a step of 2. Figure 2.4 shows that the risk function correctly captures the intuition of the sureness of the beta distributions (refer to Figure 2.2): first, with the same effective sample size (when $\alpha + \beta$ is a constant), more skewed beta distributions whose $|\alpha - \beta|$ values are larger have a smaller risk, because their variances are smaller (see Lemma 3.3.6); second, the more data sample we have (the larger $\alpha + \beta$ hence sharper beta distributions), the smaller is the risk. It is also worth noticing that, compared to the skewness of the distribution, the risk function is more sensitive to the effective sample sizes.

Chapter 3

Bayesian Network with All Independent Nodes

3.1 Problem Formulation

Before moving on to the general budgeted parameter estimation of Bayesian network problem, we first study a simplified problem where the variables in the Bayesian network are independent from each other, to derive several theoretical results and gain some insights.

Definition 3.1.1. [Independent Variable Problem] We are given a collection of N variables $\boldsymbol{X} = \{X_1, ..., X_N\}$, where the ith variable X_i has a cost $c_i = c(X_i)$ and a prior distribution $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}_i)$. All the variables are independent from each other. A fixed total budget is given by $\mathcal{B} \in \mathbb{R}_{>0}$. Each time the learner probes the variable X_i, a value in the domain of X_i is observed and a cost of c_i is spent. The job of the learner is to determine a sequence of probes, so that once the budgeted is exhausted, the joint distribution of all variables estimated by the learner based on the outcomes of all the probes $\hat{\boldsymbol{\theta}}$ gives the least KL-divergence $\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}})$ from the underlying true distribution $\boldsymbol{\theta}$.

3.1.1 Loss Function and Risk Function

In this section, we apply the loss function and risk function, which we defined in the previous chapter, to the independent variable problem and show that this leads to some simplifications.

Loss Function

A loss function is decomposable if it can be written into a weighted summation of loss function of every variable, where the weights may be dependent on the other variables; a loss function for the a problem is strongly decomposable if it can be written into a summation of the loss functions of every variable. As mentioned before, the advantage of the KL-divergence is that it decomposes over Bayesian networks, with the parameter independence assumption [17]. The decomposition property makes the computation of KL-divergence efficient. For the independent variable problem, we can show that the KL-divergence between the estimated and the true parameters for all the variables is

the sum of that for each variable. Using the fact that all the variables are independent from each other,

$$\begin{aligned}
\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}) &= \mathrm{KL}(\boldsymbol{\theta}_1 \times \boldsymbol{\theta}_2 \times ... \times \boldsymbol{\theta}_N||\hat{\boldsymbol{\theta}}_1 \times \hat{\boldsymbol{\theta}}_2 \times ... \times \hat{\boldsymbol{\theta}}_N) \\
&= \sum_{k_1=1}^{K_1}\sum_{k_2=1}^{K_2}...\sum_{k_N=1}^{K_N} \theta_{x_1^{k_1}}\theta_{x_2^{k_2}}...\theta_{x_N^{k_N}} \ln \frac{\theta_{x_1^{k_1}}\theta_{x_2^{k_2}}...\theta_{x_N^{k_N}}}{\hat{\theta}_{x_1^{k_1}}\hat{\theta}_{x_2^{k_2}}...\hat{\theta}_{x_N^{k_N}}} \\
&= \sum_{k_1=1}^{K_1}\sum_{k_2=1}^{K_2}...\sum_{k_N=1}^{K_N} \theta_{x_1^{k_1}}\theta_{x_2^{k_2}}...\theta_{x_N^{k_N}} (\ln \frac{\theta_{x_1^{k_1}}}{\hat{\theta}_{x_1^{k_1}}} + \ln \frac{\theta_{x_2^{k_2}}}{\hat{\theta}_{x_2^{k_2}}} + ... + \ln \frac{\theta_{x_N^{k_N}}}{\hat{\theta}_{x_N^{k_N}}}) \\
&= \sum_{k_1=1}^{K_1}\left(\sum_{k_2=1}^{K_2}...\sum_{k_N=1}^{K_N} \theta_{x_2^{k_2}}...\theta_{x_N^{k_N}}\right)\theta_{x_1^{k_1}} \ln \frac{\theta_{x_1^{k_1}}}{\hat{\theta}_{x_1^{k_1}}} \\
&\quad + \sum_{k_2=1}^{K_2}\left(\sum_{k_1=1}^{K_1}\sum_{k_3=1}^{K_3}...\sum_{k_N=1}^{K_N} \theta_{x_1^{k_1}}\theta_{x_3^{k_3}}...\theta_{x_N^{k_N}}\right)\theta_{x_2^{k_2}} \ln \frac{\theta_{x_2^{k_2}}}{\hat{\theta}_{x_2^{k_2}}} \\
&\quad +... + \sum_{k_N=1}^{K_N}\left(\sum_{k_1=1}^{K_1}...\sum_{k_{N-1}=1}^{K_{N-1}} \theta_{x_1^{k_1}}...\theta_{x_{N-1}{}^{k_{N-1}}}\right)\theta_{x_N^{k_N}} \ln \frac{\theta_{x_N^{k_N}}}{\hat{\theta}_{x_N^{k_N}}}, \qquad (3.1)
\end{aligned}$$

where K_i denotes the number of possible values for the ith variable X_i, $\theta_{x_i^{k_i}}$ denotes the true parameter for the ith variable with an instantiation of the k_ith value, and $\hat{\theta}_{x_i^{k_i}}$ denotes the corresponding estimate. Using the property of probability distribution $\sum_{k_2=1}^{K_2}...\sum_{k_N=1}^{K_N}\theta_{x_2^{k_2}}...\theta_{x_N^{k_N}} = 1$, $\sum_{k_1=1}^{K_1}\sum_{k_3=1}^{K_3}...\sum_{k_N=1}^{K_N}\theta_{x_1^{k_1}}\theta_{x_3^{k_3}}...\theta_{x_N^{k_N}} = 1$,..., $\sum_{k_1=1}^{K_1}...\sum_{k_{N-1}=1}^{K_{N-1}}\theta_{x_1^{k_1}}...\theta_{x_{N-1}{}^{k_{N-1}}} = 1$, we can simplify Equation 3.1 to

$$\begin{aligned}
\mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}) &= \sum_{k_1=1}^{K_1}\theta_{x_1^{k_1}} \ln \frac{\theta_{x_1^{k_1}}}{\hat{\theta}_{x_1^{k_1}}} + \sum_{k_2=1}^{K_2}\theta_{x_2^{k_2}} \ln \frac{\theta_{x_2^{k_2}}}{\hat{\theta}_{x_2^{k_2}}} + ... + \sum_{k_N=1}^{K_N}\theta_{x_N^{k_N}} \ln \frac{\theta_{x_N^{k_N}}}{\hat{\theta}_{x_N^{k_N}}} \\
&= \sum_i \mathrm{KL}(\boldsymbol{\theta}_i||\hat{\boldsymbol{\theta}}_i). \qquad (3.2)
\end{aligned}$$

It is clear that KL-divergence is strongly decomposable for the independent variable problem. We shall see in the next chapter that KL divergence is decomposable over general Bayesian networks but not strongly decomposable.

Risk Function

For the independent variable problem, the risk function also has the nice strong decomposability. With the parameter independence assumption, it decomposes as the summation of the risk for each

beta distribution:

$$
\begin{aligned}
Risk(p_{\boldsymbol{\theta}}) &= E[\mathrm{KL}(p_{\boldsymbol{\theta}})] = \int_{\boldsymbol{\theta}} \mathrm{KL}(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}})p(\boldsymbol{\theta})d\boldsymbol{\theta} \\
&= \int_{\boldsymbol{\theta}_1\boldsymbol{\theta}_2...\boldsymbol{\theta}_N} p(\boldsymbol{\theta}_1)p(\boldsymbol{\theta}_2)...p(\boldsymbol{\theta}_N)\sum_i \mathrm{KL}(\boldsymbol{\theta}_i||\hat{\boldsymbol{\theta}}_i)d\boldsymbol{\theta}_1\boldsymbol{\theta}_2...\boldsymbol{\theta}_N \\
&= \int_{\boldsymbol{\theta}_1} p(\boldsymbol{\theta}_1)\left(\int_{\boldsymbol{\theta}_2...\boldsymbol{\theta}_N} p(\boldsymbol{\theta}_2)...p(\boldsymbol{\theta}_N)d\boldsymbol{\theta}_2...\boldsymbol{\theta}_N\right)\mathrm{KL}(\boldsymbol{\theta}_1||\hat{\boldsymbol{\theta}}_1)d\boldsymbol{\theta}_1 \\
&\quad + \int_{\boldsymbol{\theta}_2} p(\boldsymbol{\theta}_2)\left(\int_{\boldsymbol{\theta}_1\boldsymbol{\theta}_3...\boldsymbol{\theta}_N} p(\boldsymbol{\theta}_1)p(\boldsymbol{\theta}_3)...p(\boldsymbol{\theta}_N)d\boldsymbol{\theta}_1\boldsymbol{\theta}_3...\boldsymbol{\theta}_N\right)\mathrm{KL}(\boldsymbol{\theta}_2||\hat{\boldsymbol{\theta}}_2)d\boldsymbol{\theta}_2 \\
&\quad + ... + \int_{\boldsymbol{\theta}_N} p(\boldsymbol{\theta}_N)\left(\int_{\boldsymbol{\theta}_1...\boldsymbol{\theta}_{N-1}} p(\boldsymbol{\theta}_1)...p(\boldsymbol{\theta}_{N-1})d\boldsymbol{\theta}_1...\boldsymbol{\theta}_{N-1}\right)\mathrm{KL}(\boldsymbol{\theta}_N||\hat{\boldsymbol{\theta}}_N)d\boldsymbol{\theta}_N.
\end{aligned}
\tag{3.3}
$$

As in the previous derivation $\int_{\boldsymbol{\theta}_2...\boldsymbol{\theta}_N} p(\boldsymbol{\theta}_2)...p(\boldsymbol{\theta}_N)d\boldsymbol{\theta}_2...\boldsymbol{\theta}_N$, $\int_{\boldsymbol{\theta}_1\boldsymbol{\theta}_3...\boldsymbol{\theta}_N} p(\boldsymbol{\theta}_1)p(\boldsymbol{\theta}_3)...p(\boldsymbol{\theta}_N)d\boldsymbol{\theta}_1\boldsymbol{\theta}_3...\boldsymbol{\theta}_N$, ..., $\int_{\boldsymbol{\theta}_1...\boldsymbol{\theta}_{N-1}} p(\boldsymbol{\theta}_1)...p(\boldsymbol{\theta}_{N-1})d\boldsymbol{\theta}_1...\boldsymbol{\theta}_{N-1}$ all integrate to 1. Thus, Equation 3.3 equals

$$
\begin{aligned}
Risk(p_{\boldsymbol{\theta}}) &= \int_{\boldsymbol{\theta}_1} p(\boldsymbol{\theta}_1)\mathrm{KL}(\boldsymbol{\theta}_1||\hat{\boldsymbol{\theta}}_1)d\boldsymbol{\theta}_1 + \int_{\boldsymbol{\theta}_2} p(\boldsymbol{\theta}_2)\mathrm{KL}(\boldsymbol{\theta}_2||\hat{\boldsymbol{\theta}}_2)d\boldsymbol{\theta}_2 + ... + \int_{\boldsymbol{\theta}_N} p(\boldsymbol{\theta}_N)\mathrm{KL}(\boldsymbol{\theta}_N||\hat{\boldsymbol{\theta}}_N)d\boldsymbol{\theta}_N \\
&= \sum_i Risk(p_{\boldsymbol{\theta}_i}).
\end{aligned}
\tag{3.4}
$$

Tong shows that when the parameters are Dirichlet distributed, the risk for each parameter distribution can be computed as

$$
\begin{aligned}
Risk(p_{\boldsymbol{\theta}_i}) &= \int_{\boldsymbol{\theta}_i} p(\boldsymbol{\theta}_i)\mathrm{KL}(\boldsymbol{\theta}_i||\hat{\boldsymbol{\theta}}_i)d\boldsymbol{\theta}_i \\
&= \sum_{k=1}^{K_i} \frac{\alpha_{x_i^k}}{\sum_{j=1}^{K_i}\alpha_{x_i^j}} \times (\Psi(\alpha_{x_i^k}+1) - \Psi(\sum_{j=1}^{K_i}\alpha_{x_i^j}+1)) - \sum_{k=1}^{K_i}\hat{\theta}_{x_i^k}\ln\hat{\theta}_{x_i^k} \\
&= \sum_{k=1}^{K_i} \frac{\alpha_{x_i^k}}{\sum_{j=1}^{K_i}\alpha_{x_i^j}} \times (\Psi(\alpha_{x_i^k}+1) - \Psi(\sum_{j=1}^{K_i}\alpha_{x_i^j}+1)) + H(P(X_i)),
\end{aligned}
\tag{3.5}
$$

where the Digamma function $\Psi(\alpha) = \frac{d}{dx}\ln\Gamma(x) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$ is defined as the derivative of the logarithm of the gamma function, and the entropy function $H(P(X)) = -\sum_x P(x)\ln P(x)$. See Tong's thesis for a full derivation [34].

3.1.2 Update Rule

Since all the variables are independent and every time we probe a variable the outcome, which falls in the domain of X_i, is fully observed, we can use the standard Bayesian update rule discussed in Section 2.2.2 to compute the posterior distributions.

3.2 Computational Complexity

In this section, we show that it is NP-hard to solve the independent variable problem even with the restriction that the domain size of each variable is 8.

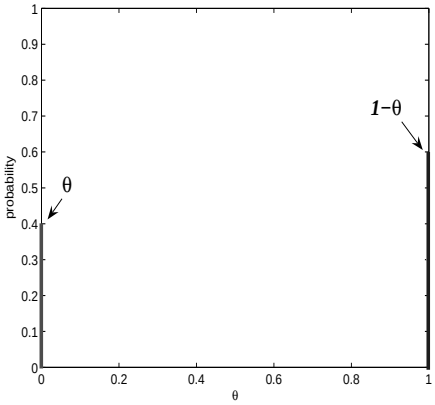


Figure 3.1: An elephant(0.4)-coin.

Definition 3.2.1. [Elephant-Coin] For any $a \in [0, 1]$, a coin is an *elephant(a)-coin* if

1. with probability a, it is a double-headed coin *i.e.* $p(\theta = 1) = a$,
2. with probability $1 - a$, it is a double-tailed coin *i.e.* $p(\theta = 0) = 1 - a$,

where θ is the head probability of the coin. It is called an elephant-coin because it never forgets: if it is seen to be a head once, it will always be heads; if it is seen to be a tail once, it will always be tails. Refer to Figure 3.1 for an example of an elephant(0.4)-coin.

