University of Alberta

Library Release Form

Name of Author: Daniel James Lizotte

Title of Thesis: Budgeted Learning of Naïve Bayes Classifiers

Degree: Master of Science

Year this Degree Granted: 2003

Daniel James Lizotte
4 Brandon Street
Quispamsis, New Brunswick
Canada, E2E 1N9

Date: _____

"...But what is this data you refer to? No textile chemist knows exactly what it is that the buyer tests when he feels a tuft of cotton. Presumably there's the average length of the threads, their feel, the extent and nature of their slickness, the way they hang together, and so on. —Several dozen items, subconsciously weighed, out of years of experience. But the *quantitative* nature of these tests is not known; maybe even the very nature of some of them is not known. So we have nothing to feed the Machine."

Isaac Asimov
I, Robot

University of Alberta

Budgeted Learning of Naïve Bayes Classifiers

by

Daniel James Lizotte

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

Edmonton, Alberta
Fall 2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Budgeted Learning of Naïve Bayes Classifiers** submitted by Daniel James Lizotte in partial fulfillment of the requirements for the degree of **Master of Science**.

Russell Greiner (Supervisor)

Peter Hooper (External)

Robert Holte

Omid Madani

Date: _____

Abstract

Sometimes, data is not free. We consider the situation where the learner must 'purchase' its training data, subject to a fixed budget. In particular, we examine classifier learning where observing the value of a feature of a training example has an associated cost, and the total cost of all feature values acquired during training must remain less than the fixed budget. This thesis compares methods for sequentially choosing which feature value to purchase next, given the remaining budget and user's current knowledge of Naïve Bayes model parameters. This problem is similar to "active learning," but active learning scenarios assume the ability to purchase *class labels*, whereas we are interested in purchasing *feature values*. Also, work on active learning has traditionally focused on myopic (greedy) approaches and uniform/round-robin policies for query selection, but we show that such methods are often suboptimal and present a tractable method for incorporating knowledge of a fixed budget in the information acquisition process. This thesis extends ideas from [MLG03], and provides a more complete treatment of the topics covered in [LMG03].

Acknowledgements

I would like to thank my supervisor Russ Greiner for his insights and support, and for his confidence in me throughout this project. I would also like to thank Omid Madani for his endless supply of ideas and enthusiasm, and to wish him the best of luck in his future endeavours.

Finally, I would like to thank my good friend and colleague Colin Cherry, who began work with me on this project, but who has since moved on to wordier problems.

Contents

1 Introduction **1**

2 The Coins Problem **3**
2.1 Modeling the Coins Problem . . . 4
2.1.1 Bayesian Updating . . . 4
2.1.2 Beta Priors . . . 5
2.1.3 Computing Expected Loss . . . 6
2.2 Markov Decision Process Framework . . . 7
2.3 Policies . . . 9
2.3.1 Optimal Policy . . . 10
2.3.2 Greedy Loss Reduction . . . 12
2.3.3 Biased-Robin Policy . . . 12
2.3.4 Random Policy . . . 13
2.4 Allocations . . . 13
2.4.1 Optimal Allocation . . . 14
2.4.2 Single Coin Lookahead . . . 14
2.4.3 Uniform Allocation . . . 16
2.5 Empirical Results . . . 22

3 The Budgeted Naïve Bayes Problem **26**
3.1 Modeling the Budgeted Naïve Bayes Problem . . . 27
3.1.1 Bayesian Updating . . . 27
3.2 Markov Decision Process Framework . . . 28
3.3 Policies . . . 29
3.3.1 Optimal Policy . . . 29
3.3.2 Greedy Loss Reduction . . . 30
3.3.3 Biased-Robin . . . 30
3.4 Allocations . . . 30
3.4.1 Single Feature Lookahead (SFL) . . . 31
3.4.2 Uniform Policies . . . 32
3.5 Empirical Results . . . 32
3.5.1 Synthesized Data . . . 33
3.5.2 UCI Data . . . 34

4 Related Work **41**
4.1 Bandit Problems . . . 41
4.2 Hoeffding Races . . . 42
4.3 Active Learning . . . 42

5 Conclusions **44**
5.1 Future Work . . . 44
5.2 Contributions . . . 45

Bibliography **46**

A Proofs **48**

List of Figures

2.1 Beta Distributions 6
2.2 Updating Example 8
2.3 An optimal policy tree for budget $b = 3$ on identical uniform priors, where $n = 4$. Diamond boxes are actions, with 'head' outcomes leading *up*, and 'tail' outcomes leading *down*. Transition probabilities are indicated. Some branches terminate early as the coin to report (circled) is already determined. 10
2.4 Performance of the optimal policy versus several other policies on 10 coins with a budget of 10. (Note different Average Loss axis scales.) 24
2.5 These figures show performance (average loss) of various policies on 10 coins. Figure 2.5(a) shows the performance of non-optimal policies with a maximum budget of 40 and uniform priors. Figure 2.5(b) shows the performance of non-optimal policies with skewed priors $(\gamma_{i1}|\mathbf{s}_0, \gamma_{i2}|\mathbf{s}_0) = (10, 1)$ and a maximum budget of 40. Figure 2.5(c) shows performance when $(\gamma_{i1}|\mathbf{s}_0, \gamma_{i2}|\mathbf{s}_0) = (1, 10)$. Note that different plots have different Average Loss axis scales. 25

3.1 Initial state of $\mathcal{P}$. No feature values are known, all class labels are known. . . 26
3.2 Naïve Bayes Structure 27
3.3 Policies 29
3.4 Performance on synthesized data. 0/1 error on a validation set consisting of 20% of the complete data set. Errors are averages of 50 trials. 33
3.5 Performance on UCI Mushroom data. 0/1 error error on a validation set consisting of 20% of the data. Errors are averages of 50 trials. 38
3.6 Performance on UCI Nursery data. 0/1 error error on a validation set consisting of 20% of the data. Errors are averages of 50 trials. 39
3.7 Performance on UCI Votes data. 0/1 error error on a validation set consisting of 20% of the data. Errors are averages of 50 trials. 40

A.1 *(Repeated from page 10.)* An optimal policy tree for budget $b = 3$ on identical uniform priors, where $n = 4$. Diamond boxes are actions, with 'head' outcomes leading up, and 'tail' outcomes leading down. Transition probabilities are indicated. Some branches terminate early as the coin to report (circled) is already determined. 50

List of Symbols

$P(x)$	Probability of event x
$\langle X \rangle$	Expected value of random variable X, *i.e.*, $\sum_{x \in \text{Range}(X)} x \cdot P(x)$
$\text{pdf}(x)$	Probability distribution function $P(X = x)$
$\text{cdf}(x)$	Cumulative density function $P(X \leq x)$
$\mathbb{R}^{\geq 0}$	The nonnegative real numbers
$\mathbb{Z}^{\geq 0}$	The nonnegative integers
$\mathscr{b}$	The budget $\in \mathbb{R}^{\geq 0}$
$\mathscr{n}$	Number of coins or features $\in \mathbb{Z}^{+}$
C_i	Bernoulli-distributed Coin, takes values $\in \{1, 0\}$
Θ_i	Beta distributed parameter of C_i, *i.e.*, $P(C_i = 1)$
γ_{ik}	Beta parameter corresponding to counts of $C_i = c_k$
$\mathbf{s}$	Belief state
S	Set of all belief states
ℓ	Loss function, $\ell : \text{S} \rightarrow \mathbb{R}^{\geq 0}$
a_i	Action of flipping coin C_i
A	Set of all possible actions
$\check{\text{a}}^*$	Optimal static allocation of budget across coins or features
$\check{\text{a}}_i^1$	Allocation of entire budget $\mathscr{b}$ to coin C_i
$\check{\text{a}}^-$	Uniform allocation of budget across all coins or features
X_i	Feature X_i, takes on values $\in \{x_1, x_2, ..., x_k\}$ where $k = \lvert X_i \rvert$.
$\lvert X_i \rvert$	number of different values X_i may take on
Y	Class label Y, takes on values $\in \{y_1, y_2, ..., x_l\}$ where $l = \lvert Y \rvert$.
Θ_{ijk}	Dirichlet distributed parameter of $X_i \lvert Y_j$, *i.e.*, $P(X_i = x_k \lvert Y_j)$
γ_{ijk}	Dirichlet parameter corresponding to counts of $X_i = x_k \lvert Y_j$
a_{ij}	Action of purchasing a value of X_i when $Y = y_j$
$\check{\text{a}}_{ij}^1$	Allocation of entire budget to purchasing a value of X_i when $Y = y_j$

Note: When certain symbols require more context, (*i.e.*, state) the *conditioning bar* is used. For example $\gamma_{ik}|\mathbf{s}$ represents the value of γ_{ik} in state $\mathbf{s}$.

Chapter 1

Introduction

A recent project was allocated $2 million to develop a diagnostic classifier for cancer subtypes. In the study, a pool of patients with known cancer subtypes was available, as were various diagnostic tests that could be performed, each with an associated cost. Experts theorized that some combination of these tests would be capable of discriminating between subtypes; the challenge was to build a classifier using these tests that would be the most accurate.

The first step is to acquire the relevant information: here, we have to decide which tests to perform on which patients. The standard approach, of course, is simple round-robin: run every test on every patient until we exhaust our fixed budget. Given our finite budget, however, this might not produce the best classifier — *e.g.*, if we can determine that two tests are equivalent, it is clearly inefficient to perform both tests on the same patient. Fortunately, there are many other options. For example, we could run all the tests on a *subset* of the large pool of patients to examine their usefulness and correlations and then run only the most relevant tests on future patients. Indeed, we could go to the extreme of building a dynamic policy that, at each time step, decides which tests to perform on which patient, based on all of the information available about the costs and apparent effectiveness of the tests, as well as the remaining available funds.

This thesis explores this idea: how to dynamically decide which tests to run on which individual to produce the most effective classifier, subject to the known firm budget.

Our "budgeted learning task" is an instance of decision making under uncertainty, a vast topic that has been explored by researchers since the advent of statistics. Chapter 4 examines how these other explorations relate (and do not relate) to the issues discussed here. In particular, we explain how our objective differs from standard bandit problems, Hoeffding

races, on-line learning, active learning, and active classification. Chapter 2 describes the 'Coins Problem,' which we use to develop the foundation of budgeted learning. Insight gained examining this problem has been extended for use in Naïve Bayes classifiers (which our learners will return) in Chapter 3. We have implemented these systems and run a number of tests on both real and synthesized datasets; Section 3.5 reports our findings. We see in particular that Round-Robin is typically not the most effective policy. The URL [Web] provides additional information, both theoretical (*e.g.*, proofs) and empirical (datasets, etc.).

Chapter 2

The Coins Problem

In order to shed some light on the general principles and ideas involved in budgeted learning, we examine the following simplified problem:

Problem 1. The Coins Problem. *We are given a set* $\mathrm{C} = \{C_i\}$ *of* n *independent Bernoulli random variables* C_i *from which we may draw a total of* b iid *samples. We are uncertain about the expected value* $\langle C_i \rangle$ *of each variable*[1]*. Our task has two phases: First, we choose which distribution each sample should come from and observe the resulting data. Once we have exhausted our budget of* b *samples, we report the random variable we believe to have the highest expected value given our observations* $\mathcal{D}$*, i.e.,* $\arg\max_i \{\langle C_i | \mathcal{D} \rangle\}$*. The performance associated with reporting the* r*th random variable is measured by the loss function* $\ell(r) = \max_i\{\langle C_i \rangle\} - \langle C_r \rangle$*, which we wish to minimize.*[2]

We call this problem the *Coins Problem* because it describes the situation of having a handful of n unfair coins that we flip some number of times to determine which one has the highest probability of turning up 'heads.' Throughout this chapter we assume uniform unit costs across coins, *i.e.*, the cost of obtaining a sample from any coin is 1. However, extending the algorithms presented here to handle nonuniform costs is straightforward.

The Coins Problem is intended to provide a simplified environment that helps us gain some intuition into the decision making required for budgeted learning. It is a problem of decision making under uncertainty that retains the same reward structure as our final goal of budgeted learning: performance is only evaluated after all intermediate decision making has been completed.

Although the Coins Problem has a similar structure to our goal problem of budgeted

[1] We denote the expected value of a random variable X by $\langle X \rangle$.

[2] In this expression, $\langle C_i \rangle$ and $\langle C_r \rangle$ represent the 'true' expected values of the underlying Bernoulli distributions.

classifier learning, it is only remotely similar to classifier learning in general. It is most closely related to feature selection: choosing the distribution with the highest expected value is akin to choosing the most correlated feature. However, the simplifications in this model would only allow us to identify the most *positively* correlated feature, for example. Nevertheless, the strategies developed to tackle the coins problem can be successfully extended to the more interesting application of budgeted classifier learning, as described in Chapter 3. In addition to providing a starting point for examining budgeted decision problems, the coins problem is interesting as a model of situations where we wish to choose the best among objects whose "goodness" can be measured in isolation, *i.e.*, independently of the goodness of other objects. The problem of selecting the best among a set of atomic medical *treatments*, subject to a budget, might have this property, for example.

2.1 Modeling the Coins Problem

First, we define random variables Θ_i that assume values $\theta_i \in [0, 1]$ and correspond to the parameters of the Bernoulli random variables C_i (*i.e.*, $\theta_i = \langle C_i \rangle$). We assign a prior distribution to each Θ_i, and compute posterior distributions using the incoming data and Bayes' Theorem. We will use these posterior distributions to compute the quantities necessary for optimizing our loss function $\ell(\cdot)$.

2.1.1 Bayesian Updating

Bayes' Theorem allows us to compute the posterior probability of a random variable given our observations of some other random variable(s), usually called 'data' or 'evidence.'

$$P(\Theta_i|\mathcal{D}) = \frac{P(\mathcal{D}|\Theta_i) \cdot P(\Theta_i)}{P(\mathcal{D})}$$

Here, $\mathcal{D}$ represents the data we have seen so far, and Θ_i is our parameter of interest. The value $P(\mathcal{D}|\Theta_i)$ represents the likelihood of the data given Θ_i, $P(\Theta_i)$ represents the *prior* distribution on Θ_i, *i.e.*, the distribution we believe Θ_i to have before we see any data. The quantity $P(\mathcal{D})$ can be viewed as a normalizing term to ensure the new posterior distribution integrates to 1.