Note that $E[\theta] = a$, $P(X = 1|\theta = 1) = 1, P(X = 1|\theta = 0) = 0$, and $P(X = 1|E[\theta] = s) = a$. Also,

$$\begin{aligned} Risk(p_{\boldsymbol{\theta}}) &= E[\mathrm{KL}(\boldsymbol{\theta}||E[\boldsymbol{\theta}])] \\ &= P(\theta = 1) \times \mathrm{KL}_{\theta=1}(\boldsymbol{\theta}||E[\boldsymbol{\theta}]) + P(\theta = 0) \times \mathrm{KL}_{\theta=0}(\boldsymbol{\theta}||E[\boldsymbol{\theta}]) \\ &= P(\theta = 1) \times \left[1 \times \ln \frac{1}{a} + 0 \times \ln \frac{0}{1-a}\right] \\ &\quad + P(\theta = 0) \times \left[0 \times \ln \frac{0}{a} + 1 \times \ln \frac{1}{1-a}\right] \\ &= a \times (-\ln a) + (1 - a) \times (-\ln (1 - a)), \end{aligned} \tag{3.6}$$

which is exactly the entropy $H(p_{\boldsymbol{\theta}})$ of the distribution $p_{\boldsymbol{\theta}}$.

Moreover, after a single flip, we know the coin is either a double-headed or a double-tailed coin. Either way, $Risk(p_{\boldsymbol{\theta}}|\text{1-flip}) = 0$. As a result, a single flip changes the information of an elephant coin from $H(p_{\boldsymbol{\theta}}) = -a \ln a - (1 - a) \ln (1 - a)$ to 0.

Definition 3.2.2. [Independent Variable Decision Problem]

INSTANCE: For a set of variables $\boldsymbol{S} = \{S_1, ..., S_M\}$, where each variable is associated with a cost $c(S_j)$ and a probability distribution p_{S_j}, a budget $\mathcal{B} \in \mathbb{R}_{>0}$, and a target value $V \in \mathbb{R}_{>0}$.

QUESTION: Is there a subset $\boldsymbol{S}' \subset \boldsymbol{S}$ such that

1. $\sum_{S_j \in \boldsymbol{S}'} c(S_j) \leq \mathcal{B}$,
2. $\sum_{S_i \in \boldsymbol{S}'} H(p_{S_i}) \geq V$.

Now consider, the *exact cover by 3-sets (X3C)* problem, which is known to be NP-complete:

Definition 3.2.3. [X3C]

INSTANCE: For a set of elements $\boldsymbol{X} = \{X_1, ..., X_{3K}\}$ and a collection of 3-element subsets of $\boldsymbol{X}$: $\boldsymbol{S} = \{S_1, ..., S_M\}$, where $\forall j |S_j| = 3$ and $S_j \subseteq \boldsymbol{X}$. Write $S_j = \{X_{j1}, X_{j2}, X_{j3}\}$.

QUESTION: Is there a subset $\boldsymbol{S}' \subset \boldsymbol{S}$, where $|\boldsymbol{S}'| = K$, that is an exact cover of $\boldsymbol{X}$ *i.e.* $\cup_{S_j \in \boldsymbol{S}'} S_j = \boldsymbol{X}$.

Note this means each X_i is in exactly one $S_j \in \boldsymbol{S}'$. Moreover, X3C remains NP-complete, even if each X_i appears in at most 3 S_js [11].

We need the following lemma for providing the hardness result.

Lemma 3.2.4. *For $f(t) = \frac{1}{4^{2t}}$ where $t \in \mathbb{Z}_{\geq 2}$,*

$$4 \times H(p_{f(t+1)}) < H(p_{f(t)}), \tag{3.7}$$

where $H(p_{f(t)}) = -f(t) \ln f(t) - (1 - f(t)) \ln (1 - f(t))$.

Proof. See Appendix A. □

With Lemma 3.2.4, we now show that the independent variable decision problem is NP-hard [1].

Theorem 3.2.5. *The independent variable decision problem is NP-hard.*

Proof. To show the problem is NP-hard, we reduce an arbitrary X3C with the constraint that each element X_i appears in at most 3 S_js to an independent variable decision problem where each variable has a domain size of 8. We map the X3C problem to an independent variable decision problem with M different variables, where each variable corresponds to a 3-element subset S_j. Each element X_i in X3C is viewed as an elephant($f(i)$)-coin, where $f(i) = \frac{1}{4^{2(N+2-i)}}$, and a cost associated with X_i is defined as $c(X_i) = 4^i$. Probing variable $S_j = \{X_{j1}, X_{j2}, X_{j3}\}$ will cost $c(S_j) = 4^{j1} + 4^{j2} + 4^{j3}$ and will basically "flip" all the 3 coins in the corresponding subset, which reduces the entropy of this variable from $H(p_{S_j}) = H(p_{f(j1)}) + H(p_{f(j2)}) + H(p_{f(j3)})$ to 0. Finally, set the budget of the independent variable decision problem to be $\mathcal{B} = \sum_{i=1}^{3K} 4^i$ and the total value to be $V = \sum_{i=1}^{3K} H(p_{f(i)})$.

[1] We can trivially map the Knapsack problem to our independent variable decision problem by setting the weight of each item to the cost of each variable, the value of each item to the value of each variable, the weight limit to the budget, and the target value of the knapsack problem to the target value of the independent variable decision problem. However, it is not clear how we can set the prior distributions, which involves computing the inverse function of the entropies.

Suppose there is a solution to the independent variable decision problem, *i.e.*, $\boldsymbol{S'}$ is a subset of variables in $\boldsymbol{S}$ such that $\sum_{S_j \in \boldsymbol{S'}} c(S_j) \leq \mathcal{B}$ and $\sum_{S_j \in \boldsymbol{S'}} H(p_{S_j}) \geq V$. We use induction to show that the corresponding union of $\cup_{S_j \in \boldsymbol{S'}} S_j$ for the X3C problem contains each x_i exactly once.

Start from X_N, where $N = 3K$. If X_N does not appear in $\cup_{S_j \in \boldsymbol{S'}} S_j$, $\sum_{S_j \in \boldsymbol{S'}} H(p_{S_j}) \leq \sum_{i=1}^{N-1} 3H(p_{f(i)}) < 4H(p_{f(N-1)}) < H(p_{f(N)}) < \sum_{i=1}^{3K} H(p_{f(i)}) = V$, where the first inequality follows from the assumption that each element appears at most 3 times in X3C, the second and third inequality follows from Lemma 3.2.4. This contradicts our assumption that $\boldsymbol{S'}$ is a solution to the independent variable decision problem because $\sum_{S_j \in \boldsymbol{S'}} H(p_{S_j}) \geq V$. So X_N has to appear at least once in $\cup_{S_j \in \boldsymbol{S'}} S_j$. If X_N appears more than once in $\cup_{S_j \in \boldsymbol{S'}} S_j$, $\sum_{S_j \in \boldsymbol{S'}} c(S_j) > 2c(X_N) \geq 2 \times 4^N > \sum_{i=1}^{N} 4^i = \mathcal{B}$, which again contradicts our assumption that $\boldsymbol{S'}$ is a solution to the independent variable decision problem because $\sum_{S_j \in \boldsymbol{S'}} c(S_j) \leq \mathcal{B}$. Thus, X_N has to appear no more than once in $\cup_{S_j \in \boldsymbol{S'}} S_j$. Combining the arguments from both the target value function and the budget, X_N appears exactly once in $\cup_{S_j \in \boldsymbol{S'}} S_j$. Use a similar approach, we can show that each X_i has to appear in $\cup_{S_j \in \boldsymbol{S'}} S_j$ exactly once. As a result, $\boldsymbol{S'}$ is a solution to the original X3C problem.

Now, suppose there is a solution to the X3C problem, *i.e.* $\boldsymbol{S'}$ is a subset of 3-element sets in $\boldsymbol{S}$ such that $|\boldsymbol{S'}| = K$ and $\cup_{S_j \in \boldsymbol{S'}} S_j = \boldsymbol{X}$. $\sum_{S_j \in \boldsymbol{S'}} c(S_j) = \sum_{i=1}^{3K} c(X_i) \leq \mathcal{B}$, and $\sum_{S_j \in \boldsymbol{S'}} H(p_{S_j}) = \sum_{i=1}^{3K} H(p_{f(i)}) \geq V$. Thus, $\boldsymbol{S'}$ is a solution to the independent variable decision problem.

□

It remains unclear if with one or more of the following constraints the independent variable problem is still NP-hard or not: binary variables, uniform costs, and unimodal prior distributions; *e.g.* a problem of binary variables with uniform costs and beta priors.

3.3 Policies

In this section, we describe several policies that are applicable to the independent variable problem. In particular, we show that an iterative greedy algorithm is an optimal allocation algorithm for the uniform-cost independent binary-variable problem. Other policies discussed include: round-robin, random; greedy, weighted greedy, and exhaustive search. We provide formal descriptions, together with theoretical and performance discussions.

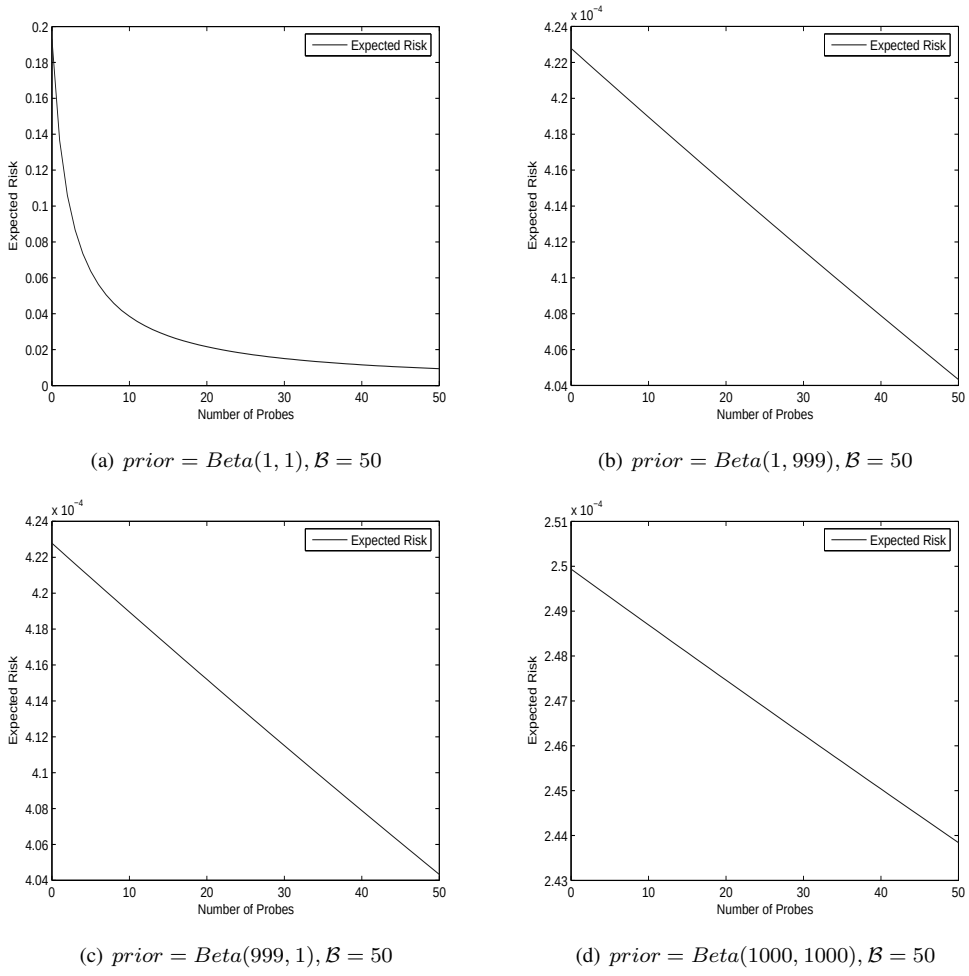
3.3.1 Adaptivity

An *adaptive policy* makes decisions sequentially, during the probing process. Hence, it can use information about outcomes of precursors, when deciding which variable to probe next. On the other hand, a *non-adaptive policy* allocates all the probes at the beginning. Therefore, it cannot use information of results of variable probes. Lizotte *et al.* call these non-adaptive policies "allocations" [23]. For a more elaborate explanation on the effects of this adaptivity, please refer to the first chapter of Dean's thesis [8].

3.3.2 Allocation Policies

In this section, we study an iterative greedy allocation algorithm, which we will prove to be the optimal allocation algorithm for the uniform-cost independent binary-variable problem, and two other well-known allocation algorithms as baselines for our empirical studies.

Iterative Greedy Allocation



(a) $prior = Beta(1, 1), \mathcal{B} = 50$

(b) $prior = Beta(1, 999), \mathcal{B} = 50$

(c) $prior = Beta(999, 1), \mathcal{B} = 50$

(d) $prior = Beta(1000, 1000), \mathcal{B} = 50$

Figure 3.2: Expected risks computed with different priors and a budget $\mathcal{B} = 50$. The height of each point at $x = b$ is the expectation taken over all the possible outcomes of b probes. (a) starts with $Beta(1, 1)$; (b) starts with $Beta(1, 999)$; (c) starts with $Beta(999, 1)$; (d) starts with $Beta(1000, 1000)$. Note all these figures re-scale on y-axis.

Although the independent variable problem is NP-hard, we shall show that among allocation algorithms, the uniform-cost independent variable problem can be solved optimally by an efficient iterative greedy algorithm. First, we need to define the score function that the iterative greedy

algorithm uses to make the allocations. We use the expected risk function as this score

$$ExpRisk_{prior}(b) = \sum_{o^b} P(o^b) Risk(p_{\boldsymbol{\theta}}|o^b), \tag{3.8}$$

where b denotes the number of probes, o^b denotes one possible outcome after b probes. For example, if we start with a binary variable X whose prior is $Beta(2,3)$, *i.e.* $\boldsymbol{\theta} \sim Beta(2,3)$ and $b = 1$, $\sum_{o^b} P(o^b) Risk(p_{\boldsymbol{\theta}}|o^b) = P(o^1 = F) Risk(p_{\boldsymbol{\theta}}|o^1 = F) + P(o^1 = T) Risk(p_{\boldsymbol{\theta}}|o^1 = T) = \frac{2}{5} Risk(Beta(3,3)) + \frac{3}{5} Risk(Beta(2,4))$. The $ExpRisk$ is actually a function of both the number of probes b and the $prior$. However, since the $prior$ is fixed when the $ExpRisk$ is computed, we put it as a subscript.