The value $P(\mathcal{D}|\Theta_i)$ is easily derived from the definition of the Bernoulli distribution [Dev95]. The likelihood of observing $C_i = 1$ is Θ_i, and the likelihood of observing $C_i = 0$ is $(1 - \Theta_i)$. Because we assume that the events we observe are independent (part of the Bernoulli assumption) the likelihood of observing $C_i = 1$ a total of $(\gamma_{i1} - 1)$ times and

$C_i = 0$ a total of $(\gamma_{i2} - 1)$ times[3] is

$$P(\mathcal{D}|\Theta_i) = \binom{\gamma_{i1} + \gamma_{i2} - 2}{\gamma_{i1} - 1} \cdot \Theta_i^{\gamma_{i1}-1}(1-\Theta_i)^{\gamma_{i2}-1}$$

If we choose a uniform prior over Θ_i, *i.e.*, $\text{pdf}(\Theta_i) = 1$, then the posterior distribution is simply

$$\begin{aligned} P(\Theta_i|\mathcal{D}) &= \frac{P(\mathcal{D}|\Theta_i) \cdot P(\Theta_i)}{P(\mathcal{D})} \\ &= \frac{\Theta_i^{\gamma_{i1}-1} \cdot (1-\Theta_i)^{\gamma_{i2}-1}}{P(\mathcal{D})} \\ &= \frac{\Theta_i^{\gamma_{i1}-1} \cdot (1-\Theta_i)^{\gamma_{i2}-1}}{\mathrm{B}(\gamma_{i1}, \gamma_{i2})} \end{aligned} \tag{2.1}$$

where the normalizing term $\mathrm{B}(x_1, x_2)$ is the (complete) Beta function

$$\mathrm{B}(x_1, x_2) = \int_0^1 \Theta^{x_1-1}(1-\Theta)^{x_2-1}\,\mathrm{d}\Theta = \frac{\Gamma(x_1)\Gamma(x_2)}{\Gamma(x_1+x_2)}$$

and $\Gamma(x)$ is the Gamma function

$$\begin{aligned} \Gamma(x) &= \int_0^\infty t^{x-1}\mathrm{e}^{-t}\,\mathrm{d}t \;=\; (x-1)\Gamma(x-1) \\ &= (x-1)! \;\text{ for } x \in \mathbb{Z}^+ \end{aligned}$$

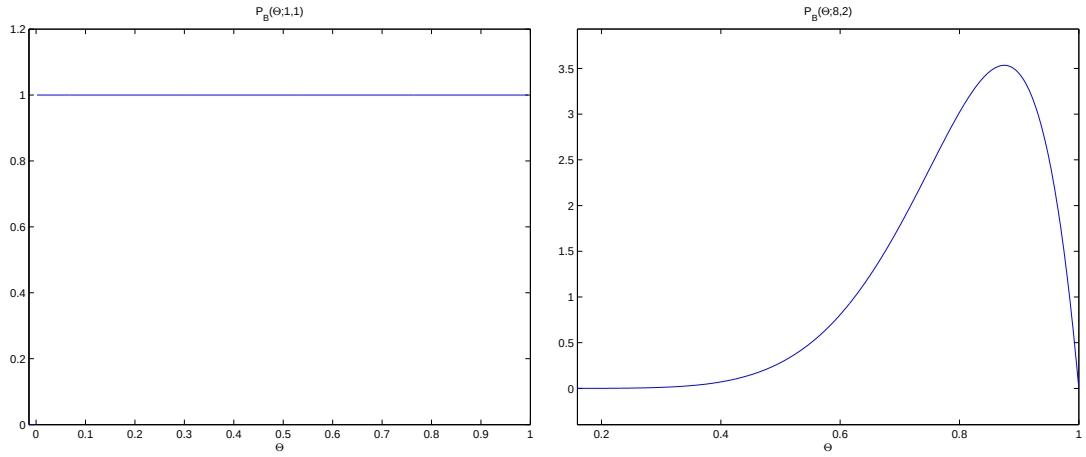
The distribution defined by Equation 2.1 is known as a *Beta distribution*, which is the *conjugate prior* of a Bernoulli parameter [Hec95]. The Beta distribution is called the conjugate prior of a Bernoulli distribution because the posterior resulting from updating a Beta prior with an event having Bernoulli likelihood is also a Beta distribution. The Beta distribution is a continuous distribution defined over $[0, 1]$ that has two parameters $\gamma_1, \gamma_2 \in \mathbb{R}^+$. A Beta distribution with parameters (γ_1, γ_2) is given by

$$P_{\mathrm{B}}(\Theta; \gamma_1, \gamma_2) = \frac{\Theta^{\gamma_1-1}(1-\Theta)^{\gamma_2-1}}{B(\gamma_1, \gamma_2)}$$

2.1.2 Beta Priors

For the above example, we chose $P(\Theta_i) = 1$, which is equivalent to $P(\Theta_i) = P_{\mathrm{B}}(\Theta_i; 1, 1)$. This choice caused the parameters of the posterior to be equal to the event counts *plus one.* In general, if we take any Beta distribution $P_{\mathrm{B}}(\Theta_i; \gamma_{i1}^0, \gamma_{i2}^0)$ as our prior, the Bayes-updated posterior after observing γ_{i1} 'heads' and γ_{i2} 'tails' will also be a Beta distribution $P_{\mathrm{B}}(\Theta_i; \gamma_{i1} + \gamma_{i1}^0, \gamma_{i2} + \gamma_{i2}^0)$. Intuitively, we can think of γ_{i1}^0 and γ_{i2}^0 as imaginary previously

[3]Here, we have defined the 'event counts' to be one less than the values γ_{ij} so that they may be used directly in the usual definition of the Beta distribution as given in Equation 2.2.



(a) $\gamma_1 = 1, \gamma_2 = 1$ (b) $\gamma_1 = 8, \gamma_2 = 2$

Figure 2.1: Beta Distributions

seen samples from the distribution of C_i. They act as parameters of the prior, and are also called *hyperparameters*. For example, using a $P_{\mathrm{B}}(\Theta_i; 1, 1)$ prior is like imagining we have seen one 'heads' and one 'tails' from coin i, while a $P_{\mathrm{B}}(\Theta_i; 8, 2)$ prior indicates that we imagine having seen eight 'heads' and two 'tails.' (And therefore that coin C_i probably has expected value close to 0.8.) Graphs of these priors are shown in Figure 2.1. Most commonly, we will assume the uniform prior $P_{\mathrm{B}}(\Theta_i; 1, 1)$ unless we have reason to believe otherwise.[4]

As we accumulate data by querying distributions, (*i.e.*, by flipping coins) we will update the original hyperparameters by incrementing γ_{i1} every time we see $C_i = 1$ and incrementing γ_{i2} every time we see $C_i = 0$. A *belief state* $\mathbf{s} \in \mathrm{S}$ is a set of pairs $\{(\gamma_{i1}, \gamma_{i2}) | i \in \{1...n\}\}$ that records the current value of Beta parameters based on our chosen priors and the data observed so far. S is the set of all possible belief states. Each time we observe a new data value, we move to a new belief state.

2.1.3 Computing Expected Loss

These posterior distributions allow us to compute and make decisions about the expected loss $\ell(j) = \max_i\{\langle C_i \rangle\} - \langle C_j \rangle$ associated with reporting a particular coin j based on the data we have seen.

[4]Note that it is not possible to have a prior that assumes having seen no 'heads' or no 'tails,' as $\Gamma(0)$ is undefined and the posterior cannot be normalized. Assuming a $P_{\mathrm{B}}(\Theta; 1, 1)$ prior is the Bayesian equivalent to the frequentist approach of Laplace smoothing.

Lemma 1. *To minimize expected loss, we should report the coin C_r where $r = \arg\max_i\{\langle\Theta_i\rangle\}$.*

Proof. In general, $\forall j$ the expectation of $\ell(j) = \max_i\{\langle C_i\rangle\} - \langle C_j\rangle$ is given by

$$\begin{aligned}\langle\ell(j)\rangle &= \langle\max_i\{\langle C_i\rangle\} - \langle C_j\rangle\rangle \\ &= \langle\max_i\{\Theta_i\} - \Theta_j\rangle \\ &= \langle\max_i\{\Theta_i\}\rangle - \langle\Theta_j\rangle \qquad (2.2)\end{aligned}$$

From Equation 2.2 and the fact that $\langle\max_i\{\Theta_i\}\rangle$ does not depend on j, we see that $\arg\min_r\{\langle\ell(r)\rangle\} = \arg\max_r\{\langle\Theta_r\rangle\}$. □

From Lemma 1, we see that if we are to report a coin and we wish to minimize loss as defined in the problem, we must report the coin that has the highest expected value[5] in the current state $\mathbf{s}$, *i.e.*, $\arg\max_i\langle\Theta_i|\mathbf{s}\rangle$. Because we will always report a coin with maximum expected value based on our observed sample, our expected loss in a particular belief state $\mathbf{s}$ is

$$\ell(\mathbf{s}) = \langle\max_i\{\Theta_i|\mathbf{s}\}\rangle - \max_i\{\langle\Theta_i|\mathbf{s}\rangle\} \qquad (2.3)$$

In practice, we compute $\arg\max_r\langle\Theta_r|\mathbf{s}\rangle$ from the posterior distributions of the Θ_i. The expected value of the i-th coin's posterior is $\langle\Theta_i|\mathbf{s}\rangle = \hat{\theta}_i|\mathbf{s} = \gamma_{i1}/(\gamma_{i1}+\gamma_{i2})$ where γ_{i1} and γ_{i2} are the parameters of the distribution of Θ_i in state $\mathbf{s}$. Therefore, $\max_i\langle\Theta_i|\mathbf{s}\rangle = \max_i\hat{\theta}_i|\mathbf{s}$ which we call $\hat{\theta}_{\max}|\mathbf{s}$. Also, for convenience we will denote the random variable $\max_i\{\Theta_i|\mathbf{s}\}$ as $\Theta_{\max}|\mathbf{s}$, which means Equation 2.3 is written as

$$\ell(\mathbf{s}) = \langle\Theta_{\max}|\mathbf{s}\rangle - \hat{\theta}_{\max}|\mathbf{s} \qquad (2.4)$$

2.2 Markov Decision Process Framework

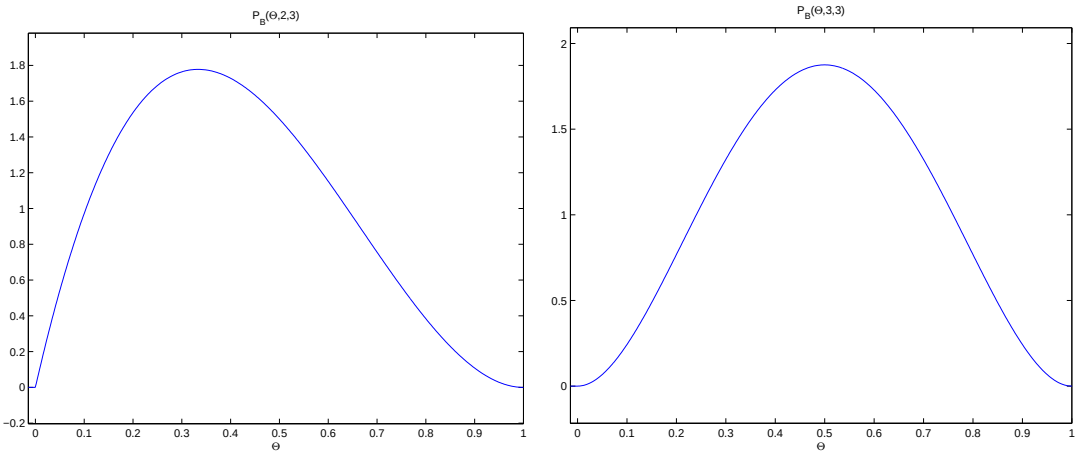
We can easily view the coins problem as a Markov Decision Process (MDP) [Put94]. To work in the MDP framework, we need the notion of *states* $\mathbf{s} \in$ S, which in our case are the posterior distributions of each coin's parameter Θ_i (defined by a list of pairs of parameters) along with the remaining budget, and a *reward* function to indicate performance, which in our case is the (negative) loss function $\ell(\cdot)$. This reward is only obtained at *outcome* states where the budget has been exhausted; reward at all intermediate states is 0. (For our purposes, we will work with the less common convention of an MDP whose goal is to minimize loss, instead of maximizing reward. These two conventions are equivalent.) To

[5]There may be more than one maximum coin. If so, we may break ties however we wish and still retain this minimum loss property.

complete the MDP formulation, we need to define *actions*, which we denote $\mathrm{a}_i \in \mathrm{A}$, where a_i represents the action of flipping coin i (and thus obtaining a new data sample and moving to a new belief state).

Moving to new belief states is accomplished via the Bayes updating procedure described in Section 2.1.1. We assume each coin i has a Bernoulli parameter $\Theta_i \sim \mathrm{B}(\gamma_{i1}, \gamma_{i2})$ that is Beta distributed with parameters γ_{i1} and γ_{i2}. When we observe a new data point $C_i = c_k,\ k \in \{1, 2\}$, we increment parameter γ_{ik}. For example, coin 3 might have its parameter $\Theta_3 \sim \mathrm{B}(2, 3)$. If we perform action a_3 (*i.e.*, flip coin 3) and observe $C_3 = 1$ (*i.e.*, a heads), then the new posterior is $\Theta_3 \sim \mathrm{B}(3, 3)$. This change in distribution is illustrated by Figure 2.2.

Finally, we need a *transition model*, which is a probability distribution over successor states given an action and our current state. An action a_i results in observing one more data point from coin i, so from our current belief state $\mathbf{s}$ we can get to one of two possible successor states by taking a_i: one where γ_{i1} has been incremented, and one where γ_{i2} has been incremented. We call these states $\mathbf{s}|\gamma_{i1}$++ and $\mathbf{s}|\gamma_{i2}$++, respectively. Given our current knowledge, the probability of incrementing the k-th 'heads' parameter γ_{i1} when performing action i is $\hat{\theta}_i = \gamma_{i1}/(\gamma_{i1} + \gamma_{i2})$. (The probability of incrementing γ_{i2} is $1 - \hat{\theta}_i$.)



(a) Coin 3 before update: $\mathrm{B}(2, 3)$

(b) Coin 3 after seeing one more 'head': $\mathrm{B}(3, 3)$

Figure 2.2: Updating Example

2.3 Policies

We define a *policy* to be a function $\pi : \mathrm{S} \rightarrow \mathrm{A}$ that takes the current belief state (*i.e.*, current posterior distributions) and the remaining budget, and outputs an action $\mathrm{a}_i \in \mathrm{A}$ that indicates which coin to flip next. Policies have the potential to be dynamic or *contingent* in the sense that their actions can be influenced by the accumulating data.

The expected loss (sometimes called regret) of a *policy* is given by

$$\ell(\pi, \mathbf{s}_0) \quad = \quad \sum_{\mathbf{s} \in \mathcal{O}(\pi, \mathbf{s}_0)} P(\mathbf{s}) \cdot \ell(\mathbf{s})$$

Here, $\mathcal{O}(\pi, \mathbf{s}_0)$ is the set of all possible *outcomes* of π starting from $\mathbf{s}_0$, *i.e.*, states reached by beginning at $\mathbf{s}_0$, and following π until the budget is exhausted, at which time we report a coin. Of course, outcomes are stochastic because the observed events that define them are stochastic. In general, with n coins and a budget of b, there are

$$\sum_{h=0}^{\mathit{b}} \left[\binom{h + \mathit{n} - 1}{h} \binom{(\mathit{b} - h) + \mathit{n} - 1}{\mathit{b} - h} \right]$$

different possible outcomes we can reach. (Note that not all of these outcomes might be reached by a given policy.)

It is possible to calculate the posterior probability of observing a particular outcome from the current distributions of Θ_i. Given a current belief state $\mathbf{s} = \{(\gamma_{i1}, \gamma_{i2}) | i = 1..\mathit{n}\}$ and a number of flips b_i allocated to coin i, the probability of reaching an end belief state $\mathbf{s}' = \{(\gamma'_{i1}, \gamma'_{i2}) | i = 1..\mathit{n}\}$ is given by $P(\mathbf{s}' | \mathbf{s}, \mathit{b}) = \prod_{i=1}^{\mathit{n}} P((\gamma'_{i1}, \gamma'_{i2}) | (\gamma_{i1}, \gamma_{i2}), \mathit{b}_i)$ because we assume that the C_i are conditionally independent.

The probability of a particular coin reaching the end state $(\gamma'_{i1}, \gamma'_{i2})$ is the probability of observing $\gamma'_{i1} - \gamma_{i1}$ 'heads' and $\gamma'_{i2} - \gamma_{i2}$ 'tails' out of a total of $(\gamma'_{i1} + \gamma'_{i2}) - (\gamma_{i1} + \gamma_{i2})$ flips. The random variable describing this event has a Beta-Binomial distribution, which is similar to the standard binomial distribution but assumes a Beta prior on the parameter θ instead of a point value. The Beta-Binomial distribution is given by [Hec95][6]

$$P((\gamma'_{i1}, \gamma'_{i2}, ..., \gamma'_{ik}) | (\gamma_{i1}, \gamma_{i2}, ..., \gamma_{ik})) = \frac{\Gamma(\sum_k \gamma_{ik})}{\Gamma(\sum_k \gamma'_{ik})} \prod_k \frac{\Gamma(\gamma'_{ik})}{\Gamma(\gamma_{ik})} \tag{2.5}$$

Note that this probability is also implicity conditioned on the number of samples obtained in moving from $\mathbf{s}$ to $\mathbf{s}'$, *i.e.*, $\sum_k \gamma'_{ik} - \sum_k \gamma_{ik}$. Equation 2.5 represents the Beta-Binomial distribution when $k \in \{1, 2\}$, *i.e.*, when k ranges over two possible events ('heads' and

[6] Heckerman describes this quantity as the 'probability of imagined future data.'

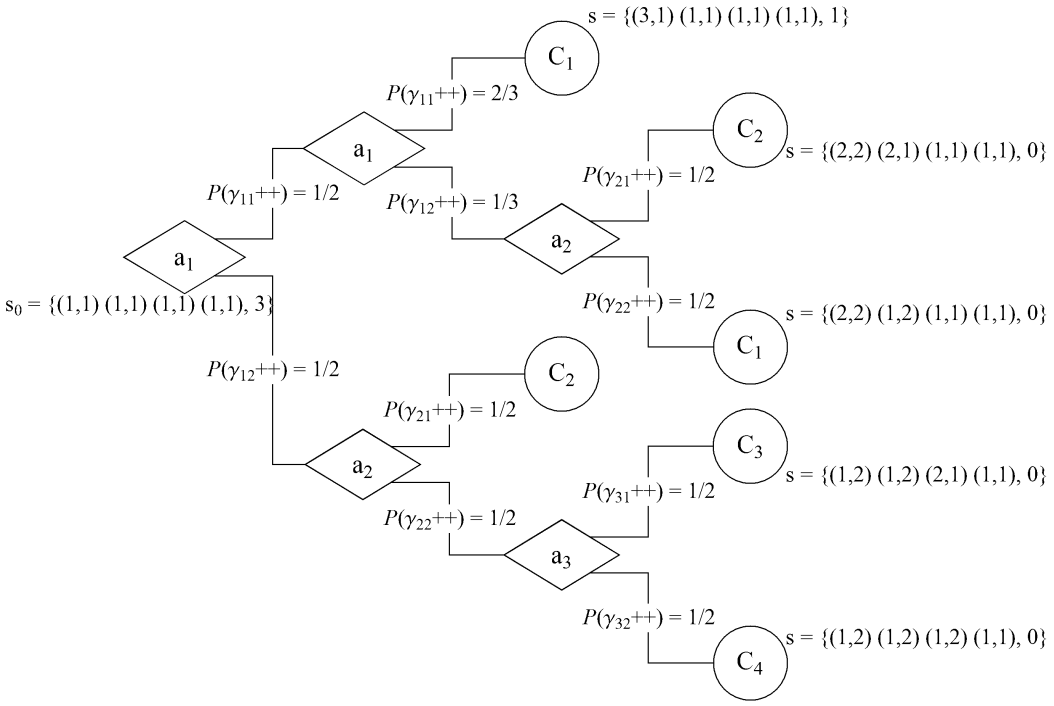


Figure 2.3: An optimal policy tree for budget $b = 3$ on identical uniform priors, where $n = 4$. Diamond boxes are actions, with 'head' outcomes leading *up*, and 'tail' outcomes leading *down*. Transition probabilities are indicated. Some branches terminate early as the coin to report (circled) is already determined.

'tails'.) It can also represent the *Dirichlet-Multinomial* distribution, which can model random variables with more than two events. (Take a die roll, for example. To model this, let $k \in \{1, 2, ..., 6\}$.) This will become important when we discuss budgeted classifier learning in Chapter 3.

2.3.1 Optimal Policy

We now present an algorithm that generates the optimal policy for a given coin problem.

We have states $\mathbf{s} \in \mathrm{S}$, a known stochastic state transition function, and a loss function ℓ. It is widely known that for every MDP there exists a deterministic optimal policy π^* that minimizes expected loss. (Or, equivalently, maximizes expected reward.) A graph describing the optimal policy for $n = 4$ coins and a budget of $b = 3$ flips is shown in Figure 2.3. In this diagram, actions are represented in diamonds, with arcs representing the two possible outcomes from each action: arcs leading up from an a_i diamond represent observing $C_i = 1$, while those leading down represent $C_i = 0$. The effect of each observation on the current parameters is noted on each arc, along with its probability of occurring. Final states are

represented as circles indicating which coin would be reported[7]. We now show how to construct an optimal policy (such as this one) that minimizes expected loss.

First, we define an optimal expected loss function $\ell^* : \mathrm{S} \rightarrow \mathbb{R}^+$ that represents the expected loss if we begin in state $\mathbf{s}$ and act optimally. (This is typically written as a *value* function V^*. For us, $\ell^* = -V^*$.)

We have already seen that reporting the coin with the highest expected value $\hat{\theta}_{\max}$ results in the lowest expected loss over all coins. Thus, given a budget $\mathit{b} = 0$, the optimal policy simply reports coin $r = \arg\max_i \{\langle \Theta_i \rangle\}$, which gives an optimal expected loss for state $\mathbf{s}$ of $\ell(\mathbf{s}) = \langle \Theta_{\max} \rangle - \hat{\theta}_{\max}$ from Equation 2.4.

Consider now the case where $\mathit{b} = 1$. Our policy may flip one coin, and must then report which coin is optimal. To see which coin the optimal policy would choose to flip, we consider all possible outcomes of all possible flips, and compute the optimal expected loss ℓ^* associated with each flip using Equation 2.4 together with the transition probabilities. The expected loss of the current state associated with flipping coin C_i is given by

$$\langle \ell^*(\mathbf{s}) | \mathrm{a}_i \rangle \quad = \quad P(C_i = 1|\mathbf{s}) \cdot \ell^*(\mathbf{s}|\gamma_{i1}\texttt{++}) \quad + \quad P(C_i = 0|\mathbf{s}) \cdot \ell^*(\mathbf{s}|\gamma_{i2}\texttt{++}) \tag{2.6}$$

Since, in this case, flipping i will exhaust our budget, we can compute $\ell^*(\mathbf{s}|\gamma_{i1}\texttt{++})$ and $\ell^*(\mathbf{s}|\gamma_{i2}\texttt{++})$ from Equation 2.4. The probabilities $P(C_i = x)$ are simply the expected values of the Θ_i random variables, so we have

$$\langle \ell^*(\mathbf{s}) | \mathrm{a}_i \rangle \quad = \quad \hat{\theta}_i|\mathbf{s} \cdot \ell^*(\mathbf{s}|\gamma_{i1}\texttt{++}) \quad + \quad (1 - \hat{\theta}_i|\mathbf{s}) \cdot \ell^*(\mathbf{s}|\gamma_{i2}\texttt{++}) \tag{2.7}$$

and

$$\ell^*(\mathbf{s}) = \min_{\mathrm{a}_i} \langle \ell^*(\mathbf{s}) | \mathrm{a}_i \rangle \tag{2.8}$$

Equations 2.8 and 2.7 together are analogous to the Bellman backup equation for MDPs [RN95]. We have demonstrated their use for the case where $\mathit{b} = 1$, but the exact value of $\ell^*(\mathbf{s})$ can be computed for any budget by starting from all possible outcome states (*i.e.*, where $\mathit{b} = 0$) and 'backing up' the values to earlier states with more budget, until the start state $\mathbf{s}_0$ is reached. This process is shown in Algorithm 1.

Once ℓ^* has been calculated for all possible states, we can use it to choose the optimal policy by simply choosing the action that is expected to result in the minimum ℓ^*, *i.e.*, $\pi^*(\mathbf{s}) = \arg\min_{\mathrm{a}_i} \langle \ell^*(\mathbf{s}) | \mathrm{a}_i \rangle$.

[7]Note that some branches terminate early, as spending the remaining budget on *any* coin after certain states would not change which coin had the highest expected value, and therefore would not change our decision of which coin to report.

Algorithm 1 Recursive algorithm for ℓ^*

compute_ℓ^* $(\mathbf{s}, \mathit{b})$
if $\mathit{b} = 0$ **then**
  return $\langle \max_i \{\Theta_i | \mathbf{s}\} \rangle - \hat{\theta}_{\max} | \mathbf{s}$
else
  return $\min_f \left\{ \begin{array}{rl} & \hat{\theta}_f \cdot \textbf{compute\_}\ell^*(\mathbf{s}|\gamma_{f1}{+}{+}, \mathit{b}-1) \\ + & (1-\hat{\theta}_f) \cdot \textbf{compute\_}\ell^*(\mathbf{s}|\gamma_{f2}{+}{+}, \mathit{b}-1) \end{array} \right\}$
end if

Of course, the number of states that can arise from a particular starting state is exponential in the size of the budget, (*i.e.*, $O(n^{\mathit{b}})$) so calculating the exact expected loss for all states by this exhaustive method is impractical except for very small instances of the coins problem.

2.3.2 Greedy Loss Reduction

The optimal policy becomes intractable because the number of potential outcomes grows exponentially with the amount we look ahead. One typical method of counteracting this growth is to limit the lookahead to one step, which is tractable. The Greedy policy always takes the action that is expected to result in the least loss *if we had to report a coin immediately after the next step.* It is fast to compute (takes $O(n)$ time), and is contingent on incoming data, but it does not take the budget into account. Empirical performance of this policy (and of others) is discussed in Section 2.5.

2.3.3 Biased-Robin Policy

Another heuristic policy that is tractable and can identify coins with high expected value is *Biased-Robin*, which operates as follows: Biased-Robin continues to flip a coin that returns

Algorithm 2 Biased-Robin for Coins Problem

$i \leftarrow 0$
while $\mathit{b} > 0$ **do**
  Flip C_{i+1}, obtaining sample D
  $\mathit{b} \leftarrow \mathit{b} - 1$
  if $D = 0$ {*i.e.*, 'tails'} **then**
    $i \leftarrow (i+1) \mod n$
  end if
end while

'heads' (which is good) until it observes a 'tails' (which is bad). At this point, it moves to the next coin and does the same thing, wrapping around as necessary. (The policy is called *Biased-Robin* because of its similarity to *Round-Robin* which flips each coin in sequence and wraps around as necessary until the budget runs out. The Round-Robin policy and its

characteristics are discussed in Section 2.4.3.) The intuitive justification for why this policy might perform well is simple: Since 'better' coins (*i.e.*, coins with higher expected value) will turn up 'heads' more, this policy will probably query them more. This means that 'good' coins should be separated from 'bad' coins fairly easily, and the set of 'good' coins will have more total data acquired, which will hopefully enable identification of the 'best' coin from among the 'good.'

Of course, the words 'probably' and 'hopefully' are used here in a completely nontechnical sense: we have no approximability guarantees regarding Biased-Robin. It is simply an example of a concise, hand-engineered policy that is extremely fast to compute. It takes into account the incoming data about coin flips, but does not consider the budget.

2.3.4 Random Policy

Another policy we will use in our empirical comparisons is the random policy, where the next coin is chosen from a uniform discrete distribution. The probability of flipping any coin i is $1/n$. This policy is the only stochastic policy we will examine. As the budget gets large we expect it to perform nearly identically to a Round-Robin policy (or, equivalently, to a Uniform Allocation; see Section 2.4.3).

2.4 Allocations

An *allocation* is a static policy that flips each coin a predetermined number of times, conveniently described by listing the amount of budget allocated to each coin C_i, *e.g.*, the allocation $(3, 0, 2)$ indicates to flip coin C_1 three times, coin C_2 zero times, and coin C_3 two times. In general, an allocation $\breve{a} = (\breve{a}_1, \breve{a}_2, ...\breve{a}_n)$ represents obtaining $\breve{a}_1 \in \mathbb{Z}^{\geq 0}$ samples from C_1, $\breve{a}_2 \in \mathbb{Z}^{\geq 0}$ samples from C_2, etc. Allocations are *not* dynamic as they do not base their actions on incoming data. With n unit-cost coins and a budget of b, there are

$$\binom{b+n-1}{n-1} \in O(b^{n-1})$$

different possible allocations. (Each allocation is an integer *composition* of b into n parts.)

It is possible to evaluate the expected loss of a particular allocation, as we could for any policy, by summing over all possible outcomes of the allocation, of which there are

$$|\mathcal{O}(\breve{a})| = \prod_{i=0}^{n} (\breve{a}_i + 1)$$

For some special cases, however, it is possible to express expected loss more compactly; see Section 2.4.3.

2.4.1 Optimal Allocation

Since the number of different possible allocations is polynomial in the budget (but exponential in n) it is possible to enumerate all allocations for modest n and evaluate their expected loss to find the optimal allocation, either exactly by summing over all possible outcomes or approximately by sampling. The resulting optimal allocation $\check{a}^*$ is the policy that results in the minimum expected loss out of all possible *non-contingent* policies.

Table 2.1 shows several examples of optimal allocations, uniform allocations, and single coin allocations, which are discussed below. It is interesting to note that neither the uniform allocation nor the best single coin allocation is optimal for any of these situations.

2.4.2 Single Coin Lookahead

Consider allocating the *entire budget* to a single coin i, denoted $\check{a}_i^1$. This *single coin allocation* (SCA) allocation has only $b+1$ different possible outcomes, so its expected loss is

$$\ell(\check{a}_i^1) = \langle \Theta_{\max} \rangle - \Gamma(\gamma_{i1}+\gamma_{i2}) \sum_{h=0}^{b} \left[\frac{1}{\Gamma(\gamma_{i1}+\gamma_{i2}+b)} \cdot \frac{\Gamma(\gamma_{i1}+h)\Gamma(\gamma_{i2}+(b-h))}{\Gamma(\gamma_{i1})\Gamma(\gamma_{i2})} \cdot \hat{\theta}_{\max}^{(\mathbf{s}')} \right]$$

Here $\mathbf{s}' = \mathbf{s}'(i, \mathbf{s}, h, b)$ represents the state equal to $\mathbf{s}_0$ except that γ_{i1} is increased by h and γ_{i2} is increased by $(b-h)$. The expected loss of a single coin allocation (which we call "SCA loss") is computable in $O(b)$ time. We could consider actually following this simplistic policy of allocating the entire budget to one coin, but as can be seen in Table 2.1, frequently even the best of these single coin allocations has high expected loss, particularly when n is large. We therefore do not propose to actually allocate all b flips to coin C_i where $i = \arg\max_i\{\ell(\check{a}_i^1)\}$, but instead develop a *contingent* policy based on the best possible expected SCA loss.

We define a policy called 'Single Coin Lookahead,' denoted π_{SCL}, that uses SCA loss values but is contingent. It works as follows: At each step, the coin with the minimum SCA loss score is found. That coin is flipped *once*, and the belief state is updated based on the outcome. The process is then repeated, always flipping the coin that has the minimum SCA loss according to the current belief state.

This policy was developed with the intention of creating a policy that is not only contingent (as is Biased-Robin, say) but also explicitly takes the budget into account. Its empirical performance is discussed in Section 2.5.

	$\Theta_1 \sim \mathrm{B}(1,1)$	$\Theta_2 \sim \mathrm{B}(1,1)$	$\Theta_3 \sim \mathrm{B}(1,1)$	$\Theta_4 \sim \mathrm{B}(1,1)$	$\langle \ell \rangle$
$\check{\mathrm{a}}^*$	3	2	2	1	0.091204
$\check{\mathrm{a}}^-$	2	2	2	2	0.102469
$\check{\mathrm{a}}^1_1$	8	0	0	0	0.188889

(a) Identical uniform priors, $n = 4, b = 8$.