Figure 3.2 plots the expected risk of one variable starting from different priors. Each point is the expectation taken over all the possible outcomes of b probes. We make four observations on the expected risk function. First, as mentioned before, a highly unbalanced beta distribution (Figure 3.2(b)) has a much lower expected risk than a flat beta distribution (Figure 3.2(d)). That is, it is harder for one to be sure about a variable X whose $P(X = 1) = 0.5$ than a variable Y whose $P(Y = 1) = 0.9$. Second, as indicated by the risk function, the expected risk is symmetric to the effective sample size of 0s and 1s (Figure 3.2(b) and 3.2(c)) [2]. Third, when the effective sample sizes are comparatively small, the expected risk is dominated by the effective sample sizes. In other words, a greedy learner would prefer variables with small effective sample sizes most of the time. Finally, the expected risk function is strictly monotonically decreasing and convex. It is the rescaling of the Y axis that makes Figure 3.2(b), 3.2(c), and 3.2(d) look steeper. The last observation is crucial in the optimal allocation proof of the iterative greedy algorithm.

We will first demonstrate how the expected risk function can be efficiently computed, and then prove its monotonicity and convexity. The following remark can be made on the computation of the expected risk.

Remark 3.3.1. [Polynomial $ExpRisk$ Computation Time] To compute $ExpRisk_{prior_i}(b)$, one needs to build a DAG with $\frac{(b+1)(b+2)}{2}$ number of nodes and compute $b+1$ risks of possible distributions at the bth level of the DAG. Level starts from zero.

The function can be computed by building a DAG of all reachable states given the budget $\mathcal{B}$ and the prior, as illustrated in Figure 3.3. Each probability in the triangle can be computed easily from its parents, and the distributions can be determined by the column and row number. For example, if $prior_i$ is $Beta(\alpha, \beta)$, where $\alpha = 2$ and $\beta = 3$ (root of the DAG) and we want to compute the risk of the distribution at $Row = 3$ and $Column = 2$, the probability is given by $\frac{2}{5} \times \frac{1}{2} + \frac{3}{5} \times \frac{1}{3} = \frac{2}{5}$ and the distribution is given by $Beta(\alpha + Row - Column, \beta + Column - 1) = Beta(3,4)$.

Theorem 3.3.2. *The expected risk function for a variable with $prior = Beta(\alpha, \beta)$, $\alpha, \beta \in \mathbb{R}_{>0}$ is strictly monotonically decreasing in the number of probes b and is convex.*

[2] This is different from the discriminative budgeted learning problem [26, 23] and the classic bandit problem [13]. It makes sense because ,in our setting, the learner tries to learn the distributions and 0s and 1s are just two symbols. For the discriminative budgeted learning problem and the bandit problem, 1s are preferred.

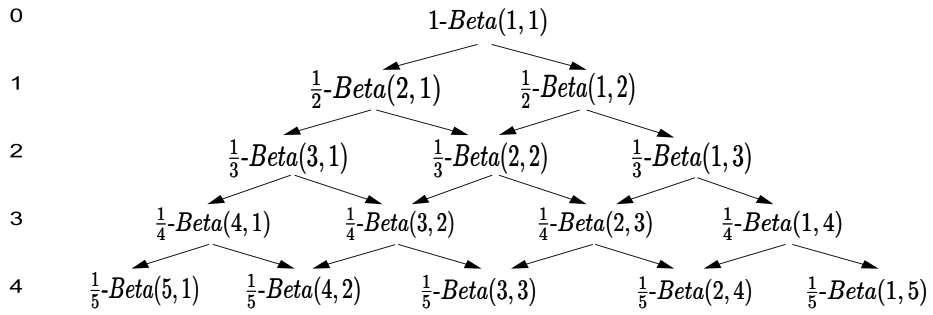


Figure 3.3: A DAG built to compute the expected risk starting from a $Beta(1,1)$. All the beta distributions denote the posterior states after probing the variable for the depth of time. The fractions denote the probability for reaching that state. This DAG should not be confused with the Bayesian network structures.

Proof. See Appendix A. □

The properties can be understood from an information theory perspective: in the expected sense, by probing a variable the learner always gathers more information about the true underlying distribution, and information never hurts. Moreover, the more probes the learner assigns to a variable, the less additional information the learner gets from another probe. These properties of the $ExpRisk$ function, together with the strong decomposability, are necessary for proving that the iterative greedy algorithm gives an optimal allocation for the uniform-cost independent binary-variable problem. If the $ExpRisk$ does not have these properties, even finding an optimal allocation of probes is actually NP-complete [5]. See also Theorem 3.2.5.

We define the iterative greedy allocation algorithm as follows: (1) For each variable and each budget $b \in \{1, ..., \mathcal{B}\}$, precompute and store the expected risk as defined in Equation 3.8. The strong decomposability allows us to do so efficiently, and we only have to compute each expected risk function up to the budget $\mathcal{B}$. Keep the numbers of allocated probes for each variable in a vector $allocation$, where each $allocation_i$ is initialized as 0; (2) Compute $RD_i(allocation_i + 1) = ExpRisk_{prior_i}(allocation_i) - ExpRisk_{prior_i}(allocation_i + 1)$ for each variable, and pick the variable that gives the greatest one-step risk reduction per cost, increase $allocation_i$ by 1, and deduct the budget B by c_i; the 2nd step is then iterated until the budget is exhausted. Pseudocode of the iterative greedy allocation algorithm is shown in Figure 3.4. Also refer to Section 3.4 for an example. We can show that the allocation generated by the iterative greedy algorithm is an optimal allocation.

Theorem 3.3.3. *The iterative greedy algorithm is an optimal allocation algorithm for the uniform-cost independent binary-variable problem in that it generates an allocation of probes minimizing expected risk $ExpRisk$.*

Proof. Strict monotonicity and convexity properties mean that for each variable X_i, $RD_i(b) \geq 0$ for $b \in \mathbb{Z}_{\geq 0}$ and $RD_i(b+1) < RD_i(b)$. We can re-express any allocation $A = [a_1, a_2, ..., a_t] \in$

Algorithm 1: Iterative Greedy Allocation

Input: N variables with priors $\{prior_1, ..., prior_N\}$ and costs $\{c_1, ..., c_N\}$, and a budget $\mathcal{B}$
Output: An $allocation$ = $\{allocation_1, ..., allocation_i, ..., allocation_N\}$

foreach *Variable* i **do**
  $expectedRiskFunction_i$ = computeExpectedRiskFunction($\{prior_1, ..., prior_N\}$, $\mathcal{B}$);
  $allocation_i$ = 0;
end
while $\mathcal{B} > 0$ **do**
  bestI = 0;
  foreach *Variable* i **do**
    RD_i = $expectedRiskFunction_i(allocation_i)$ - $expectedRiskFunction_i(allocation_i + 1)$;
  end
  bestI = $argmax_i(RD_i/c_i)$;
  $\mathcal{B} = \mathcal{B} - c_{\text{bestI}}$;
  if $\mathcal{B} \geq 0$ **then**
    $allocation_{bestI}$ = $allocation_{bestI}$ + 1;
  end
end

Figure 3.4: Iterative Greedy Allocation Policy

$\{1, 2, ..., \mathcal{B}\}^n$ as a size-$\mathcal{B}$ subset of the set of all possible indices, $IJ = \{(i, j) : i = 1, ..., n; j = 1, ..., B\} : A' = \{(1, 1), ..., (1, a_1); (2, 1), ..., (2, a_2); ...; (n, 1), ..., (n, a_n)\}$.

We use $Alloc(S)$ to denote that a subset $S \subset IJ$ is an allocation. Note that $Alloc(S)$ is true if and only if $(i, j + 1) \in S \Rightarrow (i, j) \in S$. In general, the final expected risk associated with any such set of indices is

$$\begin{aligned} ExpRisk(A) &= \sum_i ExpRisk_i(0) - \left[\sum_i \sum_{j=1}^{a_i} RD_i(j)\right] \\ &= \sum_i ExpRisk_i(0) - \left[\sum_{(i,j)\in A'} RD_i(j)\right] \end{aligned} \tag{3.9}$$

To minimize this, we just need to maximize

$$f(A') = \sum_{(i,j)\in A'} RD_i(j). \tag{3.10}$$

So our task is to find $C^* = argmax_{C \subset IJ, |C|=\mathcal{B}, Alloc(C)} \sum_{(i,j)\in C} RD_i(j)$, *i.e.*, find the size-$\mathcal{B}$ allocation subset of IJ. Now consider the simpler task of just finding the size-$\mathcal{B}$ subset of IJ: $D^* = argmax_{D \subset IJ, |D|=B} \sum_{[i,j]\in D} RD_i(j)$, *i.e.*, ignore the $Alloc(D)$ constraint. Notice

1. It is trivial to compute this D^*: we can simply take the $\mathcal{B}$ largest values of IJ.
2. This D^* is necessarily allocation, *i.e.*, it satisfies $Alloc(D^*)$.

   Proof. Towards a contradiction, assume D^* is not an allocation, *i.e.*, there is a (i, j) such that D^* includes (i,j+1) but not (i, j). Now let $D' = D + (i, j) - (i, j + 1)$, and note

$f(D') = f(D) + RD_i(j) - RD_i(j+1) > f(D)$. This means that D^* was not an optimal size-$\mathcal{B}$ subset of IJ. □

Hence, the optimal allocation is just the $\mathcal{B}$ largest values of $\{RD_i(j)\}$. Note this is exactly what iterative greedy allocation algorithm returns. (This requires re-using the observation 2. above: the largest size-$\mathcal{B}$ allocation is just the largest size-$\mathcal{B}$ subset.)

□

The arguments hold for not only binomial beta distributions but also multinomial Dirichlet distributions. For the uniform-cost independent variable problem, we can show that the iterative greedy allocation algorithm is also *consistent*.

Definition 3.3.4. [Consistency] A learning algorithm is consistent if it converges to the density that generates the data as the size of the data samples tends to infinity.

Theorem 3.3.5. *The iterative greedy allocation algorithm is consistent for the independent variable problem.*

Proof. As the standard Bayesian update is known to be consistent [34], we need only show that each variable will be probed infinitely often as the budget tends to infinity. Without loss of generality, we have to show that very variable g will be probed once after a certain number of probes. Notice that the expected risk function for each variable is strictly decreasing and convex. The one-step risk reduction $RD_f(b^f)$ of any other variable f tends to zero $\lim_{b^f \to +\infty} RD_f(b^f) = 0$ as the number of probes assigned to that variable increases. This can be easily seen from the definition of the RD function. Therefore, $RD_g(b^g) > RD_f(b^f)$ has to be true after a certain number of probes. When $\forall f \in \boldsymbol{X} \backslash g \; RD_g(b^g) > RD_f(b^f)$, variable g will then be probed by the iterative greedy allocation algorithm. □

Actually, using a similar approach we can show that all the other allocation and adaptive policies that we shall examine in the paper are consistent.

Round-Robin

The round-robin policy probes variable 1 through N in turn. Every time a variable is probed the corresponding cost is deduct from the budget. The algorithm then loops around the variables, until the budget is exhausted. For an independent variable problem with N uniform-cost variables, and a budget $\mathcal{B}$, the round-robin algorithm allocates $\lfloor \mathcal{B}/N \rfloor + 1$ probes to the first $\mathcal{B} - \lfloor \mathcal{B}/N \rfloor \times N$ variables and $\lfloor \mathcal{B}/N \rfloor$ to the rest of the variables. It does not take the costs of variables into consideration. Thus, it is a uniform allocation of probes, but might not be a uniform allocation of resources. Figure 3.5 gives the pseudocode of the round-robin policy.

Algorithm 2: Round Robin

Input: N variables with priors $\{prior_1, ..., prior_N\}$ and costs $\{c_1, ..., c_N\}$, and a budget $\mathcal{B}$

Output: An $allocation$ = $\{allocation_1, ..., allocation_N\}$

```
Initialize elements of allocation to zeros;
while B > 0 do
    foreach Variable i do
        B = B − c_i;
        if B ≥ 0 then
            allocation_i = allocation_i + 1;
        end
    end
end
```

Algorithm 3: Random

Input: N variables with priors $\{prior_1, ..., prior_N\}$ and costs $\{c_1, ..., c_N\}$, and a budget $\mathcal{B}$

Output: An $allocation$ = $\{allocation_1, ..., allocation_N\}$

```
Initialize elements of allocation to zeros;
while B > 0 do
    /* Randomly generate i.  */
    i = randInt([1, ..., N]);
    B = B − c_i;
    if B ≥ 0 then
        allocation_i = allocation_i + 1;
    end
end
```

Figure 3.5: Baseline allocation policies

Random

The random policy draws each probe from a uniform discrete distribution, *i.e.* probability of probing any variable i is $\frac{1}{N}$ when there are N variables. Since random and round-robin both asymptotically assign uniform number of probes to variables, when the budget $\mathcal{B}$ is large enough, they should have similar performances. The random policy also ignores the costs of variables. Refer to Figure 3.5 for the pseudocode.

The advantages of the two baseline allocation policies, random and round-robin, are their computational efficiency and their robustness against bad priors. We use "bad" priors to refer to the priors that are misleading. That is they are not the true priors that generates the parameters.

3.3.3 Adaptive Policies

In this subsection, we discuss adaptive policies including the greedy, weighted greedy, and the exhaustive search policies. These policies employ the information of the priors, the posterior states after every probe, the costs, and the budget when choosing the next variable. Hence, they require more computational time, and their performance may suffer if the prior information is misleading.

Greedy

An adaptive greedy policy can be described as follows: compute the one-step expected risk for each variable, pick the one with the largest one-step risk reduction $RD_i = Risk_{prior_i} - ExpRisk_{prior_i}(1)$, probe the that variable, update the current state using the outcome (standard Bayesian update), and iterate the process until the learner runs out of budget. Pseudocode of the greedy algorithm is shown in Figure 3.6. The greedy algorithm only needs to compute the one-step expected risk

Algorithm 4: Adaptive Greedy

Input: N variables with priors $\{prior_1, ..., prior_N\}$ and costs $\{c_1, ..., c_N\}$, and a budget $\mathcal{B}$
Output: N posteriors $\{posterior_1, ..., posterior_N\}$

$\{posterior_1, ...posterior_N\} = \{prior_1, ..., prior_N\}$;
while $\mathcal{B} > 0$ **do**
    bestI = 0;
    maxRD = 0;
    foreach *Variable* i **do**
        /* Greedy: RD = expectedRiskReduction($posterior_i$); */
        $RD = \frac{\text{expectedRiskReduction}(posterior_i)}{c_i}$;
        if $RD > maxRD$ **then**
            maxRD = RD;
            bestI = i;
        end
    end
    $\mathcal{B} = \mathcal{B} - c_{\text{bestI}}$;
    if $\mathcal{B} \geq 0$ **then**
        outcome = probe(bestI);
        $posterior_{\text{bestI}}$ = update(outcome, $posterior_{\text{bestI}}$);
    end
end

Figure 3.6: Adaptive greedy

$ExpRisk_{prior_i}(1)$. Therefore, the computational complexity of greedy is $O(N)$. However, we can still gain computational time with the following lemma.