	$\Theta_1 \sim \mathrm{B}(1,1)$	$\Theta_2 \sim \mathrm{B}(1,1)$	$\Theta_3 \sim \mathrm{B}(1,1)$	$\Theta_4 \sim \mathrm{B}(1,1)$	$\langle \ell \rangle$
$\check{\mathrm{a}}^*$	11	10	10	9	0.026544
$\check{\mathrm{a}}^-$	10	10	10	10	0.027523
$\check{\mathrm{a}}^1_1$	40	0	0	0	0.178049

(b) Identical uniform priors, $n = 4, b = 40$.

	$\Theta_1 \sim \mathrm{B}(10,1)$	$\Theta_2 \sim \mathrm{B}(10,1)$	$\Theta_3 \sim \mathrm{B}(10,1)$	$\Theta_4 \sim \mathrm{B}(10,1)$	$\langle \ell \rangle$
$\check{\mathrm{a}}^*$	6	2	0	0	0.042095
$\check{\mathrm{a}}^-$	2	2	2	2	0.052592
$\check{\mathrm{a}}^1_1$	8	0	0	0	0.045254

(c) Identical skewed priors, $n = 4, b = 8$.

	$\Theta_1 \sim \mathrm{B}(4,2)$	$\Theta_2 \sim \mathrm{B}(5,3)$	$\Theta_3 \sim \mathrm{B}(1,1)$	$\Theta_4 \sim \mathrm{B}(3,2)$	$\langle \ell \rangle$
$\check{\mathrm{a}}^*$	3	0	3	2	0.079402
$\check{\mathrm{a}}^-$	2	2	2	2	0.088954
$\check{\mathrm{a}}^1_1$	8	0	0	0	0.107668
$\check{\mathrm{a}}^1_2$	0	8	0	0	0.118206
$\check{\mathrm{a}}^1_3$	0	0	8	0	0.101879
$\check{\mathrm{a}}^1_4$	0	0	0	8	0.110737

(d) Various different priors, $n = 4, b = 8$.

Table 2.1: Expected losses of various allocations in different circumstances. Definitions for $\check{\mathrm{a}}^*$, $\check{\mathrm{a}}^1$, and $\check{\mathrm{a}}^-$ can be found in Sections 2.4.1, 2.4.2, and 2.4.3, respectively. Note that for situations with identical coins, any permutation of a particular allocation will have the same loss.

2.4.3 Uniform Allocation

At the other end of the spectrum from SCA would be to flip each coin b/n times. A policy that produces this effect might look like

$$\pi^{-}(\mathbf{s}, b) = [(\sum\nolimits_i \gamma_{i1} + \gamma_{i2}) \mod n] + 1$$

This results in a uniform allocation, denoted $\breve{a}^{-}$. It represents flipping coin C_1, C_2, up to C_n and then beginning again with C_1 which we also call a *Round-Robin* policy. (Note that the order is arbitrary; all that matters is the end result of having an equal number of samples from each coin.)

Expected Loss of Uniform Allocation

It is possible to express the expected loss of a uniform allocation (Round-Robin policy) in a compact form if we assume uniform Beta priors over all of the coins. The more concise form allows for asymptotic analysis to examine the effect of varying the number of coins and budget on performance. We will use a few simple properties of probability distribution functions and cumulative density functions in this derivation.

Lemma 2. *If* $\{X_1, X_2, ..., X_n\}$ *are independent real-valued random variables,* $\mathrm{cdf}_i(x) = P(X_i \leq x)$ *and* $\mathrm{cdf}_{\max}(x) = P(\max\{X_1, X_2, ..., X_n\} \leq x)$*, then*

$$\mathrm{cdf}_{\max}(x) = \prod_{i=1}^{n} \mathrm{cdf}_i(x)$$

Proof. The probability $P(\max\{X_1, X_2, ..., X_n\} \leq x) = \mathrm{cdf}_{\max}(x)$ is the same as the conjunctive probability $P(X_1 \leq x \cap X_2 \leq x \cap ... \cap X_n \leq x)$. Because the X_i are independent, this probability is simply the product $\prod_{i=1}^{n} P(X_i \leq x) = \prod_{i=1}^{n} \mathrm{cdf}_i(x)$. □

Lemma 3. *If* X *is a continuous random variable that takes on values in* $[0, 1]$*, then*

$$\langle X \rangle = 1 - \int_0^1 \mathrm{cdf}(x)\, \mathrm{d}x$$

Proof. By definition,

$$\langle X \rangle = \int_0^1 x \cdot \mathrm{pdf}(x)\, \mathrm{d}x$$

Integrating by parts, we get

$$\begin{aligned}
\int_0^1 x \cdot \mathrm{pdf}(x)\,\mathrm{d}x &= \left[x \cdot \int \mathrm{pdf}(x)\,\mathrm{d}x - \int \left(\int_0^x \mathrm{pdf}(t)\,\mathrm{d}t\right)\,\mathrm{d}x\right]\Bigg|_0^1 \\
&= \left[x \cdot \mathrm{cdf}(x) - \int \mathrm{cdf}(x)\,\mathrm{d}x\right]\Bigg|_0^1 \\
&= x \cdot \mathrm{cdf}(x)\Big|_0^1 - \int_0^1 \mathrm{cdf}(x)\,\mathrm{d}x \\
&= [(1 \cdot \mathrm{cdf}(1) - 0 \cdot \mathrm{cdf}(0)] - \int_0^1 \mathrm{cdf}(x)\,\mathrm{d}x \\
&= 1 - \int_0^1 \mathrm{cdf}(x)\,\mathrm{d}x
\end{aligned}$$

□

Theorem 1. *The expectation* $\langle \Theta_{\max}|\mathbf{s}_0\rangle$ *is invariant over policies. Given a start state* $\mathbf{s}_0$ *and any policy* π*,*

$$\sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)} P(\mathbf{s}) \cdot \langle \Theta_{\max}|\mathbf{s}\rangle = \langle \Theta_{\max}|\mathbf{s}_0\rangle$$

Proof. (Sketch.) This equivalence is derived from the following identity of Beta distributions:

$$P_{\mathrm{B}}(x;\gamma_1,\gamma_2) = \frac{\gamma_1}{\gamma_1+\gamma_2} \cdot P_{\mathrm{B}}(x;\gamma_1+1,\gamma_2) + \frac{\gamma_2}{\gamma_1+\gamma_2} \cdot P_{\mathrm{B}}(x;\gamma_1,\gamma_2+1) \tag{2.9}$$

Using Lemmas 2 and 3 and Equation 2.9, it is easy to show that

$$\langle \Theta_{\max}|s_0\rangle = \hat{\theta}_i|\mathbf{s}_0 \cdot \langle \Theta_{\max}|(s_0|\gamma_{i1}\texttt{++})\rangle + (1-\hat{\theta}_i|\mathbf{s}_0) \cdot \langle \Theta_{\max}|(s_0|\gamma_{i2}\texttt{++})\rangle \tag{2.10}$$

which means that $\langle \Theta_{\max}|\mathbf{s}_0\rangle$ does not change when averaged over the possible outcomes of any action. Simple induction extends this result to possible outcomes of any policy. A complete proof of Equation 2.9 and the derivation proving Theorem 1 can be found in Appendix A. □

We have defined the expected loss of a particular *belief state* in Equation 2.7. The expected loss (or regret) of a *policy* is given by

$$\begin{aligned}
\ell(\pi,\mathbf{s}_0) &= \sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)} P(\mathbf{s}) \cdot \ell(\mathbf{s}) \\
&= \sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)} P(\mathbf{s}) \cdot \left[\langle \Theta_{\max}|\mathbf{s}\rangle - \hat{\theta}_{\max}|\mathbf{s}\right] \\
&= \sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)} P(\mathbf{s}) \cdot \langle \Theta_{\max}|\mathbf{s}\rangle - \sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)} P(\mathbf{s}) \cdot \hat{\theta}_{\max}|\mathbf{s} \\
&= \langle \Theta_{\max}|\mathbf{s}_0\rangle - \sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)} P(\mathbf{s}) \cdot \hat{\theta}_{\max}|\mathbf{s}
\end{aligned} \tag{2.11}$$

$\langle \Theta_{\max} | \mathbf{s}_0 \rangle$ represents the expected value of the random variable $\Theta_{\max} = \max\{\Theta_1, \Theta_2, ..., \Theta_n\}$, which, assuming uniform priors, has a cumulative density function

$$\begin{aligned} \mathrm{cdf}_{\max}(\theta) &= \prod_{i=1}^{n} \mathrm{cdf}_i(\theta) && (2.12) \\ &= \mathrm{cdf}_1(\theta)^n && (2.13) \\ &= \theta^n && (2.14) \end{aligned}$$

Equation 2.12 follows from Lemma 2, Equation 2.13 from the fact that the distributions of the Θ_i are identical, and Equation 2.14 from the fact that each Θ_i is uniformly distributed. (*i.e.*, $\forall i, \quad \mathrm{pdf}_i(\theta) = 1$, and therefore $\mathrm{cdf}_i(\theta) = \int \mathrm{pdf}_i(\theta)\, \mathrm{d}\theta = \int 1\, \mathrm{d}\theta = \theta$.)

Theorem 2. *If n random variables $\Theta_1 ... \Theta_n$ are all uniformly distributed on $[0, 1]$, and if $\Theta_{\max} = \max\{\Theta_1, \Theta_2, ..., \Theta_n\}$, then*

$$\langle \Theta_{\max} \rangle = \frac{n}{n+1} \tag{2.15}$$

Proof. Using the cumulative density of $\Theta_{\max}$ and Lemma 3, we can compute the value of $\langle \Theta_{\max} \rangle$:

$$\begin{aligned} \langle \Theta_{\max} \rangle &= 1 - \int_0^1 \mathrm{cdf}_{\max}(\theta)\, \mathrm{d}\theta \\ &= 1 - \int_0^1 \theta^n\, \mathrm{d}\theta \\ &= 1 - \frac{\theta^{n+1}}{n+1}\Big|_0^1 \\ &= 1 - \frac{1}{n+1} \\ &= \frac{n}{n+1} \end{aligned}$$

□

This demonstrates that value of $\langle \Theta_{\max} | \mathbf{s} \rangle$ can be calculated in constant time from the current belief state, but the second term of Equation 2.11, $\sum_{\mathbf{s} \in \mathcal{O}(\pi, \mathbf{s}_0)} P(\mathbf{s}) \cdot \hat{\theta}_{\max} | \mathbf{s}$ is a sum over all possible *outcomes* of a policy, of which there may be an exponential number. If we assume a Beta prior and conditional independence of coins, the probability distribution over outcomes is a product of Beta-Binomial distributions, each of which is uniform. (Uniformity of the distributions can be observed by setting all hyperparameters to 1 in Equation 2.5. *N.b.* these distributions are unlike binomial distributions, which cannot be uniform except for a

few degenerate cases when there are only one or two outcomes.) Therefore, the distribution $P(\mathbf{s})$ over outcomes is uniform, which gives

$$\sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)} P(\mathbf{s})\cdot\hat{\theta}_{\max}|\mathbf{s} = \frac{1}{|\mathcal{O}(\pi,\mathbf{s}_0)|}\sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)}\hat{\theta}_{\max}|\mathbf{s}$$

where $\mathcal{O}(\pi,\mathbf{s}_0)$ is the set of outcomes that can be reached by following π, and $\hat{\theta}_{\max}|\mathbf{s}$ represents the highest observed expected value in outcome $\mathbf{s}$, *i.e.*, the expected value of the coin we would report. Unfortunately, if we consider allocating k flips to each of n coins ($b = n\cdot k$), then $|\mathcal{O}(\breve{a}^-,\mathbf{s}_0)| = (k+1)^n$, which is exponential in the number of coins.

We can avoid summing over this exponential number of outcomes by noting that there are many outcomes that have the same value of $\hat{\theta}_{\max}|\mathbf{s}$. In fact, there are only $k+1$ different possible values of $\hat{\theta}_{\max}$. Furthermore, we can calculate the probability of ending up in any of the states with a particular value of $\hat{\theta}_{\max}$. We therefore re-factor the expectation and sum over all $k+1$ possible values of $\hat{\theta}_{\max}$ instead of over all $(k+1)^n$ possible outcomes of $\breve{a}^-$.

$$\langle\hat{\theta}_{\max}\rangle = \sum_{\hat{\theta}_{\max}\in\mathcal{V}} P(\hat{\theta}_{\max})\cdot\hat{\theta}_{\max} \tag{2.16}$$

Given uniform priors and $k = b/n$ flips for each coin, it is easy to see that

$$\mathcal{V} = \{(i+1)/(k+2)|i\in\{0,1,...,k\}\}$$

Therefore $P(\hat{\theta}_{\max})$ is a discrete distribution

$$P(\hat{\theta}_{\max}\le x) = \prod_{1\le j\le n} P(\hat{\theta}_j\le x)$$

As before, $\hat{\theta}_j$ represents the empirical expected value of coin j after we exhaust the budget. If we stipulate that $\hat{\theta}_j = (i+1)/(k+2)$ where i represents the number of heads seen from coin j (that is, $0\le i\le k$), then the distribution of $\hat{\theta}_j$ is uniform over $k+1$ possible outcomes, so we have

$$P(\hat{\theta}_j\le(i+1)/(k+2)) = \frac{i+1}{k+1} \tag{2.17}$$

it follows that

$$\begin{aligned} P(\hat{\theta}_{\max}\le(i+1)/(k+2)) &= \prod_{1\le j\le n} P(\hat{\theta}_j\le(i+1)/(k+2)) \\ &= P(\hat{\theta}_j\le(i+1)/(k+2))^n \\ &= \left(\frac{i+1}{k+1}\right)^n \end{aligned}$$

This gives the cumulative distribution function for the discrete variable $\hat{\theta}_{\max}$, from which we can derive the probability distribution function

$$\begin{aligned} P(\hat{\theta}_{\max} = (i+1)/(k+2)) &= P(\hat{\theta}_{\max} \leq (i+1)/(k+2)) \\ &\quad - P(\hat{\theta}_{\max} \leq i/(k+2)) \\ &= \left(\frac{i+1}{k+1}\right)^n - \left(\frac{i}{k+1}\right)^n \\ &= \frac{(i+1)^n - i^n}{(k+1)^n} \end{aligned}$$

Substituting this for Equation 2.16 we get

$$\begin{aligned} \sum_{\mathbf{s} \in \mathcal{O}(\pi, \mathbf{s}_0)} P(\mathbf{s}) \cdot \hat{\theta}_{\max} | \mathbf{s} &= \sum_{\hat{\theta}_{\max}} P(\hat{\theta}_{\max}) \cdot \hat{\theta}_{\max} \\ &= \sum_{i=0}^{k} P(\hat{\theta}_{\max} = (i+1)/(k+2)) \cdot (i+1)/(k+2) \\ &= \sum_{i=0}^{k} \frac{(i+1)^n - i^n}{(k+1)^n} \cdot \frac{i+1}{k+2} \end{aligned}$$

Therefore, given identical uniform priors, n coins and k flips for each coin, the expected loss associated with flipping each coin k times is given by

$$\begin{aligned} \ell(\breve{\mathbf{a}}^-, \mathbf{s}_0) &= \langle \Theta_{\max} \rangle - \sum_{\hat{\theta}_{\max}} P(\hat{\theta}_{\max}) \cdot \hat{\theta}_{\max} \\ &= \frac{n}{n+1} - \sum_{i=0}^{k} \frac{(i+1)^n - i^n}{(k+1)^n} \cdot \frac{i+1}{k+2} \end{aligned} \tag{2.18}$$