Lemma 3.3.6. *For uniform-cost binary variables with beta priors $\boldsymbol{\theta}_i \sim Beta(\alpha_i, \beta_i)$ and the same effective sample size, i.e. $\alpha_i+\beta_i$ is constant, the greedy learner prefers variables with more balanced parameter distributions. More precisely, if $\alpha_i + \beta_i = \alpha_j + \beta_j$ and $|\alpha_i - \beta_i| < |\alpha_j - \beta_j|$ for variable i and j, the greedy algorithm prefers variable i.*

Proof. See Appendix A. □

Therefore, the greedy learner can divide the whole set of variables into subsets, where variables in each subset have the same effective sample size. For each subset of variables, the greedy learner only needs to compute the value $|\alpha_i - \beta_i|$ for each beta distribution, and pick the variable with the smallest value. To make the final probe choice for the step, the greedy learner has to compute the one-step risk reductions for all of the subset-winner variables, and choose the one with the greatest risk reduction.

The greedy algorithm might not give a good approximation bound for the general independent variable problem. Let ε be an arbitrarily small positive quantity. Suppose we have one variable with a cost $c_1 = 1$ and the largest one-step risk reduction $RD_1 = r$, a large number of variables with costs $c_j = \varepsilon$, for $j = 2, ..., N$ and risk reductions $RD_j = r - \varepsilon$, for $j = 2, ..., N$, and a budget

$\mathcal{B} = 1$. The greedy algorithm would probe the first variable and get a risk reduction of r, but by probing the rest of the variables it is possible for the learner to get a much better risk reduction.

Algorithm 5: Exhaustive Search

Input: N variables with priors $\{prior_1, ..., prior_N\}$ and costs $\{c_1, ..., c_N\}$, and a budget $\mathcal{B}$
Output: N posteriors $\{posterior_1, ..., posterior_N\}$

$\{posterior_1, ...posterior_N\} = \{prior_1, ..., prior_N\}$;
`/* Precompute the risks for all variable states reachable given` $\mathcal{B}$. `*/`
riskTable = mkRiskTable;
`/* Grow the tree with all possible combinational states in a top-down approach. Each of the combinational state is a node. This can be done using a breath first search. */`
tree = growTree($\{posterior_1, ..., posterior_N\}$, $\mathcal{B}$);
`/* Compute the risk of each node; value of any leaf node is the sum of the risk for each variable; values for other nodes is the mean of the risks of their children. This is done in a bottom-up approach. */`
risks = computeRisks(tree, riskTable);
while $\mathcal{B} > 0$ **do**
    bestI = 0;
    minRisk = Inf;
    `/* find the variable with the minimum risk and probe that variable. */`
    foreach *Variable* i **do**
        risk = expectedRisk(risks, $\{posterior_1, ...posterior_N\}$);
        if *risk* $<$ *minRisk* **then**
            minRisk = risk;
            bestI = i;
        end
    end
    $\mathcal{B} = \mathcal{B} - c_{\text{bestI}}$;
    if $\mathcal{B} \geq 0$ **then**
        outcome = probe(bestI);
        $posterior_{\text{bestI}}$ = update(outcome, $posterior_{\text{bestI}}$);
    end
end

Figure 3.7: Dynamic programming of exhaustive search policy

Weighted Greedy

An immediate idea to solve this issue is to run a weighted greedy algorithm (greedy on $\frac{RD_i}{c(i)}$). Unfortunately, the weighted greedy algorithm by itself may not perform well either. Consider the case where we have two variables and the budget $\mathcal{B} = r$; the first variable X_1 having a cost $c_1 = r$ and a one-step risk reduction $RD_1 = r$, and the second variable having a cost $c_2 = (\frac{r}{2} + \varepsilon)(1 + \varepsilon)$ and a one-step risk reduction $RD_2 = \frac{r}{2} + \varepsilon$. The optimal solution is to probe X_1 and get a risk reduction of r, but the weighted greedy algorithm probes the second variable and get a risk reduction

of $(\frac{r}{2}+\varepsilon)(1+\varepsilon)$, which is slightly more than $\frac{r}{2}$. Pseudocode of the weighted greedy algorithm is given in Figure 3.6.

Exhaustive Search

Another well-studied algorithm is the exhaustive search. The idea is to enumerate all possible combinations of probe sequences that are reachable given the budget $\mathcal{B}$. Dynamic programming can be employed. See Figure 3.7 for the pseudocode. By definition, exhaustive search is optimal. However, due to the intensive computation, the exhaustive search algorithm is not practical for real problems. Also, it might be trapped by bad priors just as the greedy based algorithms. Because of its extensive time of computation, exhaustive search is not included in our empirical study. However, we will give an example in the following section to show how a deeper lookahead might be beneficial to our learner.

3.4 An Example

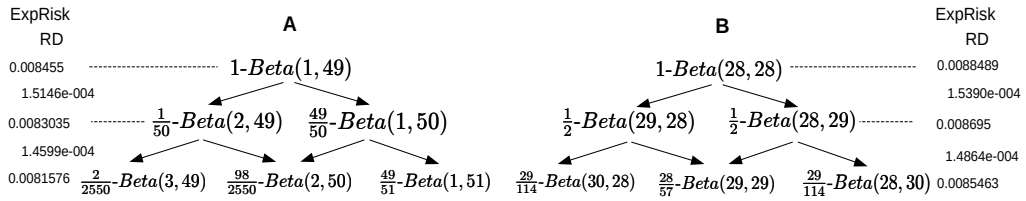


Figure 3.8: An example showing the benefit of a deeper lookahead with two DAGs built by the interative greedy allocation algorithm. In the example, the learner starts with two uniform-cost variables $Beta(1,49)$ and $Beta(28,28)$, and a budget $\mathcal{B}=2$. $ExpRisk$s for each level are displayed by the side of the DAGs (to the left for the left DAG, and to the right for the right DAG). Numbers with indentations are the difference between each pair of $ExpRisk$s *a.k.a.* the risk reductions.

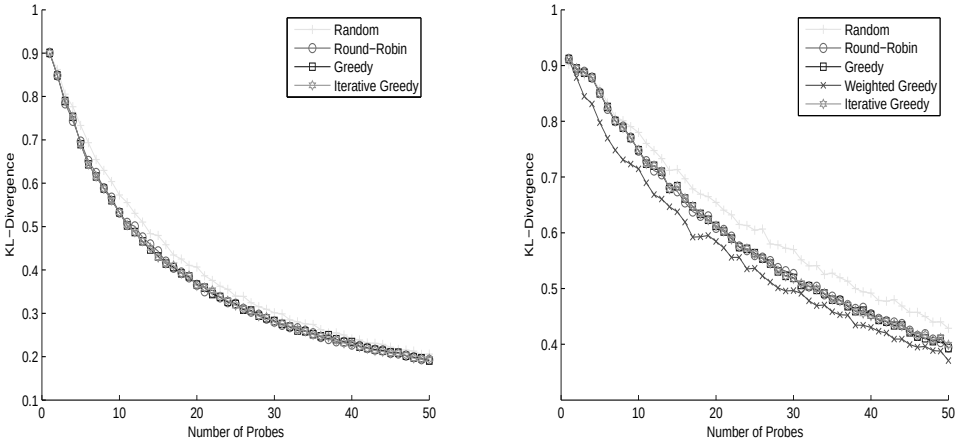
Figure 3.8 gives an example of two uniform-cost binary variables with a budget $\mathcal{B}$ of 2. The iterative greedy algorithm first computes the $ExpRisk$ for each variable at all levels reachable given $\mathcal{B}$. It then computes the one step risk reduction RD for each variable from the current level. In the example [3], $RD_{A1}=1.5146\times10^{-4}$ and $RD_{B1}=1.5390\times10^{-4}$. It then compares the RDs and allocates 1 probe to B, which gives the greatest RD at this step. In the second step, it compares $RD_{A1}=1.5146\times10^{-4}$ with $RD_{B2}=1.4864\times10^{-4}$, and allocates 1 probe to A. So the final allocation for this problem is 1 probe to A and 1 probe to B. It is clear that both the iterative greedy allocation and the adaptive greedy algorithm would probe the second and the first variable, because after 1 probe to B, this B would end up in states with the same risk and RD. However, the optimal policy is to probe the first variable; and if it is 0, probe it again ($RD_{Beta(2,49)}=1.6314\times10^{-4}$), otherwise, probe the second variable. The illustrates the benefit of a deeper lookahead. If we change

[3] We use RD_{A1} to denote the one step risk reduction for variable A from the first level.

the second variable to $Beta(28, 29)$, with the same analysis, we can show that both the adaptive greedy and the exhaustive search algorithm would behave exactly like the optimal policy in the original example, which is still optimal for the modified example. However, the iterative greedy allocation algorithm would probe the first variable and second variable in order, which is suboptimal. This illustrates the benefit of adaptivity.

3.5 Empirical Results

In this section, we report empirical results of a series of experiments to compare the effectiveness of the policies for the independent variable problem. Here, all the experiments are conducted on an independent variable problem with 5 binary variables and a budget of 50. Non-uniform costs are generated from a uniform discrete distribution on $\{1, 2, ..., 5\}$, where each integer cost has a probability of $\frac{1}{5}$. Non-uniform priors are generated as follows: first, an effective sample size e is sampled from a uniform discrete distribution on $\{10, 11, ..., 30\}$; α is then sampled from another interval $[1, e-1]$, and β is assigned as $e - \alpha$. To test the robustness of the algorithms, we also consider misleading (bad) priors, which are not the priors that generate the true $\boldsymbol{\theta}$s. When the priors are good, the true θs are generated from the priors; whereas, for bad priors, the true θs are generated from a uniform distribution. Each experiment is based on 4000 runs.

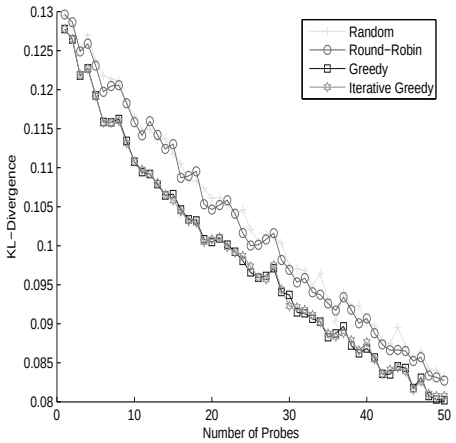


(a) Uniform-cost with uniform priors.

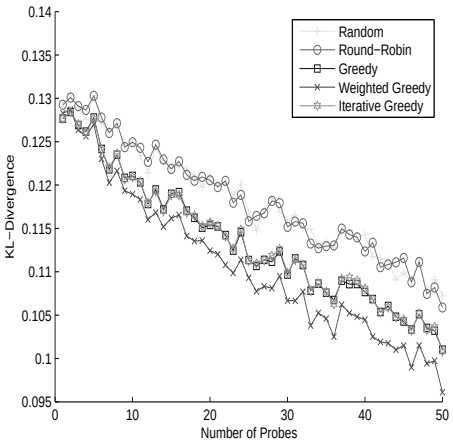
(b) Non-uniform-cost with uniform priors.

Figure 3.9: Uniform-cost and non-uniform-cost independent variable problem of 5 variables with uniform priors.

Figure 3.9(a) and 3.9(b) present the results starting with uniform priors. Uniform priors are not informative, and as observed before, the expect risk function is dominated by the effective sample size. Thus, for the uniform-cost case, uniform allocation algorithms like random and round-robin has a very similar performance to the iterative greedy allocation and the adaptive algorithms. However, round-robin is not effective as the greedy algorithm. For example, after a certain number of

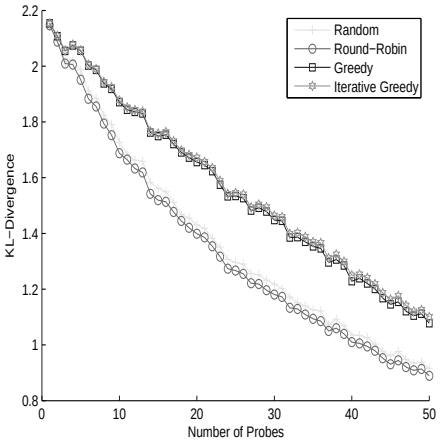


(a) Uniform-cost with good informative priors.

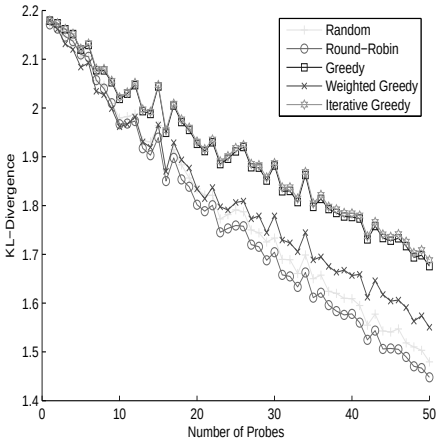


(b) Non-uniform-cost with good informative priors.

Figure 3.10: Uniform-cost and non-uniform-cost independent variable problem with good informative prior.



(a) Uniform-cost with bad informative priors.



(b) Non-uniform-cost with bad informative priors.

Figure 3.11: Uniform-cost and non-uniform-cost independent variable problem with bad informative prior.

probes starting from uniform priors of two variables, the learner may encounter a state $\{Beta(14,1), Beta(8,8)\}$ with a remaining budget of 1, a uniform sampling policy like round-robin might sample $Beta(14,1)$ which gives a $RD = 0.0016$, whereas the greedy learner would sample $Beta(8,8)$ and get a $RD = 0.0017$. Note that these cases rarely happen, which is the reason why in the randomized empirical study the difference is not pronounced.

Only the performance of greedy ,out of greedy and weighted greedy, is shown in Figure 3.9(a), Figure 3.10(a) and 3.11(a), because when the costs are uniform, greedy and weighted greedy are the same algorithm. However, for the non-uniform-cost case, the weighted greedy learner beats the other algorithms. This suggests that considering the costs is beneficial.

We also investigated the independent variable problem with good informative priors. Since the true parameters are generated from the priors, we believe that taking the priors into consideration will benefit our greedy-based learners. This is shown in Figure 3.10(a) and 3.10(a). Figure 3.10(a) is bumpier than the other graphs. This may result from the randomness when generating the good priors.

We experimented with the misleading bad priors. As foreseen in the previous section, greedy-based learners are trapped in these tests, and uniform sampling policies are preferred if the true priors are uniform distributions. This fact does not contradict the optimal allocation statement of the iterative greedy algorithm.

Finally, iterative greedy and greedy have almost the identical performance. The fact suggests that the benefit of adaptivity is limited. This is also observed by Dean [8] and Guha and Munagala [14] for other stochastic scheduling problems.

Due to the multiple degrees of randomness, the variances of the results are fairly large. However, by running the Wilcoxon signed rank test, we can verify that the performance comparison above are statistically significant. For example, in Figure 3.10(a) at 5% significance level, the Wilcoxon signed rank test fails to reject the null hypothesis of zero median in the difference between each pair of the greedy, the weighted greedy, and the iterative greedy allocation algorithm, but rejects the null hypothesis between round-robin and the greedy algorithm.

3.5.1 Summary

In general, the greedy-based learners are favored for tasks when we have a prior that it is not misleading. If we have a prior but are not certain of its quality, it is safer to use the round-robin algorithm. If our problem has different costs for variables, we had better take the costs into consideration by choosing a weighted greedy algorithm. The adaptivity is not crucial for the specific problem. Algorithms with and without adaptivity deliver a similar performance.

Chapter 4

Budgeted Parameter Estimation in General Bayesian Networks

4.1 Problem Formulation

In this section, we formally define the budgeted parameter estimation of Bayesian networks problem. Here, we assume:

1. *Correct Structure*: The structure $\mathcal{G}$ of the Bayesian network is given, and it is one of the correct graphical representations of the underlying distribution. We do not require the network structure to have all the true causal relationships between the variables, but we do need the given structure to be a Markov equivalent of the true causal structure. Since in the dissertation each variable of the underlying distribution corresponds to one and only one node in the Bayesian network, we use the term variable and node interchangeably, unless mentioned otherwise.

2. *Finite and Discrete Variables*: Each of the variables in the underlying distribution has a finite number of discrete values. This assumption simplifies our analysis and implementation, and enables us to utilize the Dirichlet distributions for parameter modeling.

3. *Parameter Independence*: As mentioned in the previous section, we make the parameter independence assumption [12], so that we can impose independent Dirichlet distribution to each parameter in the network. The parameter independence assumption $p(\boldsymbol{\theta}) = \prod_i \prod_{\boldsymbol{u}_i} p(\theta_{X_i|\boldsymbol{u}_i})$ is a combination of the global $p(\boldsymbol{\theta}) = \prod_i p(\boldsymbol{\theta}_i)$ and the local $p(\boldsymbol{\theta}_i) = \prod_{\boldsymbol{u}_i} p(\theta_{X_i|\boldsymbol{u}_i})$ parameter independence assumptions.

4. *Dirichlet Distributions*: Each of the parameters is Dirichlet distributed $p(\boldsymbol{\theta}_{X_i|\boldsymbol{u}_i}) = p(\theta_1, ...\theta_K) = Dirichlet(\alpha_1, ...\alpha_K) = \frac{\Gamma(\sum_{i=1}^K \alpha_k)}{\prod_{i=1}^K \Gamma(\alpha_k)} \prod_{i=1}^K \theta_k^{\alpha_k - 1}$. Because the Dirichlet distribution is the conjugate prior of the multinomial distribution, both the prior and the posterior given complete instances are Dirichlets.

4.1.1 Formal Model

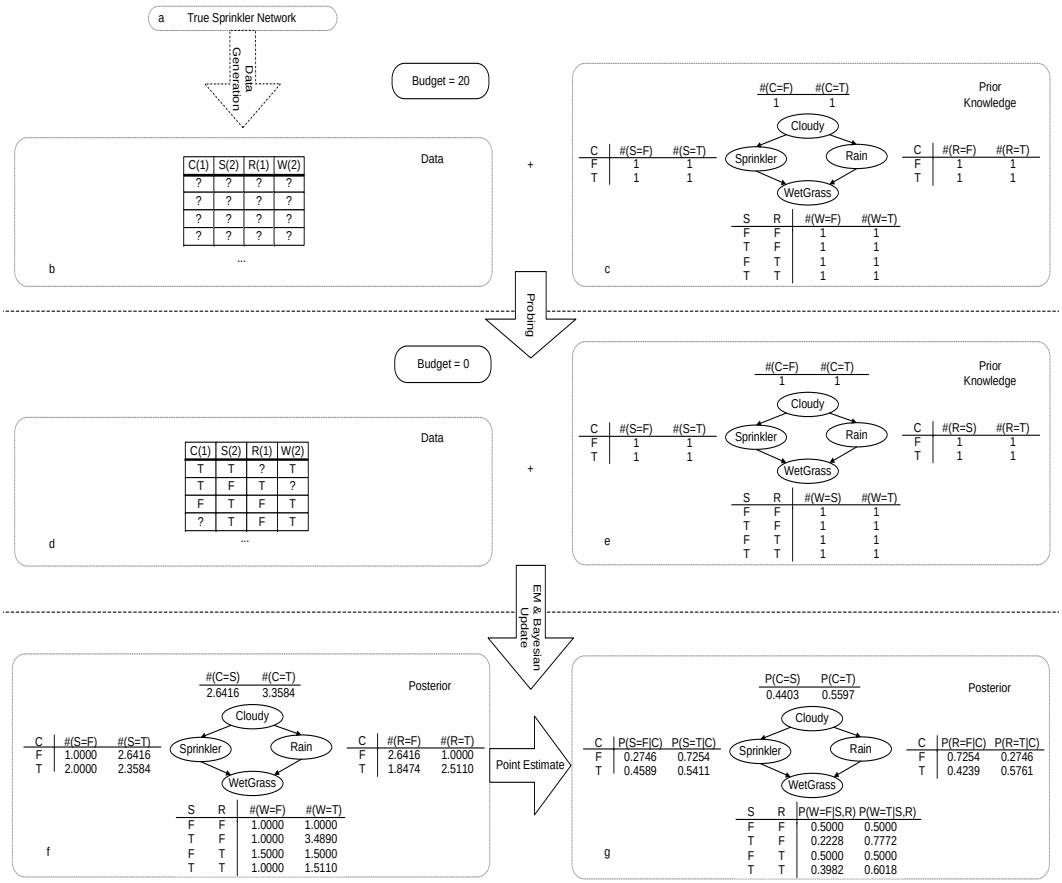


Figure 4.1: Budgeted parameter learning. In (a), a data pool generated from the underlying distribution (the true Sprinkler network) is shown in a table. Each row of the table corresponds to a data instance d^j, while each column corresponds to the values of a data attribute d_i. Initially, all the values of the data pool are unobserved (denoted by "?"s). The cost for observing value of a certain attribute is shown in the brackets here, *e.g.* $W(2)$ means that it costs "2" units of the budget to observe the variable W for a data instance. Beside the data is our prior knowledge, which is composed of the correct structure and priors (here, uniform). We use $\#(\cdot)$ to denote the parameters for the prior, *e.g.*, $\#(C = T) = 1$. The budgeted learner then makes decisions using the prior knowledge in (c) on which data value to buy, and continues probing entries in the table until the budget is exhausted, which yields the data in (d). The probed data, together with the prior, give the posterior estimate (f) of the network parameters after an update procedure. By taking the mean values of the posterior distributions, we get the point estimate (g).

In the budgeted learning setting, we start with the given structure $\mathcal{G}$ and a fixed budgeted $\mathcal{B} \in \mathbb{R}_{\geq 0}$. Moreover, we know it costs $c_i = c(X_i)$ to see the value of variable i for a certain data instance. At the start, no data is provided, and a strategy has to be employed to collect the data. Since a Bayesian approach is taken, we assume that some prior over the parameters is given. It is helpful to view the data set $\mathcal{D}$ as a growing matrix with each row corresponding to a data instance and each column corresponding to an attribute. We start with an empty matrix, where the values

of the cells can be achieved through probes. A probe is defined as a purchase of the value of the i-th attribute of the jth instance, cell d_i^j, at cost $c(X_i)$. A $probe_i^j$ probes the ith attribute of the jth instance. A probe can be applied to a previously probed instance or an un-probed instance [1]. A probe to an un-probed instance will increase the number of rows of $\mathcal{D}$ by one. The task of the learner is to make the probes wisely, so that the point estimate of posterior parameters $\langle \hat{\boldsymbol{\theta}} | \mathcal{D} \rangle$ updated by the probed data $\mathcal{D}$ is as close as possible to the underlying distribution. This budgeted parameter learning process is illustrated in Figure 4.1. In the example, we set the budget $\mathcal{B}$ to 20, and costs $c(C) = 1, c(S) = 2, c(R) = 1$, and $c(W) = 2$. Figure 4.1 shows that the budgeted learner interferes with the standard parameter learning procedure at the data collecting stage.

4.1.2 Loss Function and Risk Function

Recall that we chose KL-divergence as our loss function for our parameter estimation and expected KL-divergence as our risk function for distributions of parameters. Heckerman [17] and Tong [34] have shown that both of this two functions are decomposable over Bayesian networks.

Loss Function

The KL-divergence for measuring the final quality of the learned parameters can be written as [17]:

$$\mathrm{KL}(\boldsymbol{\theta} || \hat{\boldsymbol{\theta}}) = \sum_i \mathrm{KL}(\boldsymbol{\theta}_{X_i|\boldsymbol{U}_i} || \hat{\boldsymbol{\theta}}_{X_i|\boldsymbol{U}_i}), \tag{4.1}$$

where $\boldsymbol{\theta}$ denotes the true parameters, $\hat{\boldsymbol{\theta}}$ denotes our estimate, and each form of the right hand side sum, $\mathrm{KL}(\boldsymbol{\theta}_{X_i|\boldsymbol{U}_i} || \hat{\boldsymbol{\theta}}_{X_i|\boldsymbol{U}_i})$, is defined as the *conditional KL-divergence*:

$$\mathrm{KL}(\boldsymbol{\theta}_{X_i|\boldsymbol{U}_i} || \hat{\boldsymbol{\theta}}_{X_i|\boldsymbol{U}_i}) = \sum_{\boldsymbol{u}_i} P_{\boldsymbol{\theta}}(\boldsymbol{u}_i) \mathrm{KL}(\boldsymbol{\theta}_{X_i|\boldsymbol{u}_i} || \hat{\boldsymbol{\theta}}_{X_i|\boldsymbol{u}_i}). \tag{4.2}$$

Risk Function

The risk function for the estimated intermediate distributions of true parameters can be written as [34]:

$$\begin{aligned} Risk(p_{\boldsymbol{\theta}}) &= Exp[\mathrm{KL}(p_{\boldsymbol{\theta}})] = \int_{\boldsymbol{\theta}} \mathrm{KL}(\boldsymbol{\theta} || \hat{\boldsymbol{\theta}}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \sum_i \sum_{\boldsymbol{u}_i} P_{\hat{\boldsymbol{\theta}}}(\boldsymbol{u}_i) \sum_{k=1}^{K_i} \frac{\alpha_{x_i^k|\boldsymbol{u}_i}}{\sum_{j=1}^{K_i} \alpha_{x_i^j|\boldsymbol{u}_i}} \times (\Psi(\alpha_{x_i^k|\boldsymbol{u}_i} + 1) - \Psi(\sum_{j=1}^{K_i} \alpha_{x_i^j|\boldsymbol{u}_i} + 1)) + H(P(X_i|\boldsymbol{u}_i)). \end{aligned} \tag{4.3}$$

Unlike the independent variable problem in the previous chapter, both the KL-divergence and the risk function are not strongly decomposable for a general Bayesian network. They can only

[1] The budgeted learning problem is different from the interventional active learning problem [35, 27, 33] in that the interventional active learner fixes values of certain attributes (interventions) of an instance, finds the values of the remaining attributes, and uses the the whole instance without missing data to update the current belief. However, our budgeted learner uses probes to get values for cells, and uses the observed data to update the current belief.

be factorized to a *weighted* sum of the corresponding function of each variable, where the weights depend on the parent(s) of the variable. There is where the correlations between the variables take place. On the other hand, for the independent variable problem, the function value of the whole problem can be strongly decomposed into the sum of function values of each variable. Due to this correlation, we have to deal with a learning problem with missing data, unless the optimal option for the budgeted learner involves using full tuples.

4.1.3 Applicability of Imputation Methods for Missing Data

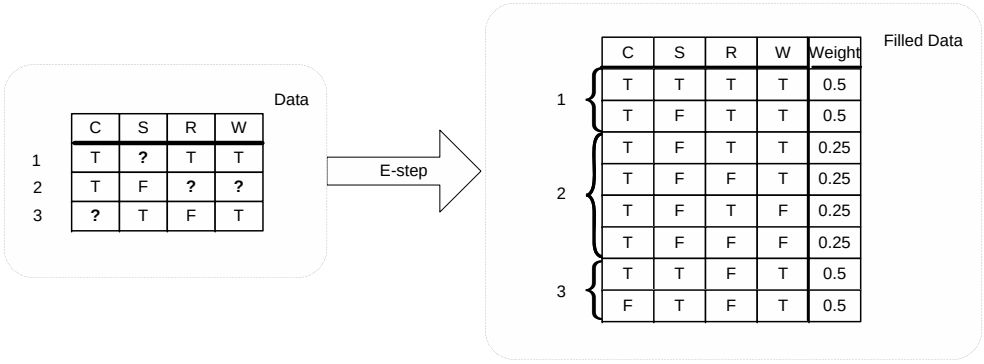


Figure 4.2: An example of the E-step of the EM algorithm. "?" denotes cells with missing values. Another column with weights is added to the data matrix after the E-step.

When training data is partially observed, we cannot apply the simple standard Bayesian update to get the posterior distribution. Imputation algorithms have been applied to learn the parameters from missing data in Bayesian networks. These algorithms all iterate between an imputation step, where the learner somehow fills the missing entries of the data using the current believes of the parameters and the partial data, and a update step, where the learner uses the imputed data to compute the new parameters. For example, we consider applying the Expectation Maximization (EM) algorithm, which is able to learn the model parameter in the presence of missing data, to our budgeted learning problem. It estimates model parameters and computes the expected sufficient statistics in turn. The algorithm starts with some parameter [2] $\boldsymbol{\theta}^0$, which can be generated randomly or chosen as the point estimate of the priors. The performance of EM algorithm might be highly sensitive to the choice of starting point.