Asymptotic Examination

Equation 2.18 can be rewritten in a more convenient form

$$\begin{aligned} \ell(\breve{\mathbf{a}}^-) &= \frac{n}{n+1} - \sum_{i=0}^{k} \frac{(i+1)^n - i^n}{(k+1)^n} \cdot \frac{i+1}{k+2} \\ &= \frac{n}{n+1} - \frac{1}{(k+1)^n \cdot (k+2)} \sum_{i=0}^{k} [(i+1)^n - i^n] \cdot (i+1) \\ &= \frac{n}{n+1} - \frac{1}{(k+1)^n \cdot (k+2)} \\ &\quad \cdot \left[\sum_{i=0}^{k} [(i+1)^n \cdot (i+1)] - \sum_{i=0}^{k} i^n \cdot i - \sum_{i=0}^{k} i^n \right] \end{aligned}$$

We can then change the indices on the summations to obtain

$$\begin{aligned}
\ell(\breve{\mathrm{a}}^{-}) &= \frac{n}{n+1} - \frac{1}{(k+1)^n \cdot (k+2)} \\
&\quad \cdot \left[\sum_{i=1}^{k+1} [i^n \cdot i] - \sum_{i=1}^{k} i^n \cdot i - \sum_{i=1}^{k} i^n \right] \\
&= \frac{n}{n+1} - \frac{1}{(k+1)^n \cdot (k+2)} \cdot \left[(k+1)^n \cdot (k+1) - \sum_{i=1}^{k} i^n \right] \\
&= \frac{n}{n+1} - \left[\frac{k+1}{k+2} - \frac{\sum_{i=1}^{k} i^n}{(k+1)^n \cdot (k+2)} \right] \qquad (2.19)
\end{aligned}$$

We now make two observations about Equation 2.19. First, for $n = 1$ (*i.e.*, one coin only)

$$\begin{aligned}
\ell(\breve{\mathrm{a}}^{-}, \mathbf{s}_0) &= \frac{n}{n+1} - \left[\frac{k+1}{k+2} - \frac{\sum_{i=1}^{k} i^n}{(k+1)^n \cdot (k+2)} \right] \\
&= \frac{1}{1+1} - \left[\frac{k+1}{k+2} - \frac{\sum_{i=1}^{k} i}{(k+1) \cdot (k+2)} \right] \\
&= \frac{1}{2} - \left[\frac{k+1-\frac{k}{2}}{k+2} \right] \\
&= \frac{1}{2} - \frac{1}{2} \left[\frac{k+2}{k+2} \right] \\
&= 0
\end{aligned}$$

That is, if we only have one coin it must be the one with the highest expected value, and therefore we report it and achieve zero loss. Also, if we examine what happens as the budget k approaches infinity

$$\begin{aligned}
\lim_{k \to \infty} \ell(\breve{\mathrm{a}}^{-}) &= \lim_{k \to \infty} \frac{n}{n+1} - \left[\frac{k+1}{k+2} - \frac{\sum_{i=1}^{k} i^n}{(k+1)^n \cdot (k+2)} \right] \\
&= \frac{n}{n+1} - \lim_{k \to \infty} \left[\frac{k+1}{k+2} - \frac{\sum_{i=1}^{k} i^n}{(k+1)^n \cdot (k+2)} \right] \\
&= \frac{n}{n+1} - \left[1 - \lim_{k \to \infty} \frac{\sum_{i=1}^{k} i^n}{(k+1)^n \cdot (k+2)} \right] \qquad (2.20)
\end{aligned}$$

The last term of Equation 2.20 can be expressed in terms of a power formula $p(k)$ (a polynomial of degree $n+1$) in the numerator and a polynomial of degree $n+1$ in the denominator [Wei].

The coefficient of the leading term of $p(k)$ is $1/(n+1)$, and the coefficient of the leading term of the polynomial in the denominator is 1, so the limit is

$$\lim_{k \to \infty} \frac{\sum_{i=1}^{k} i^n}{(k+1)^n \cdot (k+2)} = \frac{1}{n+1}$$

$k \backslash n$	1	2	3	4	5
0	0.000000	0.166667	0.250000	0.300000	0.333333
1	0.000000	0.083333	0.125000	0.154167	0.177083
2	0.000000	0.055556	0.083333	0.102469	0.117284
3	0.000000	0.041667	0.062500	0.076563	0.087240
4	0.000000	0.033333	0.050000	0.061067	0.069333
5	0.000000	0.027778	0.041667	0.050772	0.057485

Table 2.2: Expected loss associated with a uniform allocation of k flips to each of n coins.

by $n+1$ applications of l'Hôpital's Rule. Equation 2.20 therefore simplifies to

$$\begin{aligned}
\lim_{k\to\infty} \ell(\breve{\mathrm{a}}^-) &= \frac{n}{n+1} - \left[1 - \lim_{k\to\infty} \frac{\sum_{i=1}^{k} i^n}{(k+1)^n \cdot (k+2)}\right] \\
&= \frac{n}{n+1} - \left[1 - \frac{1}{n+1}\right] \\
&= \frac{n}{n+1} - \left[\frac{n}{n+1}\right] \\
&= 0
\end{aligned}$$

This indicates that, as expected, as we accumulate more and more data, our information gets better and better, and so our expected loss goes to zero. Table 2.2 shows the numerical value of $\ell(\breve{\mathrm{a}}^-)$ for various k (number of flips per coin) and n (number of coins.) We also note that, for a fixed k, loss increases as n increases. Intuitively this is because the more coins we have, the more coins appear to be equally good (*i.e.*, show k heads out of k flips), and our inability to distinguish between them results in a higher probability of making a high-loss choice.

2.5 Empirical Results

It is feasible to compute the optimal loss (and the optimal policy) using Algorithm 1 for the range of only about $n \leq 10$ and $b \leq 10$. Figures 2.4(a) and 2.4(b) show the performance of the optimal strategy against some of the other algorithms on uniform priors and on skewed priors[8]. Athough we are concerned with loss only after the budget has been expended, the current loss is plotted at intermediate points to illustrate the behaviour of policies during data acquisition. The performances of Single Coin Lookahead and Biased-Robin are very close to that of optimal, while Round-Robin lags behind considerably[9]. Because of the

[8]Each point is the average of 4000 trials. Initially in each trial, every coin's actual head probability is drawn from the prior, and then tossed by the algorithms.

[9]We remark that the empirical loss from Round-Robin with $n = 10, b = 10$ and uniform priors shown in Figure 2.4(a) is very close to the number predicted by Equation 2.18, which is ≈ 0.24275. Also, Figure 2.5(a) shows an empirical loss for Round-Robin with $n = 10$, $b = 40$ close to the theoretical value of ≈ 0.09468.

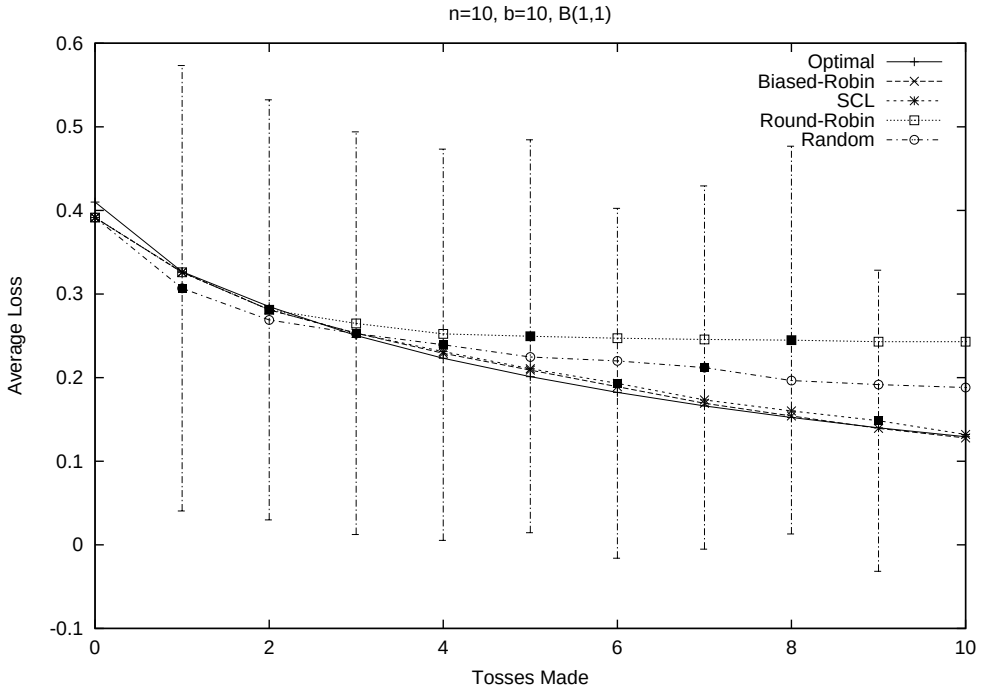
nature of the experiment wherein a new set of coins is drawn for each of 4000 trials and then given to the algorithms to flip, the variance of the loss function is very high, as is indicated by the error bars.

We have made similar observations on other types of identical priors on larger problem sizes (*e.g.*, Figures 2.5(a), 2.5(b), and 2.5(c), which show performance with a budget of 40). In Figures 2.5(b) and 2.5(c), we observe that the loss of the Round-Robin policy has a step-like shape, with sharp reductions in loss occurring at points where there is sufficient budget to 'wrap around' and flips some coins one more time *i.e.*, the maximum number of flips to *any coin* increases by one. We regard this as further evidence that it may be useful to allocate more flips to fewer coins, even though some may be flipped very seldom or not at all.

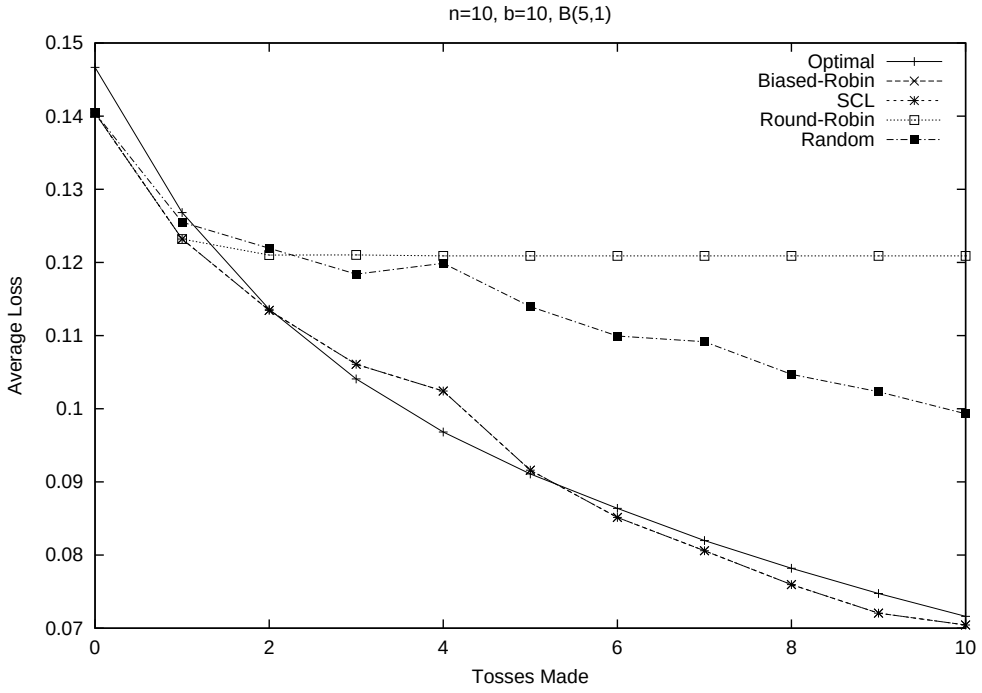
The Biased-Robin and Single Coin Lookahead policies consistently outperform the others to varying degrees depending on priors. The relative difference in performance increases with priors skewed toward higher head probabilities (Beta densities $\mathrm{B}(5,1)$ and $\mathrm{B}(10,1)$ in the figure), and with increasing n (see [Web]). Conversely, as the probability of good coins decreases in general (as with a $\mathrm{B}(1,10)$ prior; see Figure 2.5(c)), performance becomes generally bad for all policies. We believe the cause of the poor performance of the greedy policy is its inability to make a definitive decision in many situations. Frequently, looking only one step ahead, two coins may have the *exact same* expected loss. This causes the greedy policy to toss many arbitrary coins that may be inefficient choices when considering the budget. For example, with uniform priors, as soon as some coin gives a head, a single toss of any coin does not change the expected highest mean and therefore each coin is as good as any other. Although SCL considers single-coin allocations only, it performs well since it takes the whole budget into account which allows it to discriminate between coins better than the greedy policy.

Again, the Coins Problem is a gross simplification of the budgeted learning problem: we chosen the simplest distributions, assumed unit costs[10], and made independence assumptions. Nonetheless, we see that even a simple problem such as this requires careful resource allocation to obtain good performance.

[10]It is very simple to extend the coin problem to have nonuniform costs. When constructing an optimal policy or finding the best SCL score, we only admit outcomes that respect the budget, *i.e.*, outcomes for which the total cost of the actions that lead to them, stays less than b.

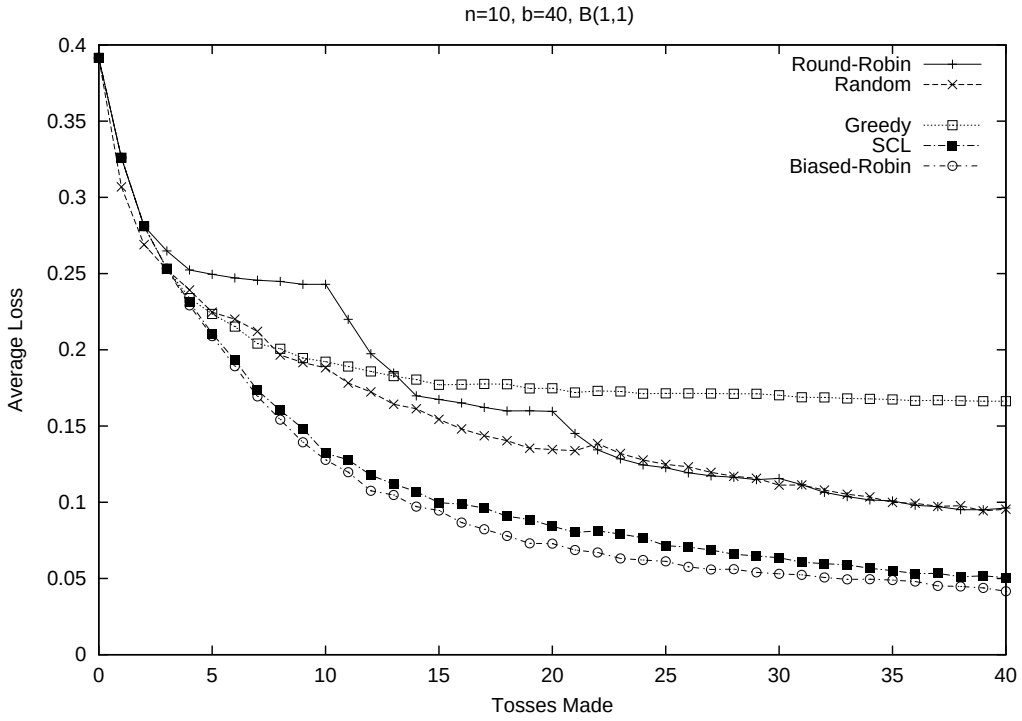


(a) 10 Coins, $\Theta_i \sim \mathrm{B}(1,1)$

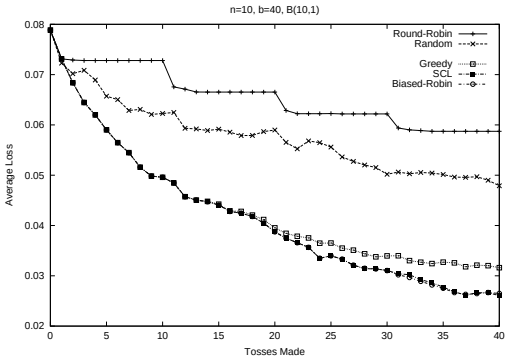


(b) 10 Coins, $\Theta_i \sim \mathrm{B}(5,1)$

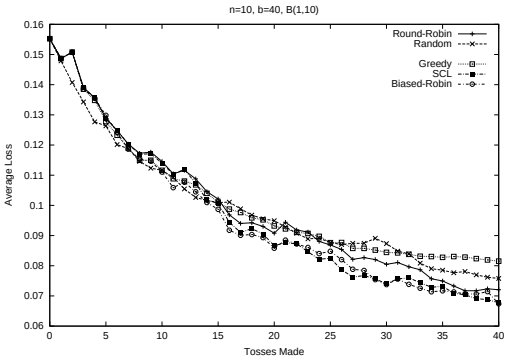
Figure 2.4: Performance of the optimal policy versus several other policies on 10 coins with a budget of 10. (Note different Average Loss axis scales.)