In this section, we briefly describe an application of the EM algorithm to our budgeted parameter learning problem, and discuss why it is not very suitable for our problem. The algorithm alternates the following Expectation step (E-step) and Maximization step (M-step) until some stopping criterion like $\boldsymbol{\theta}^i \simeq \boldsymbol{\theta}^{i+1}$ or $\ell(\boldsymbol{\theta}^i|\mathcal{D}) \simeq \ell(\boldsymbol{\theta}^{i+1}|\mathcal{D})$ is met.

E-step Figure 4.2 shows an example of the E-step starting from the point estimate of uniform priors.

[2] $\boldsymbol{\theta}^i$ denotes the set of all parameters in the ith iteration.

The learner splits each partial data instance into a set of weighted complete data instances. Each missing entry of a split instance is filled with all possible assignments to the missing variables, *e.g.* In Figure 4.2, the first row of the left table $\langle$T, ?, T, T$\rangle$ becomes the first 2 rows of the right table $\langle$T, T, T, T$\rangle$ and $\langle$T, F, T, T$\rangle$. The total number of imputed instances for each data instance equals the number of all possible value assignments to the missing variables. A weight $P(Assignment|ObservedValues, \boldsymbol{\theta}^i)$ is then computed through an inference on the imputed instance using the current parameter and associated with the imputed instance, *e.g.* weights for the first 2 rows of the right table in Figure 4.2 are both 0.5, because of the uniform priors. After the E-step, we achieve a data set with no missing values and each instance is associated with a weight $\in (0, 1]$.

M-step In the M-step, the learner updates the hyper-parameters of the Dirichlet distributions using the imputed data. Instead of counting "1" as the sufficient statistics for each data instance, it counts the weight. Theoretical analysis show that this procedure guarantees the improvement of the log-likelihood function $\ell(\boldsymbol{\theta})$ [19].

There are several reasons why the EM algorithm, which is designed from an frequentist perspective, is not a good choice for the budgeted learning problem. The major problem is: We need the entire posterior distribution to compute the $Risk$s, but the above EM algorithm estimates the *mode* of the posterior distribution, treats the posteriors as unimodal distributions, and uses the expected sufficient statistics (the weights) to update the prior. When data are missing, the true posterior distribution is a mixture of Dirichlet distributions, which is not unimodal, and a Dirichlet with the mean value set to one mode of the mixture does not give a good approximation. Also, the EM algorithm only finds the local maxima. A good approximation of the global maxima requires multiple restarts or earlier pruning of unpromising starting points. Finally, the EM algorithm is computationally very expensive, due to multiple inferences. Other imputation algorithms, like Gradient Ascent and Gibbs Sampling, can also be applied, but they suffer the same problem of the EM algorithm. See Section 5.3 for a review.

4.2 Discussions on Budgeted Parameter learning in Bayesian Networks

In this section, we give several conjectures on the budgeted parameter learning problem in Bayesian networks.

4.2.1 Complete Bayesian Networks with BDe Priors and Uniform Costs

Definition 4.2.1. A complete Bayesian network is a Bayesian network whose structure is a DAG in which every node is connected to every other node in the DAG.

A graph with the above property is also called a *clique*.

Conjecture 4.2.2. *For complete Bayesian networks with BDe priors and uniform costs, when the budget $\mathcal{B}$ is a multiple of the number of variables N, uniform allocation algorithms like round-robin in Figure 3.5 that takes full tuples gives a posterior distribution with the minimum expected $Risk$.*

The conjecture says that if the learner starts with a complete Bayesian network and BDe priors, it probes the cells of the training data in a way that generates no missing data. Hooper proved that we can impose one Dirichlet prior for a complete Bayesian network with BDe priors [18]. Therefore, we can also view a complete Bayesian network with BDe priors as one multivariate variable in our independent variable problem. As a result, if the above conjecture holds and we are given a network that is composed of several independent cliques of subset of variables with BDe priors, we can run our greedy-based algorithms to determine the number of probes for each clique and take full tuples inside of each clique.

4.2.2 Connected Bayesian Networks with BDe Priors and Uniform Costs

We believe the above conjecture could be generalized to non-complete connected Bayesian networks. Consider a network with a structure $X \rightarrow Y$ where both X and Y are cliques composed of many nodes. Conjecture 4.2.2 implies that for X and Y separately , the learner should take full tuples to get a maximum RD. Since the structure $X \rightarrow Y$ is a clique itself, the learner should take full tuples for the whole network as well. In all, by recursively applying Conjecture 4.2.2, we believe that for connected Bayesian networks with BDe priors taking full tuples is optimal.

4.2.3 Connected Bayesian Networks with Non-BDe Priors and Uniform Costs

When the priors do not satisfy the BDe constraint, the problem becomes more complex. For example, given a network structure $X \rightarrow Y$ where both X and Y are binary variables, if the learner has a prior of a very small $Risk$ on X, *e.g.* $\boldsymbol{\theta}_X \sim Beta(999, 1)$, but two priors of large $Risk$s on Y, *e.g.* $\boldsymbol{\theta}_{Y|X=T} \sim Beta(1, 1)$ and $\boldsymbol{\theta}_{Y|X=F} \sim Beta(1, 1)$, it might want to keep probing Y for a while, because this will have a much larger RD on Y and the correlation between X (X is nearly deterministic) and Y can almost be inferred. However, when the values of the hyper-parameters change, the optimal solution becomes less obvious.

Chapter 5

Related Work

5.1 Interventional Active Learning of Generative Models in Bayesian Networks

Interventional active learning of a generative model has been studied in the context of Bayesian networks, albeit in an myopic approach [36, 35, 27, 33]. In this setting, the learner collects instances and updates its current state sequentially. For each instance, the learner is able to choose some data attributes and set them to certain values (this process is called a clamp or an intervention), and the rest of the attributes in that instance are sampled from the underlying distribution given the clamped variables. The goal of the learner is to make an estimation as close as possible to the underlying distribution measured by KL divergence with a minimal number of data instances.

The cost is measured differently in interventional active learning versus budgeted learning. Interventional active learning takes the number of data instances as the cost, while budgeted learning takes the number of probed data attributes times the corresponding attribute costs as the total cost.

In both the interventional active learning and budgeted learning problem, the learning agent is allowed to choose salient data that helps to learn a more accurate model with minimal cost instead of just passively observing data. However, one major difference between interventional active learning and budgeted learning is whether the learner is allowed to clamp the values of the chosen attributes. In interventional active learning, the learner has the power of intervention, whereas, in budgeted learning, the learner does not. Murphy asserts that this power of intervention is necessary for causal discovery, distinguishing two Markov equivalent network structures [27].

Another distinction is whether the learning agent is provided with fully observed data tuples or data tuples with observed values for only chosen attributes. In interventional active learning setting, full data tuples are observed. Since the chosen data attributes are forced to certain values by the learner, the other passive observational data attributes are crucial for the learner to learn the underlying distribution. The problem is that the clamped node values do not tell us anything about how often a value for a variable occurs in the real world. By contrast, budget learning agent takes only values of the chosen data attributes into consideration and treat all the other data attributes as

missing. As a result, an interventional active learning agent does not have to deal with missing data, while our budgeted learning agent does.

The third difference is whether they have an explicit budget. Interventional active learning has no explicit budget. Therefore, the objective of interventional active learning is typically to learn the parameters or the structure of the Bayesian network as fast/cheaply as possible [36, 35, 27, 33]. In other words, the active learning agent tries to reduce the need for training data. On the other hand, in budgeted learning, we try to make the purchasing as wisely as possible so as to learn a good model, subject to a firm budget. Thus, the active learning problem is attacked mostly greedily, whereas, people tend to use a deeper lookahead to address the budgeted learning problem.

Tong and Koller investigate interventional active learning for both parameters and structure in Bayesian networks [36, 35]. For parameter estimation, they choose expected KL divergence between the posterior distribution and the posterior point estimate of parameters as their loss function. They do a myopic search on all possible interventions, which they term queries, and pick the one that reduces the expected loss function the most. The parameter distributions of the network are then updated using standard Bayesian network update approach, except that the values of the clamped nodes and their parents are simply ignored. This is because the values of the clamped nodes and their parents are not drawn from the underlying distribution due to our intervention. The experimental results show that this greedy algorithm learns the parameters of the network faster than a random uniform-sampling algorithm. For structure learning, the same idea is applied. The loss function is defined as the summation of the entropies of each edge distribution, and a myopic search is employed to find the best query. The distribution over all possible graphs is then updated using the results by Cooper and Yoo [6]. To simplify the computation, Tong and Koller sample a set of ordering of nodes from the distribution over models using MCMC, and compute the averaged expected utility of the query over all the samples, and pick the best query based on the averages. Again, they show that the number of observations is reduced by employing the interventional active structure learning algorithm.

Murphy studies the same problem of learning of Bayesian models actively [27]. The difference between Murphy's algorithm and Tong and Koller's algorithm is that Murphy chooses the entropy of the whole model distribution as the loss function instead of the summation of all edge entropies, and he uses a MCMC on all possible DAGs instead of all possible node orderings.

Steck and Jaakkola investigate three different loss functions, namely expected KL divergence, backward expected KL divergence, and the sum of the two, on the same active learning task. They argue that the third loss function can be computationally more efficient when applied with their "Query by Committee" method to compute the loss of a distribution of models [33], and empirical results show that the resulting algorithm is effective for various problems.

Despite all the differences, research on interventional active learning, especially their formulation of loss functions, and the ideas to simplify the computation, sheds light on our budgeted learning

task, *e.g.* Tong and Koller, Murphy, and we all use the same risk function defined on the Dirichlet distributions to quantify the their qualities, while Steck and Jaakkola use a modified version of it.

5.2 Active Model Selection and Budgeted Learning for Classifiers

Active model selection and Budgeted Learning for Classifiers are studied by Madani *et al.* [26] and Lizotte *et al.* [23] respectively. Madani *et al.* simplify the active model selection problem to the "coins problem" [26]. They define the coins problem as a case where you are given a collection of independent coins with different priors, each coin is associated with a cost to flip and observe the outcome (head/tail), and a fixed budget is given. The objective of active model selection is to arrange the sequence of the coin flips wisely, so that after the budget is spent during the exploration phase, the learner is able to pick the coin with the highest head probability to exploit. The different objectives play an very important role based on our results and theirs.

Lizotte *et al.* extend the coins problem to learning naive Bayesian network classifiers, where a class node and a set of feature nodes are given. Each feature is associated with a cost and a prior. All the class values for each instance are known beforehand, together with a predetermined budget. The task is to build a good classifier after spending all the budget probing features corresponding to certain class values. Computing the optimal solution of these two problems are shown to be NP-hard [26, 14], so extensive heuristics are proposed and studied. Some of the policies can be briefly described as follows:

Optimal Build the whole decision tree using enumeration. Because the two problems are NP-hard [26, 14], the optimal policy is only computationally feasible in very limited scale.

Random Flip the coins or probe the features randomly.

Round-Robin Sequentially flip the coins or probe the features for each data tuple in order.

Greedy Compute the action with the best expected utility with one step lookahead, and take the best action.

Single Coin Lookahead Compute the expected utility for allocating all the remaining budget for one coin or feature, flip the best coin or probe the best feature once, and keeping doing this until the budget is exhausted.

Gittins Index Compute the gittins index for each coin/feature, and flip the coin or probe the feature with the highest gittins index in turn [13].

Madani *et al.* [26, 23] show that policies like Biased Robin and Single Coin Lookahead work pretty well in practice for problem with unit costs. In this paper, we study the budgeted learning for generative models instead of discriminative models, but some of the policies are adapted from or

inspired by the above policies. We define similar simpler independent variable problem to motivate our study and then apply the techniques to more complex Bayesian networks. However, definitions of the Active Model Selection and Budgeted Learning for Classifiers avoids problem of learning with missing data (see below).

5.3 Learning with Missing Data

After the probing process, the learner acquires values for the probed data while all the other data entries are left as missing. Hence we must deal with the learning parameters in the presence of incomplete data. Three ways of parameter estimation in Bayesian networks with missing data are well studied [17]: gradient ascent, Gibbs sampling, and Expectation-Maximization (EM). Note that all of these algorithms assume that the missing data mechanism can be "ignored" as defined by Rubin [29]. The incomplete data set that we have to face satisfies Rubin's condition, "...the probability that some component is missing may depend on observed components, but not on unobserved components" [28].

The gradient of the parameters of a Bayesian network given the partially observed data can be computed analytically [30]. With the constraint that the parameters of the network describe a legal probability distribution, we can then use a standard conjugate gradient ascent procedure to optimize the parameters.

The Gibbs sampling method deals with the missing data in a different way. It first fills the missing entries by drawing samples from the current estimate of the distribution, and then updates the parameters using the filled complete data set. This process is then iterated utill the parameters converge [17].

Lauritzen [22] applied the EM algorithm proposed by Dempster *et al.* [9] to Bayesian networks. The EM algorithm contains two steps that are similar to the Gibbs sampling process as discussed in Section 4.1.3.

In Section 4.1.3, we describe an EM algorithm for our budgeted parameter learning task, and gave the reasons why it is not quite applicable. The other two methods are also designed from a frequentist perspective, and share the same imputation properties of the EM algorithm.

Our work is also related to the on-line parameter estimation problem studied by Bauer *et al.* [2], which involves learning from missing data. Bauer *et al.* proposed a parameterized version of the EM algorithm and show that by carefully selecting the weights, the parameterized EM out-performs the standard EM as it converges faster. An on-line parameter estimator updates the parameters after observing one data sample at a time. This is very similar to our learner, who would "think", "probe", "observe", and "update" for one data attribute at a time. However, there are two key differences between the on-line learner and our budgeted learner. First, an important limitation of the on-line learner is that it can not store any of the past data. So the on-line learner can not save all the data and run a batch parameter estimation algorithm. On the contrary, the budgeted learner keeps track of all

the probed data, and is able to update its state using the new data and the whole data pool collected. Second, by probing a new attribute of a previously probed data instance, the budgeted learner not only has a new data instance but also loses an old one.

5.4 The Value of Information Problem

The budgeted parameter learning problem is different from the value of information problem studied by Krause and Guestrin [20, 21]. The value of information problem is defined as the problem where both the structure and the parameters of a graphical model is given together with a budget; and the task is to select a subset of variables to observe where each variable is associated with a cost, so that the residual entropy of the network is minimized. Note that there is no real learning in the value of information problem. As a result, the hardness results and approximation algorithms do not apply to our budgeted parameter learning problem.

5.5 Stochastic Scheduling

Our research of budgeted parameter learning in Bayesian networks is related to the work of stochastic scheduling in the area of both algorithm, and operations research. Stochastic scheduling problem can be viewed as a knapsack problem with nondeterministic costs. In his dissertation [8], Dean studies stochastic scheduling problems under hard deadline constraints. The basic formulation of the problem is: We are given a set of jobs, where each job is associated with a known distribution of processing time and a value, and all the jobs share the same start/completion deadline. The task is to find the scheduling policy that maximizes the total value of successfully scheduled jobs. Dean studies a weighted shortest expected processing time policy (WSEPT), which basically ranks the jobs by their weights (value divided by the expected processing time), and schedules the job with the highest weight first. Using linear programming (LP), Dean shows that in the worst case scenario this policy delivers a performance that is a quarter of the optimal solution (4 approximation) for the start deadline model.

Guha and Munagala extend the same idea to solve the active model selection/budgeted learning for classifier problem, and present a 4-approximation algorithm [14]. The algorithm first builds a DAG for each of the coins. The root of each DAG is the original state of a coin, which is represented by the sufficient statistics of the beta distribution for that coin. The out degree of each node in the DAG is the number of possible values for that variable (for the coins problem, this number is 2). Each node in the DAG is a possible outcome by flipping the corresponding coin certain number of times, *e.g.* the nodes at the third level of the DAG for a Beta(1, 1) coin would be Beta(3, 1), Beta(2, 2), and Beta(1, 3). Each node of the DAGs is then associated with 3 values: a probability to reach that node, a probability to stop at that node and pick the coin as the best, and a probability to keep flipping that coin. A LP is then formulated to maximize the total reward when certain coins are chosen for

exploitation, with the constraint that the total cost never exceeds the budget and the total probability for the coins to be picked for exploitation never exceeds 1. With the solution of the LP, Guha and Munagala then employ the idea by Dean, and view the problem as a two-constraint stochastic packing problem. They show that by applying the same WSEPT policy, we could achieve a gain at least 1/4 of the optimal policy. Even though the approximation gives the worst case guarantee, it dose not out-perform the heuristics, like biased robin, by Madani *et al.* [26]. Note that this approach only works for the problem where the coins are independent from each other.

Another relevant research area in stochastic optimization is the bandit problem. The gittins index policy in the active model selection and budgeted learning for classifier section is due to the result by Gittins [13] for the classic bandit problem. There are three subtle differences between the classic bandit problem and the problem studied by Madani *et al.* [26]. Firstly, the classic bandit problem assumes that rewards are collected during the exploration phase; secondly, there is a discount associated with discrete time steps, which basically makes the learner puts more weights on the nearer rewards; finally, the classical bandit problem is defined with an infinite time horizon. Schneider and Moore [32] investigate a classic bandit problem with finite exploration time (a budget), and show that by properly converting the total rewards from the infinite horizon with a temporal discount to a budgeted case, gittins index policy works reasonably well as a heuristic.

The classic bandit problem has been generalized by Whittle to restless bandits [37], where the agent is able to operate on multiple bandits (flip multiple coins), and the non-activated bandits can still change states with passive rewards. Heuristics [3] and approximation algorithms [15] are developed. The restless bandits problem is related to our budgeted parameter learning task in general Bayesian networks, because when the variables are correlated, by probing one variable the state of the other variables may change. Some of their results might be applicable to the budgeted learning problem. We leave this as future work.

Chapter 6

Conclusions

6.1 Future Work

The first extension is to derive some theoretical hardness results on the adaptive *uniform-cost* independent variable problem. We believe the problem is at least NP-hard, but could not reduce it to a hard problem so far. Another related extension is to show performance bounds for the algorithms that we studied for the independent variable problem, especially for the iterative greedy allocation and the adaptive greedy algorithm. We also hope to quantify the performance effects of adaptivity by the *adaptivity gap* proposed by Dean [8].

For the budgeted parameter learning problem for general Bayesian networks, an immediate extension is to prove the conjectures that we proposed. This might be achieved through doing an accurate analysis of the $Risk$ reduction of different probe patterns for Bayesian networks with different structures and priors. Also, to derive an efficient algorithm for the budgeted learning problem in general Bayesian networks, we need an effective way to approximate the posterior distributions, which are mixtures of Dirichlets. This requires learning the parameters of a model with missing data using a Bayesian approach. For policies, we hope to adapt heuristic [37, 3] and approximation [15] algorithms for the restless bandit problem to our task.

Finally, the proposed loss function and $Risk$ function use the point estimate of the posterior distribution as the final estimate. A different problem may be formulated using a pure Bayesian approach, where the learner take the whole posterior distribution as the estimate instead of taking a point estimate. In this setting, the learner needs to measure the information for each posterior distribution. We could define an information gain as the KL divergence between a pair of prior and posterior distribution, and then to solve this new problem, we need to maximize this information gain. This approach have been studied by Mackey for discriminative active learning problems [24].

6.2 Contributions

In this dissertation, we have studied the budgeted parameter learning problem for generative models. We have formally formulated the problem, and shown that it is NP-hard to solve even a very

restricted independent variable problem optimally. We proposed the iterative greedy allocation algorithm, and show that, for the uniform-cost independent variable problem, it is an optimal allocation algorithm using the monotonicity and convexity of the defined risk reduction. In empirical studies, we compared both our greedy-based allocation and adaptive policies with two uniform allocation baseline policies (round-robin, random), and observed that learning algorithms that utilize the correct prior information and consider different costs of probing different variables generally performs better; and the adaptivity does not significantly affects algorithms' performances. Empirical results on synthesized data show that our greedy based algorithms work well for problems with no misleading priors.

We also extended our budgeted learning formulation to parameter learning problems of general Bayesian networks. We have pointed out that learning with missing data is a main issue for solving the budgeted parameter learning problem in general Bayesian networks, and proposed conjectures on problems with certain restrictions. Finally, we have reviewed research related to our budgeted learning problem, and propose several future directions that may be interesting in the budgeted learning setting.

Bibliography

[1] Density estimation and connections to kl divergence. 2008. URL:http://isites.harvard.edu/fs/docs/icb.topic251888.files/Lecture03.pdf.

[2] Eric Bauer, Daphne Koller, and Yoram Singer. Update rules for parameter estimation in bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, 1997.

[3] Dimitris Bertsimas and José Nino-Mora. Restless bandits, linear programming relaxations, and a primal-dual index heuristic. *Operations Research*, 2000.

[4] John Breese, David Heckerman, and Koos Rommelse. Troubleshooting under uncertainty. Technical Report MSR-TR-94-07, Microsoft Research, Redmond, Washington, 1994.

[5] Vincent Conitzer and Tuomas Sandholm. Definition of complexity of some basic metareasoning problems. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

[6] Gregory F. Cooper and Changwon Yoo. Causal discovery from a mixture of experimental and observational data. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[7] Thomas Cover and Joy Thomas. *Elements of Information Theory 2nd Edition*. Wiley-Interscience, 2006.

[8] Brian Dean. *Approximation Algorithms for stochastic scheduing problems*. PhD thesis, MIT, 2005.

[9] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 1977.

[10] Thomas Ferguson. *Game Theory*. URL:http://www.math.ucla.edu/~tom/Game_Theory/Contents.html.

[11] Michael Garey and David Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W.H. Freeman & Company, 1979.

[12] Dan Geiger and David Heckerman. A Characterization of the Dirichlet Distribution Through Global and Local Independence. Technical Report MSR-TR-94-16, November 1994.

[13] John Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society*, 1979.

[14] Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *ACM Symposium on Theory of Computing*, 2007.

[15] Sudipto Guha, Kamesh Munagala, and Peng Shi. On index policies for restless bandit problems. 2008. URL:http://arxiv.org/abs/0711.3861.

[16] David Heckerman. An empirical comparison of three inference methods. In *Proceedings of the Fourth Conference on Uncertainty in Artificial Intelligence*, 1988.

[17] David Heckerman. A tutorial on learning with Bayesian networks. In Michael Jordan, editor, *Learning in Graphical Models*. MIT Press, 1999.

[18] Peter Hooper. Exact distribution theory for belief net responses. *Bayesian Analysis*, 3, 2008.

[19] Daphne Koller and Nir Friedman. *Structured Probabilistic Models*. 2005. Unpublished.

[20] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopc value of information in graphical models. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence*.

[21] Andreas Krause and Carlos Guestrin. Optimal nonmyopic value of information in graphical models - efficient algorithms and theoretical limits. In *Proceedings of the Nineteenth International Joint Conference on Artifical Intelligence*, 2005.

[22] Steffen Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 1995.

[23] Dan Lizotte, Omid Madani, and Russ Greiner. Budgeted learning of Naive-Bayes classifiers. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, 2003.

[24] David Mackay. Information-based objective functions for active data selection. *Neural Computation*, 4, 1992.

[25] David Mackay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.

[26] Omid Madani, Dan Lizotte, and Russ Greiner. Active model selection. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, 2004.

[27] Kevin Murphy. Active learning of causal Bayes net structure. Technical report, EECS Department, University of California, Berkeley, 2001.

[28] Carsten Riggelsen and Ad Feelders. Learning Bayesian network models from incomplete data using importance sampling. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[29] Donald Rubin. Inference and missing data. *Biometrika*, 1976.

[30] S.J. Russell, J. Binder, D. Koller, and K. Kanazawa. Local learning in probabilistic networks with hidden variables. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995.

[31] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2002.

[32] Jeff Schneider and Andrew Moore. Active learning in discrete input spaces. In *Proceedings of the 34th Interface Symposium*, 2002.

[33] Harald Steck and Tommi Jaakkola. Unsupervised active learning in large domains. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 2002.

[34] Simon Tong. *Active Learning: Theory and Application*. PhD thesis, Stanford University, 2001.

[35] Simon Tong and Daphne Koller. Active learning for parameter estimation in Bayesian networks. In *Advances in Neural Infomation Processing Systems*, 2001.

[36] Simon Tong and Daphne Koller. Active learning for structure in Bayesian networks. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 2001.

[37] Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 1988.

Appendix A

Proofs

In this section, we provide proofs for some of the lemmas and theorems.

We shall frequently use Gibbs inequality [25]:

$$-\sum_{k=1}^{n} p_k \ln p_k \leq -\sum_{k=1}^{n} p_k \ln q_k \tag{A.1}$$

with equality if and only if $p_k = q_k$ for all k for two probability distributions.

We shall also use the following inequality:

$$\frac{x}{x+1} < \ln(1+x) < x \tag{A.2}$$

for all $x > -1$ and $x \neq 0$.

Lemma 3.2.4 (used in page 15) *For $f(t) = \frac{1}{4^{2t}}$ where $t \in \mathbb{Z}_{\geq 2}$,*

$$4 \times H(p_{f(t+1)}) < H(p_{f(t)}), \tag{A.3}$$

where $H(p_{f(t)}) = -f(t)\ln f(t) - (1-f(t))\ln(1-f(t))$.

Proof. See Appendix A.

$$\begin{aligned}
& H(p_{f(t)}) - 4 \times H(p_{f(t+1)}) \\
&= 4 \times \left[\frac{1}{4^{2t+2}} \ln\left(\frac{1}{4^{2t+2}}\right) + \left(1 - \frac{1}{4^{2t+2}}\right) \ln\left(1 - \frac{1}{4^{2t+2}}\right)\right] \\
&\quad - \left[\frac{1}{4^{2t}} \ln\left(\frac{1}{4^{2t}}\right) + \left(1 - \frac{1}{4^{2t}}\right) \ln\left(1 - \frac{1}{4^{2t}}\right)\right] \qquad \text{(A.4)} \\
&> 4 \times \left[\frac{1}{4^{2t+2}} \ln\left(\frac{1}{4^{2t}}\right) + \left(1 - \frac{1}{4^{2t+2}}\right) \ln\left(1 - \frac{1}{4^{2t}}\right)\right] \\
&\quad - \left[\frac{1}{4^{2t}} \ln\left(\frac{1}{4^{2t}}\right) + \left(1 - \frac{1}{4^{2t}}\right) \ln\left(1 - \frac{1}{4^{2t}}\right)\right] \qquad \text{(A.5)} \\
&= \frac{12t}{4^{2t+1}} \ln 2 + \left(3 + \frac{3}{4^{2t+1}}\right) \ln\left(1 - \frac{1}{4^{2t}}\right) \\
&> \frac{12t}{4^{2t+1}} \ln 2 + \left(3 + \frac{3}{4^{2t+1}}\right) \times \frac{-1}{4^{2t} - 1}, \qquad \text{(A.6)}
\end{aligned}$$

where Inequality A.5 follows from Gibbs inequality and Inequality A.6 follows from Inequality A.2.

Define

$$\begin{aligned} r(t) &= 4^{2t+1}\left[\frac{12t}{4^{2t+1}}\ln 2 + (3+\frac{3}{4^{2t+1}}) \times \frac{-1}{4^{2t}-1}\right] \\ &= 12t\ln 2 + (3+3\times 4^{2t+1}) \times \frac{-1}{4^{2t}-1}. \end{aligned} \tag{A.7}$$

Taking the derivative of $r(t)$ yields

$$\frac{d(r(t))}{dt} = 12\ln 2 + \frac{30\times 4^{2t+1}\ln 2}{(4^{2t}-1)^2},$$

which is larger than 0 for $t > 0$. Also, when $t = 2$, $r(t) = 4.5767 > 0$. Thus, $H(p_{f(t)}) - 4 \times H(p_{f(t+1)}) > 0$ when $t \in \mathbb{Z}_{\geq 2}$. This proves the lemma. □

Theorem 3.3.2 (used in page 18) *The expected risk function for a variable with* $prior = Beta(\alpha, \beta)$, $\alpha, \beta \in \mathbb{R}_{>0}$ *is strictly monotonically decreasing in the number of probes* b *and is convex.*

Proof. To prove the expected risk function is strictly monotonically decreasing in b, we need to show:

$$ExpRisk_{prior}(b) > ExpRisk_{prior}(b+1). \tag{A.8}$$

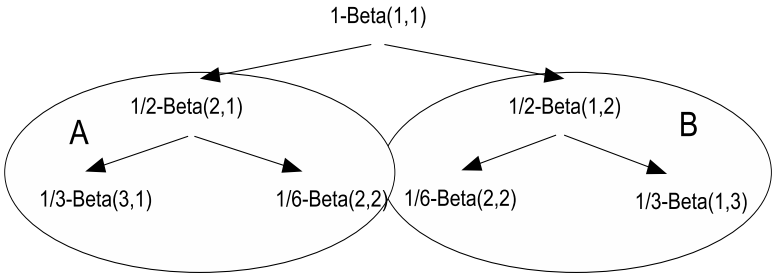


Figure A.1: A tree structure showing the outcomes of two probes to a variable with a prior of $Beta(1,1)$. If we can show the risk of the first level nodes in A and B are bigger than the expected risks of the second level nodes respectively, the weighted sum of the risks of the first level nodes is certainly bigger than the weighted sum of the expected risks of the second level nodes. This tree structure shows another way to compute the expected risk.