(a) 10 Coins, $\Theta_i \sim \mathrm{B}(1,1)$



(b) 10 Coins, $\Theta_i \sim \mathrm{B}(10,1)$



(c) 10 Coins, $\Theta_i \sim \mathrm{B}(1,10)$

Figure 2.5: These figures show performance (average loss) of various policies on 10 coins. Figure 2.5(a) shows the performance of non-optimal policies with a maximum budget of 40 and uniform priors. Figure 2.5(b) shows the performance of non-optimal policies with skewed priors $(\gamma_{i1}|\mathbf{s}_0, \gamma_{i2}|\mathbf{s}_0) = (10,1)$ and a maximum budget of 40. Figure 2.5(c) shows performance when $(\gamma_{i1}|\mathbf{s}_0, \gamma_{i2}|\mathbf{s}_0) = (1,10)$. Note that different plots have different Average Loss axis scales.

Chapter 3

The Budgeted Naïve Bayes Problem

We now examine the idea of budgeted learning a Naïve Bayes classifier.

Problem 2. The Budgeted Naïve Bayes Problem. *We are given a pool $\mathcal{P}$ of training instances, where each instance is characterized by a set of n features $\mathbf{X} = \{ X_1, X_2, ..., X_n \}$, as well as a class label Y. Initially, our learner R knows only the class* labels *of a large set of training instances;* n.b., *R does not know the value of any feature for any instance.*

X_1	X_2	...	X_n	Y
?	?	?	?	y_1
?	?	?	?	y_2
?	?	?	?	y_1
⋮	⋮	⋮	⋮	⋮

Figure 3.1: Initial state of $\mathcal{P}$. No feature values are known, all class labels are known.

R begins with a known, fixed total budget $b \in \mathbb{R}^{\geq 0}$, and knows the costs $c_i = c(X_i)$ to obtain a value of feature X_i from any instance. At each time, R can, at cost $c(X_i)$, obtain the value of the i-th feature of an instance with any particular label y. (Hence, R can explicitly request, say, the previously unknown X_1 value of a $Y = y_1$ instance.) We assume that the cost of an action is independent of its outcome, and independent of j. E.g., the cost of purchasing X_1 is the same whether the value observed is x_{11} or x_{13}, and the same whether we purchase it from a $Y = y_1$ or a $Y = y_2$ instance. R continues until exhausting its budget. At that point, R returns a classifier. Its goal is to obtain a classifier whose expected loss $\ell(R)$ is minimum among all those learned from any pool of training data whose total cost is less than b.

We will focus on the case where R is a Naïve Bayes (NB) *classifier, which is a belief*

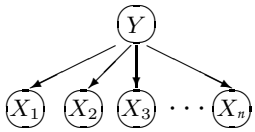


Figure 3.2: Naïve Bayes Structure

net with a structure that assumes that feature values are conditionally independent given the class label y [DHS01]; see Figure 3.2. The parameters of the classifier are estimated from the event counts observed in the purchased data and the priors using m-estimates, for example see [Mit97]. When presented with a new feature vector $\mathbf{x}$*, Bayes' Theorem is used to compute* $\arg\max_y \ P(Y = y \,|\mathbf{X} = \mathbf{x})$*, which is returned as the most likely classification given* $\mathbf{x}$ *and the model.*

3.1 Modeling the Budgeted Naïve Bayes Problem

We will model the Budgeted Naïve Bayes Problem as a Markov Decision Process over belief state space, much as we did for the Coins problem. We again define random variables Θ_{ijk}, this time to describe the posterior distributions of each feature X_i given each class label $Y = y_j$.

In this case, we also need parameters Θ_j representing the unconditional probabilities $P(Y = y_j)$. Since each instance in our pool of data has its class label, these parameters can be estimated in the same way as for normal (non-budgeted) Naïve Bayes learning. Once initially estimated, these parameters do not change during the data purchasing process.

3.1.1 Bayesian Updating

We again use the Bayes updating procedure to define distributions that are modified as data accumulates. In this problem setting, however, we need to handle features that may take on more than two different values, so we will assume that the conditional distribution of each feature X_i is a discrete, multi-valued generalization of the Bernoulli distribution[1] with parameters

$$\Theta_{ijk} = P(X_i = x_{ik} | Y = y_j)$$

These parameters are in turn drawn from *Dirichlet* conjugate priors [Hec95] with parameters γ_{ijk}, and each state $\mathbf{s}$ corresponds to the collection of these γ_{ijk} parameters. If the feature

[1]The distribution of event counts over multiple iid samples from X_i is drawn from a *Multinomial* distribution, which is a generalization of the Binomial distribution.

X_i has r values, then its multinomial parameters

$$\Theta_{ij} \;=\; \{\, \Theta_{ij1}, \ldots, \Theta_{ijr} \,\} \sim \mathrm{Dir}(\gamma_{ij1}, \ldots, \gamma_{ijr})$$

are Dirichlet distributed, with parameters $\gamma_{ijk} > 0$. The density function is given by

$$p(\theta_{ij1}, \theta_{ij2}, ..., \theta_{ijr}) \;=\; \frac{\prod_{k=1}^{r} \Gamma(\gamma_{ijk})}{\Gamma(\sum_{k=1}^{r} \gamma_{ijk})} \cdot \prod_{k=1}^{r} \theta_{ijk}^{\gamma_{ijk}-1}$$

subject to $\sum_k \theta_{ijk} = 1$.

These distributions are direct extensions of the Bernoulli and Beta distributions we used to model the Coins Problem, and are updated in much the same way, using priors and observation counts. (Setting $\gamma_{ij\cdot}|\mathbf{s}_0 = \{\, 1, 1, ..., 1 \,\}$ represents the uniform prior.) *E.g.*, the 3-valued feature X_7 associated with the class having its first value $Y = y_1$, might be distributed as $\Theta_{7,1,\cdot}|s = \{\, \Theta_{7,1,1}|s, \Theta_{7,1,2}|s, \Theta_{7,1,3}|s \,\} \sim \mathrm{Dir}(3, 8, 2)$. Our learning process must perform a sequence of actions $\{\mathrm{a}_{ij}\}$, where action a_{ij} represents requesting the value of feature X_i from an instance with label $Y = y_j$. (*E.g.*, a_{71} represents requesting the value of feature X_7 from a $Y = y_1$ instance. Thanks to our Naïve Bayes independence assumptions, it does not matter *which* $Y = y_1$ instance this value comes from.) If we later request the X_7 value of a $Y = y_1$ instance (*i.e.*, take action $a_{7,1}$), and observe the second value $x_{7,2}$, then the posterior distribution in the new state s' will be $\Theta_{7,1,\cdot}|s' \sim \mathrm{Dir}(3, 8+1, 2) = \mathrm{Dir}(3, 9, 2)$. In general, if $\Theta_{ij}|s \sim \mathrm{Dir}(\gamma_{ij1}, \ldots, \gamma_{ijr})$, then after observing a value, which is say the k-th value of X_i (x_{ik}), the new distribution is $\Theta_{ij}|s' \sim \mathrm{Dir}(\gamma_{ij1}, \ldots, \gamma_{ijk}+1, \ldots, \gamma_{ijr})$. Note that the expected value of this $\Theta_{7,1,2}|s'$ variable is $\hat{\theta}_{7,1,2}|s' = 9/(3+9+2) = 9/14$. Given our current knowledge, the probability of incrementing the k-th value $\gamma_{ijk}|s$, when purchasing a value for X_i for the j-th label, is $\hat{\theta}_{ijk}|s$, which defines our state transition probability.

3.2 Markov Decision Process Framework

We again have *belief states* $\mathbf{s} \in \mathrm{S}$ which are collections of Dirichlet parameters, *actions* $\mathrm{a}_{ij} \in \mathrm{A}$ which describe the process of purchasing a feature value to get to a new belief state, and a state transition probability distribution.

The final component of our budgeted learning MDP is the loss function which is only evaluated when the budget has been expended (*i.e.*, loss is 0 for all states except final states). At that time a loss of $\ell(\mathrm{NB}(\mathbf{s}))$ is received where ℓ is a loss function of the Naïve Bayes model induced by the parameters of the state $\mathbf{s}$. Possible loss functions are described in

RoundRobin(pool $\mathcal{P}$, budget $\mathcal{b}$)
$\underline{c} \leftarrow \min_i c(X_i)$
$i \leftarrow 0$
while $\mathcal{b} > \underline{c}$ **do**
  if $\mathcal{b} > c(X_{i+1})$ **then**
    Draw j from $P(Y = y_j)$
    do_action$(a_{i+1,j})$
    Update $\mathcal{b}$, $\mathbf{s}$
  end if
  $i \leftarrow (i+1) \bmod n$
end while

{ *For RoundRobin and BiasedRobin, j is drawn from the prior class distribution.* }

(a) Round-Robin

BiasedRobin(pool $\mathcal{P}$, budget $\mathcal{b}$)
$\underline{c} \leftarrow \min_i c(X_i)$
$\ell_{\text{old}} \leftarrow \ell(\text{NB}(\mathbf{s}))$
$i \leftarrow 0$
while $\mathcal{b} > \underline{c}$ **do**
  if $\mathcal{b} > c(X_{i+1})$ **then**
    Draw j from $P(Y = y_j)$
    do_action$(a_{i+1,j})$
    Update $\mathcal{b}$, $\mathbf{s}$
    $\ell_{\text{new}} \leftarrow \ell(\text{NB}(\mathbf{s}))$
  end if
  if $\ell_{\text{new}} \geq \ell_{\text{old}}$ **then**
    $i \leftarrow (i+1) \bmod n$
  end if
  $\ell_{\text{old}} \leftarrow \ell_{\text{new}}$
end while

(b) Biased-Robin

SFL(pool $\mathcal{P}$, budget $\mathcal{b}$)

$\underline{c} \leftarrow \min_i c(X_i)$
while $\mathcal{b} > \underline{c}$ **do**
  $a_{ij} \leftarrow \arg\min_{i,j} \langle \ell(\text{NB}(\mathbf{s}|\check{\text{a}}^1_{ij})) \rangle$
  do_action(a_{ij})
  { *i.e., purchase value of X_i when $Y = y_j$* }
  Update $\mathcal{b}$, $\mathbf{s}$
end while
return(NB($\mathbf{s}$))

{ *Replacing $\check{\text{a}}^1_{ij}$ with a_{ij} gives the greedy policy.* }

(c) Single Feature Lookahead (SFL) and Greedy

Figure 3.3: Policies

Section 3.5, and include such measures as 0/1 error, GINI index, and entropy [HTF01]. We can now define policies that are intended to deliver reduced expected loss.

3.3 Policies

This section first indicates the complexity of finding the *optimal* policy, then outlines a number of plausible policies that we have implemented.

3.3.1 Optimal Policy

As we saw in Section 2.3.1, it is possible, given an MDP, to compute an optimal deterministic policy π^* that will result in the minimum expected loss. This policy can be found in a way analogous to Algorithm 1, but in principle it can be found by any standard MDP solution method (*i.e.*, value iteration, policy iteration, linear programming) in time polynomial in the size of the state space |S|. Unfortunately we again have the problem that the state space grows exponentially in $\mathcal{b}$, precluding exact computation for problems of an interesting size if we are to rely on an exhaustive state tree expansion. In fact, this problem is also NP-hard under different feature costs, a result inherited from [MLG03] (see also [Web]). In light of

this, we now examine several simpler, tractable policies that, while not optimal, improve on naïve approaches.

3.3.2 Greedy Loss Reduction

One common technique used in active learning is to calculate the expected loss of taking an action a_{ij} from the current belief state, given by

$$\langle\ \ell(\mathrm{NB}(\mathbf{s}))\,|\,\mathrm{a}_{ij}\ \rangle\ =\ \sum_{\mathbf{s}'\in\mathrm{S}} P(\,\mathbf{s}'\,|\,\mathbf{s},\mathrm{a}_{ij}\,)\cdot\ell(\mathrm{NB}(\mathbf{s}'))$$

A simple greedy policy is to evaluate these expectations for each possible action from the current state, and perform the action that has the lowest expected loss if we had to report a classifier immediately after taking that action. This technique has the advantage of directly minimizing the error of the resulting classifier, but we show that this greedy, single-step policy is outperformed by other policies that use deeper lookahead, as was the case for the Coins Problem, see Figure 3.5(a).

3.3.3 Biased-Robin

While examining the Coins Problem, we developed the Biased-Robin heuristic policy. We extend this policy to the Naïve Bayes problem as follows: Repeatedly take an action (*i.e.*, purchase feature X_i.) as long as it continues to reduce our current loss. As soon as an action results in an increase in our estimate of the loss according to the current model of the input distribution, we begin taking the next action. Note there is an important difference between this policy and the Greedy Loss Reduction policy: The Greedy policy chooses the next action that is expected to minimize loss. Biased-Robin either chooses the same action if it just caused our current loss to decrease, otherwise it begins taking the next action.

Again, the Biased-Robin policy is based on the Round-Robin or Uniform policy, discussed in Section 3.4.2. See Figure 3.3(b) for a more detailed description.

3.4 Allocations

Adapting the work of Section 2.4, we make the following definition: An *allocation* is an array of integers that describes the number of times a feature's value is purchased in conjunction with a certain class label; *i.e.*, the number of times action a_{ij} is executed. For example, under allocation ă, $\breve{\mathrm{a}}_{32} = 4$ would mean that the value of feature X_3 is purchased 4 times

from distinct tuples where $Y = y_2$. With normalized uniform costs ($c(X_i) = 1$ for all X_i) an allocation can be viewed as an integer *composition*.

The expected loss of executing the actions of an allocation $\breve{\mathrm{a}}$ in the current state $\mathbf{s}$ is given by

$$\langle \ell(\mathrm{NB}(\mathbf{s})) \,|\, \breve{\mathrm{a}} \rangle \quad = \quad \sum_{\mathbf{s}' \in \mathrm{S}} P(\,\mathbf{s}' \,|\, \mathbf{s}, \breve{\mathrm{a}}\,) \cdot \ell(\mathrm{NB}(\mathbf{s}')) \tag{3.1}$$

where

$$|\{\mathbf{s}' \in \mathrm{S} : P(\,\mathbf{s}' \,|\, \mathbf{s}, \breve{\mathrm{a}}\,) > 0\}| = \prod_{i,j} \binom{\breve{\mathrm{a}}_{ij} + |X_i| - 1}{\breve{\mathrm{a}}_{ij}}$$

$|X_i|$ represents the number of different values feature X_i may take on. Again, enumerating all possible states resulting from an allocation is an integer composition enumeration problem with a straightforward solution, except that there are an exponential number of compositions.

3.4.1 Single Feature Lookahead (SFL)

Although the Greedy policy is provably suboptimal, it is tractable since there are not too many actions to evaluate. Our goal is to incorporate knowledge of the budget when scoring potential actions while retaining this tractability. We accomplish this by formulating a method that introduces lookahead without expanding the entire state space that is derived from the SCL policy of Section 2.4.2.