Due to the structure of the problem presented in Figure A.1, we only need to show

$$Risk(prior) = ExpRisk_{prior}(0) > ExpRisk_{prior}(1), \tag{A.9}$$

for all priors.

Tong demonstrates that [34]:

$$
\begin{aligned}
& ExpRisk_{prior_i}(0) - ExpRisk_{prior_i}(1) \\
= \;& H(\frac{\alpha}{\alpha+\beta}, \frac{\beta}{\alpha+\beta}) \\
& - \left(\frac{\alpha}{\alpha+\beta} H(\frac{\alpha+1}{\alpha+\beta+1}, \frac{\beta}{\alpha+\beta+1}) + \frac{\beta}{\alpha+\beta} H(\frac{\alpha}{\alpha+\beta+1}, \frac{\beta+1}{\alpha+\beta+1}) \right).
\end{aligned} \tag{A.10}
$$

We need to prove that Equation A.10 is greater than zero [1]. Applying Gibbs inequality to Equation A.10 yields:

$$
\begin{aligned}
& ExpRisk_{prior}(0) - ExpRisk_{prior}(1) \\
= \;& -\frac{\alpha}{\alpha+\beta}\ln\frac{\alpha}{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\ln\frac{\beta}{\alpha+\beta} \\
& - \left[\frac{\alpha}{\alpha+\beta}(-\frac{\alpha+1}{\alpha+\beta+1}\ln\frac{\alpha+1}{\alpha+\beta+1} - \frac{\beta}{\alpha+\beta+1}\ln\frac{\beta}{\alpha+\beta+1}) \right. \\
& \left. + \frac{\beta}{\alpha+\beta}(-\frac{\alpha}{\alpha+\beta+1}\ln\frac{\alpha}{\alpha+\beta+1} - \frac{\beta+1}{\alpha+\beta+1}\ln\frac{\beta+1}{\alpha+\beta+1}) \right] \\
\geq \;& -\frac{\alpha}{\alpha+\beta}\ln\frac{\alpha}{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\ln\frac{\beta}{\alpha+\beta} \\
& - \left[\frac{\alpha}{\alpha+\beta}(-\frac{\alpha+1}{\alpha+\beta+1}\ln\frac{\alpha}{\alpha+\beta} - \frac{\beta}{\alpha+\beta+1}\ln\frac{\beta}{\alpha+\beta}) \right. \\
& \left. + \frac{\beta}{\alpha+\beta}(-\frac{\alpha}{\alpha+\beta+1}\ln\frac{\alpha}{\alpha+\beta} - \frac{\beta+1}{\alpha+\beta+1}\ln\frac{\beta}{\alpha+\beta}) \right] \\
= \;& -\frac{\alpha}{\alpha+\beta}\ln\frac{\alpha}{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\ln\frac{\beta}{\alpha+\beta} \\
& - \left[\frac{\alpha}{\alpha+\beta}\left(-(\frac{\alpha+1}{\alpha+\beta+1} + \frac{\beta}{\alpha+\beta+1})\ln\frac{\alpha}{\alpha+\beta}\right) \right. \\
& \left. + \frac{\beta}{\alpha+\beta}\left(-(\frac{\beta+1}{\alpha+\beta+1} + \frac{\alpha}{\alpha+\beta+1})\ln\frac{\beta}{\alpha+\beta}\right) \right] \\
= \;& -\frac{\alpha}{\alpha+\beta}\ln\frac{\alpha}{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\ln\frac{\beta}{\alpha+\beta} \\
& -(-\frac{\alpha}{\alpha+\beta}\ln\frac{\alpha}{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\ln\frac{\beta}{\alpha+\beta}) \\
= \;& 0
\end{aligned} \tag{A.11}
$$

Because Inequality A.11 is built on Inequality A.1, moreover $p_k = \frac{\beta}{\alpha+\beta+1} \neq q_k = \frac{\beta}{\alpha+\beta}$ for $\alpha, \beta \in \mathbb{R}_{\geq 1}$, the equality never holds. This proves the strictly monotonically decreasing part.

To prove the convexity, we need to show:

$$
ExpRisk_{prior}(b) - ExpRisk_{prior}(b+1) > ExpRisk_{prior}(b+1) - ExpRisk_{prior}(b+2). \tag{A.12}
$$

Use the same argument from the preceeding proof, we only need to prove:

$$
ExpRisk_{prior}(0) - ExpRisk_{prior}(1) > ExpRisk_{prior}(1) - ExpRisk_{prior}(2). \tag{A.13}
$$

[1] If we used *Log loss* $LL(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}) = -\sum_x P_{\boldsymbol{\theta}}(X)\ln P_{\hat{\boldsymbol{\theta}}}(X)$ as the loss function, the difference would remain the same [27, 34]. In other words, the expected risk reduction defined in the dissertation is the same as the expected entropy reduction of the parameters.

The left side of the inequality equals:

$$
\begin{aligned}
&ExpRisk_{prior_i}(0) - ExpRisk_{prior_i}(1) \\
&= -\frac{\alpha}{\alpha+\beta}\ln\frac{\alpha}{\alpha+\beta} - \frac{\beta}{\alpha+\beta}\ln\frac{\beta}{\alpha+\beta} \\
&\quad - \Bigg[\frac{\alpha}{\alpha+\beta}(-\frac{\alpha+1}{\alpha+\beta+1}\ln\frac{\alpha+1}{\alpha+\beta+1} - \frac{\beta}{\alpha+\beta+1}\ln\frac{\beta}{\alpha+\beta+1}) \\
&\quad + \frac{\beta}{\alpha+\beta}(-\frac{\alpha}{\alpha+\beta+1}\ln\frac{\alpha}{\alpha+\beta+1} - \frac{\beta+1}{\alpha+\beta+1}\ln\frac{\beta+1}{\alpha+\beta+1})\Bigg] \\
&= -\Bigg[\frac{\alpha}{\alpha+\beta}(\frac{\alpha+1}{\alpha+\beta+1}\ln\frac{\alpha}{\alpha+\beta} + \frac{\beta}{\alpha+\beta+1}\ln\frac{\alpha}{\alpha+\beta}) \\
&\quad + \frac{\beta}{\alpha+\beta}(\frac{\alpha}{\alpha+\beta+1}\ln\frac{\beta}{\alpha+\beta} + \frac{\beta+1}{\alpha+\beta+1}\ln\frac{\beta}{\alpha+\beta})\Bigg] \\
&\quad - \Bigg[\frac{\alpha}{\alpha+\beta}(-\frac{\alpha+1}{\alpha+\beta+1}\ln\frac{\alpha+1}{\alpha+\beta+1} - \frac{\beta}{\alpha+\beta+1}\ln\frac{\beta}{\alpha+\beta+1}) \\
&\quad + \frac{\beta}{\alpha+\beta}(-\frac{\alpha}{\alpha+\beta+1}\ln\frac{\alpha}{\alpha+\beta+1} - \frac{\beta+1}{\alpha+\beta+1}\ln\frac{\beta+1}{\alpha+\beta+1})\Bigg] \\
&= -\Bigg[\frac{\alpha}{\alpha+\beta}\left(\frac{\alpha+1}{\alpha+\beta+1}\ln\frac{\alpha(\alpha+\beta+1)}{(\alpha+1)(\alpha+\beta)} + \frac{\beta}{\alpha+\beta+1}\ln\frac{\alpha+\beta+1}{\alpha+\beta}\right) \\
&\quad + \frac{\beta}{\alpha+\beta}\left(\frac{\alpha}{\alpha+\beta+1}\ln\frac{\alpha+\beta+1}{\alpha+\beta} + \frac{\beta+1}{\alpha+\beta+1}\ln\frac{\beta(\alpha+\beta+1)}{(\beta+1)(\alpha+\beta)}\right)\Bigg] \qquad \text{(A.14)}
\end{aligned}
$$

Use a similar approach, and we have:

$$
\begin{aligned}
&ExpRisk_{prior_i}(1) - ExpRisk_{prior_i}(2) \\
&= -\Bigg[\frac{\alpha}{\alpha+\beta}\left(\frac{\alpha+1}{\alpha+\beta+1}(\frac{\alpha+2}{\alpha+\beta+2}\ln\frac{(\alpha+1)(\alpha+\beta+2)}{(\alpha+2)(\alpha+\beta+1)} + \frac{\beta}{\alpha+\beta+2}\ln\frac{\alpha+\beta+2}{\alpha+\beta+1}) \right.\\
&\quad \left. + \frac{\beta}{\alpha+\beta+1}(\frac{\alpha+1}{\alpha+\beta+2}\ln\frac{\alpha+\beta+2}{\alpha+\beta+1} + \frac{\beta+1}{\alpha+\beta+2}\ln\frac{\beta(\alpha+\beta+2)}{(\beta+1)(\alpha+\beta+1)})\right) \\
&\quad + \frac{\beta}{\alpha+\beta}\left(\frac{\alpha}{\alpha+\beta+1}(\frac{\alpha+1}{\alpha+\beta+2}\ln\frac{\alpha(\alpha+\beta+2)}{(\alpha+1)(\alpha+\beta+1)} + \frac{\beta+1}{\alpha+\beta+2}\ln\frac{\alpha+\beta+2}{\alpha+\beta+1}) \right.\\
&\quad \left. + \frac{\beta+1}{\alpha+\beta+1}(\frac{\alpha}{\alpha+\beta+2}\ln\frac{\alpha+\beta+2}{\alpha+\beta+1} + \frac{\beta+2}{\alpha+\beta+2}\ln\frac{(\beta+1)(\alpha+\beta+2)}{(\beta+2)(\alpha+\beta+1)})\right)\Bigg] \qquad \text{(A.15)}
\end{aligned}
$$

Subtracting Equation A.15 from Equation A.14 yields:

$$
\begin{aligned}
&ExpRisk_{prior_i}(0) + ExpRisk_{prior_i}(2) - 2ExpRisk_{prior_i}(1) \\
&= \frac{\alpha}{\alpha+\beta}\times\frac{\alpha+1}{\alpha+\beta+1}\times\frac{\alpha+2}{\alpha+\beta+2}\times\ln(\frac{\alpha+1}{\alpha}\times\frac{\alpha+1}{\alpha+2}) \\
&\quad + \frac{\beta}{\alpha+\beta}\times\frac{\beta+1}{\alpha+\beta+1}\times\frac{\beta+2}{\alpha+\beta+2}\times\ln(\frac{\beta+1}{\beta}\times\frac{\beta+1}{\beta+2}) \\
&\quad - \ln\frac{(\alpha+\beta+1)(\alpha+\beta+1)}{(\alpha+\beta)(\alpha+\beta+2)} \qquad \text{(A.16)}
\end{aligned}
$$

Applying Inequality A.2 yields

$$\begin{aligned}
& ExpRisk_{prior_i}(b) + ExpRisk_{prior_i}(b+2) - 2ExpRisk_{prior_i}(b+1) \\
> \quad & \frac{\alpha}{\alpha+\beta} \times \frac{\alpha+1}{\alpha+\beta+1} \times \frac{\alpha+2}{\alpha+\beta+2} \times \frac{1}{(\alpha+1)^2} \\
& + \frac{\beta}{\alpha+\beta} \times \frac{\beta+1}{\alpha+\beta+1} \times \frac{\beta+2}{\alpha+\beta+2} \times \frac{1}{(\beta+1)^2} \\
& - \frac{1}{(\alpha+\beta)(\alpha+\beta+2)} \\
= \quad & \frac{1}{(\alpha+\beta)(\alpha+\beta+2)} \times \frac{\alpha\beta-1}{(\alpha+\beta+1)(\alpha+1)(\beta+1)} \\
\geq \quad & 0
\end{aligned} \tag{A.17}$$

for $\alpha, \beta \in \mathbb{R}_{\geq 1}$. This proves the convexity. □

Lemma 3.3.6 (used in page 23) *For uniform-cost binary variables with the same effective sample size $\boldsymbol{\theta}_i \sim Beta(\alpha_i, \beta_i)$ when $\alpha_i + \beta_i$ is constant, the greedy learner prefers variables with more balanced parameter distributions. More precisely, if $\alpha_i + \beta_i = \alpha_j + \beta_j$ and $|\alpha_i - \beta_i| < |\alpha_j - \beta_j|$ for variable i and j, the greedy algorithm prefers variable i.*

Proof. As the $Risk$ reduction shown in Equation A.10 is a sum of logarithms, it is differentiable for $\alpha, \beta \in \mathbb{Z}_{>0}$. Let $s = \alpha + \beta$, which is a constant. Taking the derivative *w.r.t* α, we have:

$$\frac{\partial RD(1)}{\partial \alpha} = \frac{\ln \frac{s-\alpha}{s}}{s} - \frac{\ln \frac{\alpha}{s}}{s} - \frac{\alpha}{s}\left(\frac{\ln \frac{s-\alpha}{s+1}}{s+1} - \frac{\ln \frac{\alpha-1}{s+1}}{s+1}\right) - \frac{s-\alpha}{s}\left(\frac{\ln \frac{s-\alpha+1}{s+1}}{s+1} - \frac{\ln \frac{\alpha}{s+1}}{s+1}\right). \tag{A.18}$$

Multiplying Equation A.18 with $s(s+1)$ then applying some algebraic manipulation, we obtain:

$$s(s+1)\frac{\partial RD(1)}{\partial \alpha} = \ln \frac{s-\alpha}{\alpha} + \alpha \ln \frac{\alpha+1}{\alpha} - (s-\alpha) \ln \frac{s-\alpha+1}{s-\alpha}. \tag{A.19}$$

For Lemma 3.3.6 to hold, we need to show that if α takes an integer value in $(0, \frac{s}{2})$, Equation A.19 is greater than zero; if α takes an integer value in $(\frac{s}{2}, s)$, Equation A.19 is smaller than zero. By symmetry, we only need to prove the first part.

For $2\alpha \leq s-1$, apply Equation A.2

$$\begin{aligned}
s(s+1)\frac{\partial RD(1)}{\partial \alpha} \quad & > \quad \frac{s-2\alpha}{s-\alpha} + \frac{\alpha}{\alpha+1} - 1 \\
& = \quad \frac{a(s-2a-1)}{(s-a)(a+1)} \\
& \geq \quad 0
\end{aligned} \tag{A.20}$$

Thus, for the independent binary-variable problem, where $\alpha, \beta \in \mathbb{Z}_{>0}$, the smaller $|\alpha - \beta|$, the larger the $RD(1)$. This proves the lemma. □