Again, we consider allocations where $\breve{\mathrm{a}}_{ij} = \lfloor \mathcal{b}/c(\mathrm{a}_{ij}) \rfloor$ for one (i, j) and $\breve{\mathrm{a}}_{ij} = 0$ elsewhere. We denote this allocation $\breve{\mathrm{a}}^1_{ij}$. It represents spending the entire budget on the single action a_{ij}, and has a state space of size

$$|\{\mathbf{s}' \in \mathrm{S} : P(\mathbf{s}'|\mathbf{s}, \breve{\mathrm{a}}^1_{ij}) > 0\}| = \binom{\breve{\mathrm{a}}^1_{ij} + |X_i| - 1}{\breve{\mathrm{a}}^1_{ij}}$$

For a feature with two possible values, for example, there are only $2\lfloor \mathcal{b}/c(\mathrm{a}_{ij}) \rfloor$ distinct states. The probability of reaching any one of these states is given by [Hec95]

$$P(\mathbf{s}'|s, \breve{\mathrm{a}}^1_{ij}) = \frac{\Gamma(\sum_k \gamma_{ijk}|\mathbf{s})}{\Gamma(\sum_k \gamma_{ijk}|\mathbf{s}')} \prod_k \frac{\Gamma(\gamma_{ijk}|\mathbf{s}')}{\Gamma(\gamma_{ijk}|\mathbf{s})}$$

Our Single Feature Lookahead (SFL) operates as follows: For all i and j, compute the expected value of the loss of $\breve{\mathrm{a}}^1_{ij}$ as defined in Equation (3.1). We find the action with the minimum loss and perform it **once**, update the belief state and budget, and repeat. The algorithm is described in Figure 3.3(c).

This policy, like the greedy policy in Section 3.3.2, is suboptimal. However, the SFL score of an action will be influenced by additional factors that affect the mobility of a distribution.

This mobility is affected by a distribution's current belief state (distributions with smaller parameters are more mobile), the cost of an action (the distributions of cheaper actions are more mobile) and by the remaining budget (more budget means more mobility for all distributions.)

3.4.2 Uniform Policies

We again examine as a baseline the Round-Robin scheme where features are queried sequentially, regardless of outcomes; see Figure 3.3(a). We purchase feature X_1, then X_2, etc. If action costs are uniform, this is equivalent to following a *uniform allocation* policy.

The preceding description really only specifies half of each action to take by identifying which *feature* to purchase. The NB problem adds another dimension to actions: What should the *class label* be of the instance the feature comes from? We could simply cycle through class labels as we cycle through feature values, but this would not reflect the input distribution. (It would be equivalent to oversampling and undersampling in such a way as to force a uniform distribution over Y.) We have chosen instead to draw the class label from the distribution $P(Y)$ initially estimated.

Another simple policy when action costs are nonuniform would be to spend b/n on each feature, purchasing more expensive features fewer times. (Hence, ask for around $b/(n \cdot c(X_i))$ values of each X_i.) We call this a *uniform expenditure* policy.

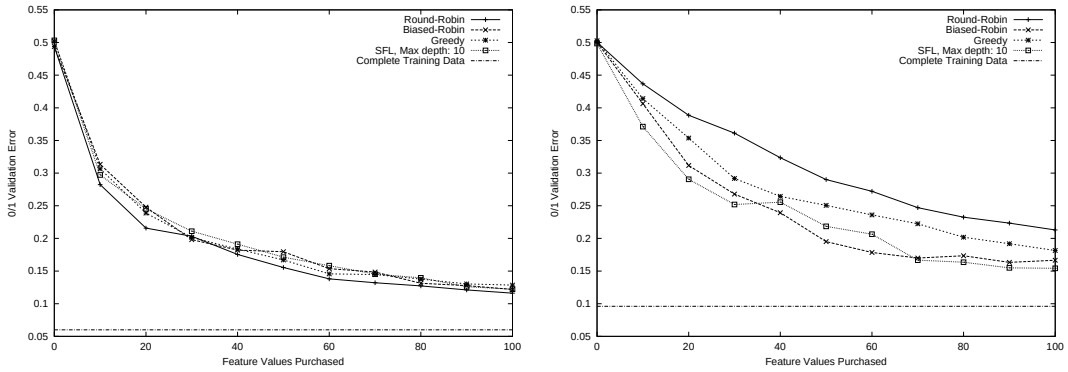
We will use the uniform allocation policy as a baseline for empirical performance evaluation.

3.5 Empirical Results

We will use the GINI index of the NB classifier as our loss function ℓ for choosing the actions of the greedy and SFL policies [HTF01]:

$$\ell_{\text{GINI}}(\,\text{NB}(\mathbf{s})\,) \;=\; \sum_{y \in Y} \sum_{\mathbf{x} \in \mathbf{X}} P(\mathbf{x}) P(y|\mathbf{x})(1 - P(y|\mathbf{x}))$$

Here, $\mathbf{X}$ represents the space of all possible inputs to the classifier, *i.e.*, the domain of the input distribution. Since this space has dimension n, we will sample to compute these loss values in practice. We have chosen the GINI index to guide action selection because we found that, being continuous, smooth, and nonlinear, it is more sensitive than 0/1 error to the small changes in the NB distribution caused by a single action. (Because 0/1 error is



(a) Uniform Distributions

(b) Nonuniform Distributions

Figure 3.4: Performance on synthesized data. 0/1 error on a validation set consisting of 20% of the complete data set. Errors are averages of 50 trials.

piecewise constant, certain actions do not change its expected value at all.) We have found that both GINI and entropy are similar in this respect.

Although we use the GINI index to choose actions, we have plotted 0/1 cross validation error in Figures 3.4, 3.5, 3.6, and 3.7 (discussed below). This gives a more useful measure of how well the classifiers learned under a budget are performing. In the interest of saving computational time, we have estimated some quantities needed in the greedy and SFL calculations by sampling. (Generating iid tuples from a belief net is trivial, especially for a Naïve Bayes.) Also, in the case of SFL, we look as deep as either the remaining budget or a fixed maximum indicated by "Max-depth", whichever is smaller. For example, with a budget $\mathcal{b} = 300$ and a max depth of 80, SFL will look ahead 80 purchases at each step until it expended all but 79 purchases (*i.e.*, until purchase 221) at which point it will look 79 ahead, then 78, etc. Adjusting this Max-depth allows us to examine the effects of varying degrees of lookahead.

3.5.1 Synthesized Data

Our initial experiments involve data synthesized from Naïve Bayes distributions to test our policies in a setting where the conditional independence assumptions are true. In these experiments, all the feature and class variables are Boolean, with $P(Y = y_1) = P(Y = y_2) = 0.5$. Each experiment represents an average over 50 trials. In each trial, a Naïve Bayes model with 10 features is generated from defined priors, and the model is used to generate 1000 iid instances, of which the first 80% are used for training and the remainder

are used to compute the 0/1 cross-validation error. (Because the true distribution is known, 0/1 error could be computed exactly; however, cross validation error was used so that the error computation is the same for our synthesized data as for the UCI datasets described below.) The vertical axis shows the 0/1 error of the model trained by the various algorithms after a number of purchases. The "Complete Training Data" line is the 0/1 cross-validation error of the Naïve Bayes model trained on all of the training data.

The two experiments differ in the priors from which the Naïve Bayes model is generated. In the first experiment (Figure 3.4(a)), under each class, each feature's multinomial parameters are drawn from a uniform distribution; $\{ \Theta_{ij\cdot} \} \sim \mathrm{Dir}(1,1)$. Therefore, all features are discriminative to varying degrees. We observe that the performances of the algorithms are all comparable, and there is nothing to be gained from selective querying (*i.e.*, even Round-Robin works well). The reason for the comparable performance of the algorithms is basically that purchasing any feature is expected to reduce the loss of the whole NB model somewhat, but highly discriminative features are so rare that it does not pay to hunt for them.

Figure 3.4(b) displays the other extreme where all features except one are irrelevant. In this experiment, each feature's parameters are drawn from a uniform (Dir(1,1)) distribution *independently* of the class. One feature, X_i, chosen at random is selected to be discriminative; in particular we set $P(\, X_i = x_{i1} \,|\, Y = y_1 \,) = 0.9$ and $P(\, X_i = x_{i1} \,|\, Y = y_2 \,) = 0.1$. We observe that lookahead (with Max-depth 10) and Biased-Robin algorithms significantly outperform Round-Robin in this scenario, requiring only about half as many purchases to obtain the same error level. These policies are capable of identifying the most promising features and obtaining better estimates of their posteriors, which improves performance. We can increase this difference in performance by increasing the discrimination level of the relevant feature (*e.g.*, the extreme case would be $P(\, X_i = x_{i1} \,|\, Y = y_1 \,) = 1.0$ and $P(\, X_i = x_{i1} \,|\, Y = y_2 \,) = 0$) and by increasing the number of irrelevant features.

3.5.2 UCI Data

For a less contrived test bed, we have chosen several datasets from the UCI Machine Learning Repository [BM98]. These plots show averaged validation error of the policies on a holdout set (20% of the dataset) on the mushroom, nursery, and votes datasets. Each point is an average of 50 trials where in each trial a random balanced partition of classes was made for training and validation. The five-fold cross-validation error of a standard (non-budgeted)

Naïve Bayes learner with the whole training set available is also shown ("Complete Training Data").

While we are really only interested in performance after the entire budget has been expended, we have plotted performance at intermediate points during learning to examine the behaviour of the different policies as feature values are being acquired.

The mushroom dataset is a binary class problem (poisonous vs. edible), with 22 features, 8124 instances, and a positive class probability of 0.52. One of the features, feature 5, is a very discriminative 10-valued feature, while others are less discriminative [Hol93]. Figures 3.5(a) and 3.5(b) show the performance of the different policies.

Figure 3.5(a) represents what we imagine to be a typical application of the policies discussed in this paper. The budget has been set at 100, and we allow SFL a "Max-depth" of 100, meaning that it always looks ahead as far as its remaining budget. Here we see that the contingent policies (*i.e.*, Biased-Robin, Greedy, and SFL) outperform the simplistic Round-Robin. Of the contingent policies, SFL, which is the only policy to use knowledge of the budget in decision making, performs best. It is obvious from the error bars that there is a great deal of variance over the 50 trials, which is an effect of learning from a very small amount of data. (Over the fifty trials we are likely to get some partitions of data that are very informative and others that are not.) Variance in the validation error of the different policies is comparable; however it appears that for larger amounts of data (80 feature values and up) the standard deviation of SFL is approximately half that induced by Round-Robin. We suspect that this is because SFL uses incoming data and the remaining budget to try to make the best of the current sample at hand, whereas Round Robin cannot adapt if a particular sample gives a poor estimate for a particular feature. This behaviour of variance is similar for other datasets; bars are shown only on this plot for clarity.

In Figure 3.5(b) again we see that the adaptive policies perform best, and we also see the effect of varying the degree of lookahead, with Max-depth 30 SFL dominating earlier in the run and Max-depth 80 SFL performing best later. This plot is illustrative of the effect of altering lookahead depth as it shows a 'more greedy' policy with shallower lookahead performing well initially before it is bested by a more farsighted policy. This is an indication that it is important to match the depth of lookahead to the actual budget.

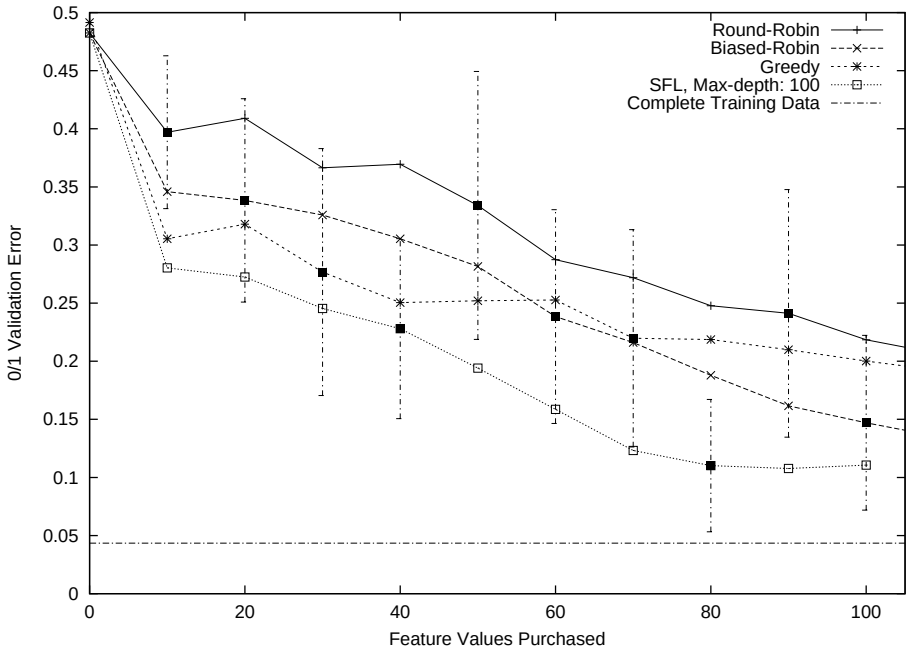
Further illustration of this is shown in the Nursery data. Figure 3.6(a) illustrates a similar phenomenon of shallow lookahead performing well early on, as Max-depth 10 SFL performs better than Max-depth 30 SFL earlier in the purchasing phase. Figure 3.6(b)

illustrates the difference in performance caused by by assuming a different *budget* for SFL when the Max-depth is kept the same. After 50 purchases, Max-depth 30 SFL with an assumed budget of 300 (■) performs significantly worse than Max-depth 30 SFL with an assumed budget of 50 (○). As these policies approach the 50 purchase mark, ○ is looking ahead to a *total budget* 50 purchases (*i.e.*, at step 49 it is looking 1 puchase ahead), but ■ is still looking 30 purchases ahead at each step (*i.e.*, at step 49 it is still looking 30 purchases ahead) which results in a performance hit at the 50 purchase mark. The nursery dataset (Figures 3.6(a) and 3.6(b)) is a five class problem with nine features that can take on between two and five values. The relative performances of the policies are closer to each other, but their behaviour is similar to Figure 3.5(b). Regardless of depth, SFL is capable of picking out relevant features: Out of the 300 purchases, feature 5 is bought by SFL an average of 75 times, while a non-discriminative feature such as feature 18 is bought an average of only 2 times. For some budgets, the 0/1 error of SFL is nearly half that generated by Round-Robin.

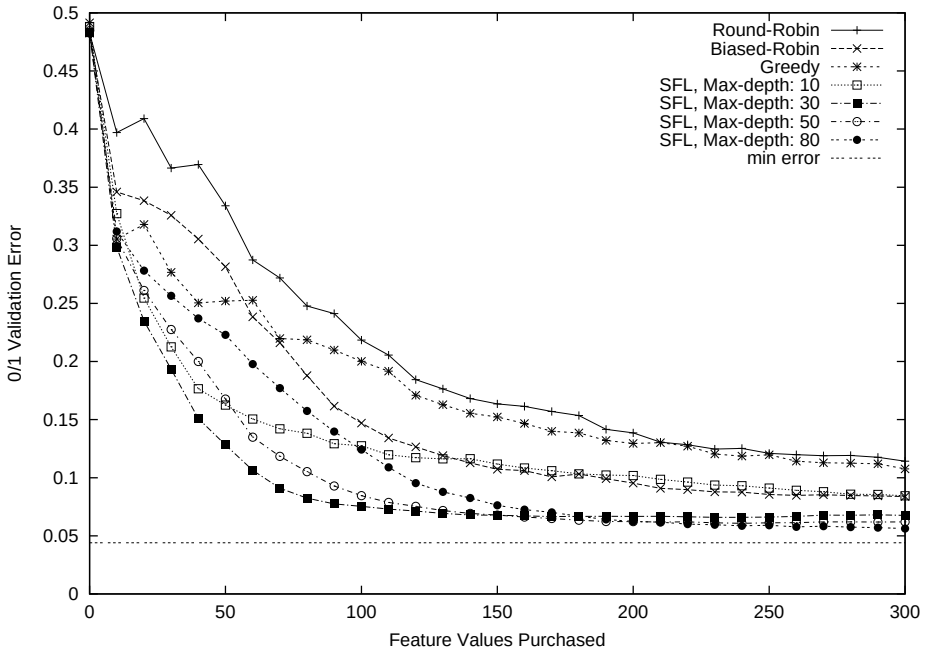
The votes dataset (Figures 3.7(a) and 3.7(b)) is a binary class problem (democrat vs. republican), with 16 binary features, 435 instances, and a positive class probability of 0.61. In the votes dataset, there is a high proportion of discriminative features, and we observe that all policies within relatively few purchases reduce the error to the minimum possible for a Naïve Bayes classifier. In fact, because of independence assumption violations, it is possible for selective policies to perform better than a NB classifier trained on the whole data set, as these policies are doing a form of feature selection: when beginning from uniform priors, features that are never queried will have no influence on the computation of the likelihood of a class label, and therefore will not 'interfere' with the prediction of features that are queried many times. Therefore, if discriminative features are found and are are queried extensively while other inferior features are ignored, performance will be better than if these noisy or inferior features are allowed to sway the classifier's decision. Again, for the SFL policy, the assumed budget is significant: as shown in Figure 3.7(b), the performance of SFL after 50 purchases is better when the budget is assumed (correctly) to be 50 than when it is assumed to be 300 (*i.e.*, when it looks ahead farther than it should). Other policies do not take the budget into account.

We have observed the same overall patterns on several other datasets that we have tested the policies on so far (CAR, DIABETES, CHESS, BREAST): the performance of SFL is superior or comparable to the performance of other policies, and Biased-Robin is the best

algorithm among the budget insensitive policies; see [Web] for additional details. Run times for Round-Robin and Biased-Robin are very short, taking only seconds, with the greedy policy taking slightly longer. Run times for SFL took the longest, and were on the order of minutes. (Its runtime is $\in O(n \cdot \max_i |X_i| \cdot |Y| \cdot b)$.)

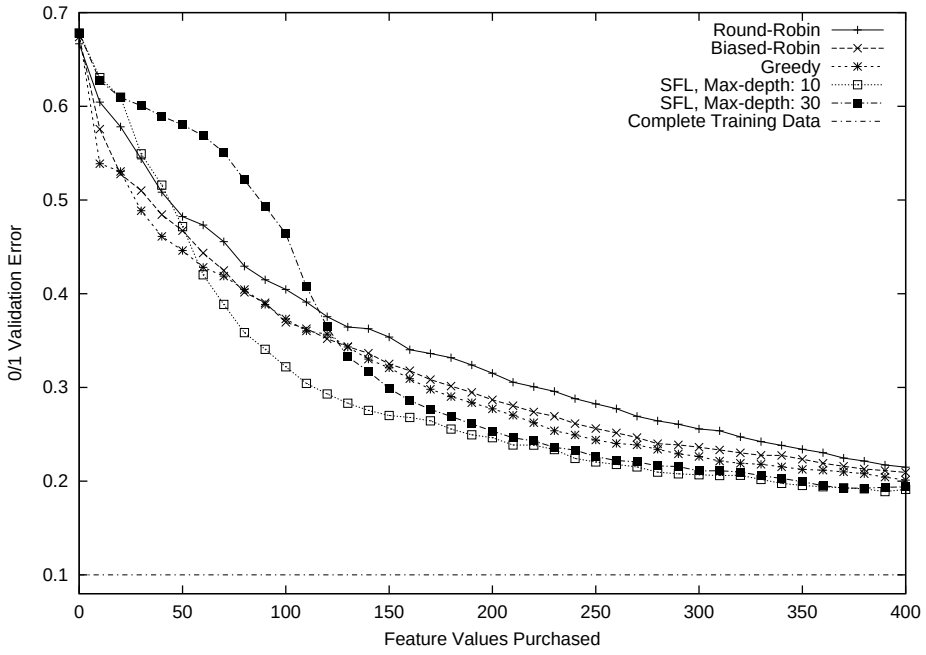


(a) Mushroom Data: Max Depth equal to Budget

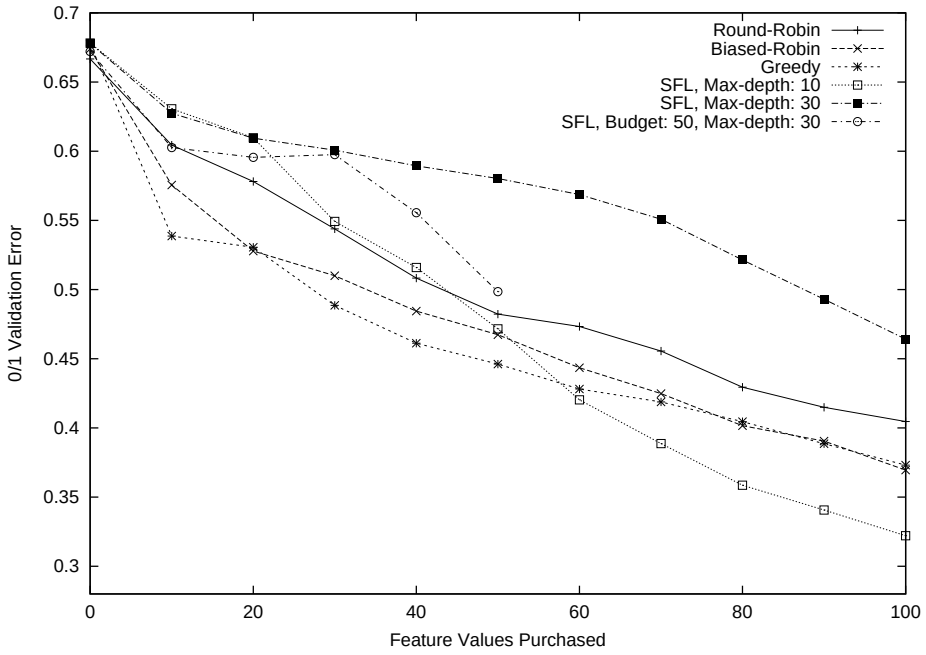


(b) Mushroom Data: Various Max Depths

Figure 3.5: Performance on UCI Mushroom data. 0/1 error error on a validation set consisting of 20% of the data. Errors are averages of 50 trials.

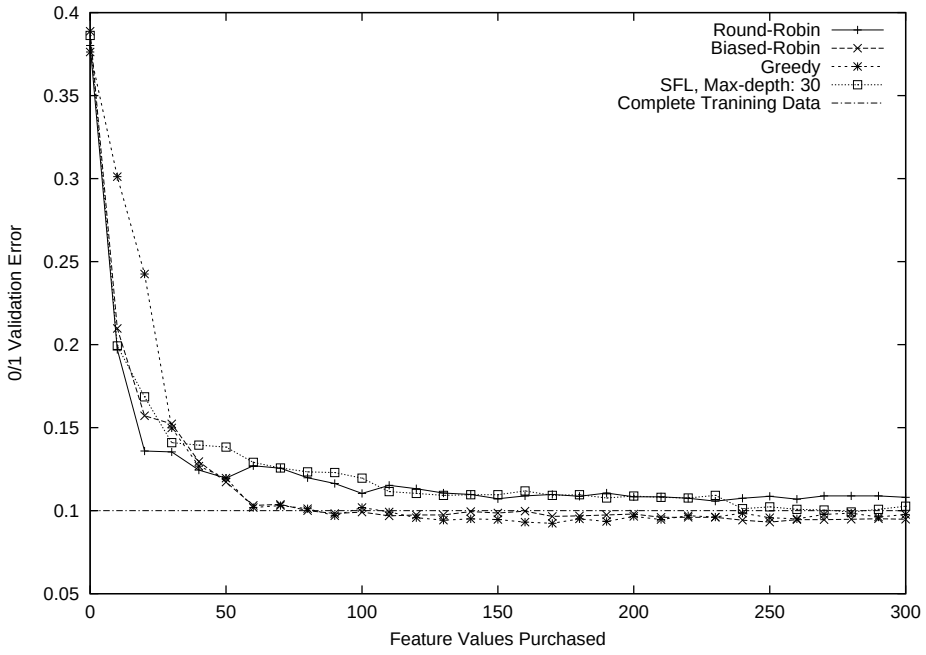


(a) Nursery Data

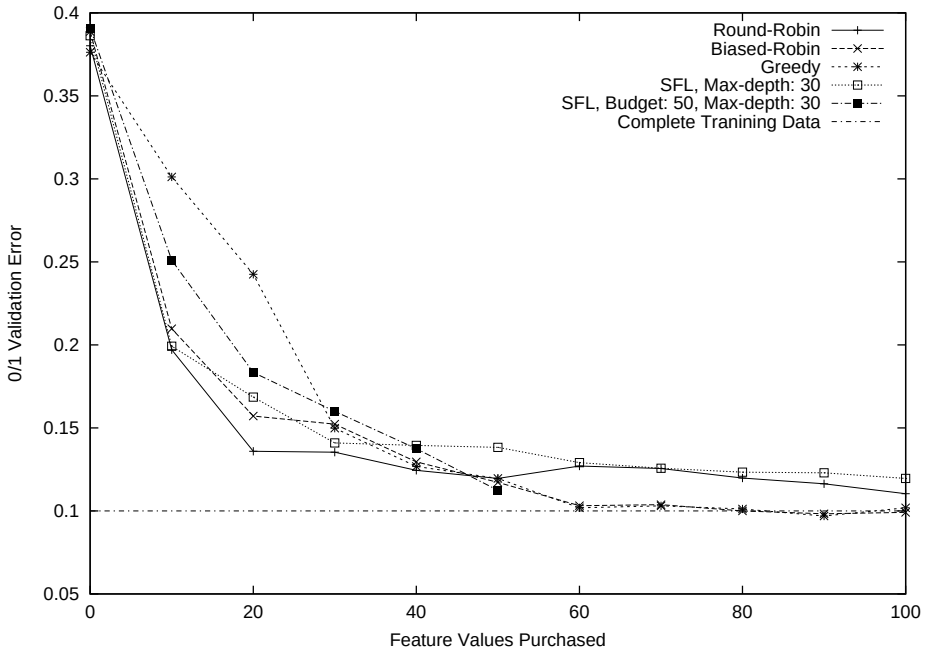


(b) Nursery Data, SFL with $b = 50$

Figure 3.6: Performance on UCI Nursery data. 0/1 error error on a validation set consisting of 20% of the data. Errors are averages of 50 trials.



(a) Votes Data



(b) Votes Data, SFL with $b = 50$

Figure 3.7: Performance on UCI Votes data. 0/1 error error on a validation set consisting of 20% of the data. Errors are averages of 50 trials.

Chapter 4

Related Work

4.1 Bandit Problems

The 'Coins Problem,' discussed in Chapter 2, is a bandit problem [BF85]. 'Bandit' problems are named after slot machines called 'one-armed-bandits.' These slot machines will, for a price, allow a gambler to pull an 'arm,' which results in some immediate reward (possibly zero). The reward distribution is assumed to be independent of previous rewards. (That is, the rewards are iid.) Closest to our work is the 'multi-armed bandit problem,' where there are n different arms we could pull, and our goal is to maximize our expected reward over time. The Coins Problem can be immediately reduced to a multi-armed bandit problem, with the coins mapping to arms, and the loss mapping to (negative) reward. This formulation provides intuition about the behaviour and characteristics of different policies, and details which policies are effective for tackling budgeted learning in the context of multi-armed bandits. The bandit formulation provides a theoretical grounding for budgeted problems in a crisp setting, before it is extended in Chapter 3 to full classifier learning.

The important difference between the Coins Problem and the current state of the art of the multi-armed bandit literature is the reward structure. Nearly all multi-armed bandit work is dependent on a *discounted reward structure*, possibly up to a finite horizon. (*E.g.*, use of the Gittins index solution requires this assumption [BF85].) This discounted structure does not apply to our problem, which offers *zero* reward until the last action when expected loss is computed. This reward structure has not as yet been addressed in the bandit literature[1].

[1] Personal communication with D. Berry of [BF85].

4.2 Hoeffding Races

Hoeffding races [MM97] describe another statistical decision making procedure that is concerned with efficiency in terms of the number of queries made. The application here is model selection, where we wish to choose the best out of a set of classifiers (*i.e.*, k-nn, neural net, etc.) The set of models may be large, and cross-validating may be computationally very expensive, so we want to minimize unnecessary queries (*i.e.*, classifications of test instances) as much as possible. This is accomplished by keeping statistics about the 'goodness' of each model, and using them with Hoeffding/Chernoff bounds [Mit97] to construct confidence intervals about the relative goodness of different models. This allows various strategies to be used when deciding which model to query next. (For example, it is common to query the model with the highest upper bound.) Also, once a particular model is shown to be worse than another model with high probability (*i.e.*, the models' confidence intervals do not overlap) we no longer test the inferior model.

Hoeffding races offer insight into how we might wish to go about budgeted learning, for example, by using statistics to compute bounds or other measures of how good or bad we think different features are. However, while Hoeffding races attempt to be as efficient as possible with queries, they have no concept of a prior *fixed* budget. Also, we will make several distributional assumptions for budgeted learning (as we would for ordinary Bayesian learning) which obviate the need for probability approximations in the from of Chernoff or Hoeffding bounds.

Other learning systems that use techniques similar to Hoeffding races include PALO [Gre96], which provides performance guarantees on hill-climbing, and Leslie Kaelbling's 'interval estimation' (IE) algorithm [Kae93], which constructs approximate confidence intervals to improve reinforcement learning. Again, these systems to not consider having a fixed prior budget.

4.3 Active Learning

Many on-line learners try to minimize the number of training examples, either explicitly or implicitly (*e.g.*, [MCR93], [SG95], or other PAC results that deal with reducing sample complexity). These approaches allow the learner to acquire as many examples as are needed to meet some requirements — *e.g.*, for some statistical test, or some specified ϵ and δ values in the case of PAC-learners [Val84].

We, however, have a firm total budget, specified before the learning begins. Moreover, our approach is fine-grained, as our system can explicitly ask for the value of a *single* specified feature, rather than an entire tuple of values, one for each feature of an instance. (In fact, our results show that this alternative "Round-Robin" approach is often inferior to other policies.)

This kind of problem is also related to active learning scenarios as described in [TK00], [RM01], and [LMRar]. In typical pool-based active learning, we have a pool $\mathcal{P}$ of instances that have all feature values specified, but no class labels. We are considering the complement of the problem: *class labels are available but feature values are not.* The work in [TK00] could be applied to our problem since it is designed for general belief nets of which our Naïve Bayes model is a special case. However, our goal is to build a good *classifier* as opposed to a good generative model.

Also, in previous active learning results (including [TK00]) greedy methods have been shown effective in reducing training sample size, and deeper lookahead has not been used because of inefficiency and insignificant gains (specifically see [LMRar]). However, we observe that in our case the greedy method often has poor performance, and that looking deeper can pay significant dividends.

Budgeted learning is also related to cost-sensitive learning and active classification (*e.g.*, [Ang92, Tur00, GGR02]), although feature costs in [Tur00, GGR02] refer to costs at *classification* time, while we are concerned with total cost during the *learning* phase. Nonetheless, like several active learning results [LMRar, TK00, RM01], we show that *selective querying* can be much more efficient than simplistic methods such as round-robin.

Chapter 5

Conclusions

5.1 Future Work

This work can be extended in several obvious directions. We have chosen to use a Naïve Bayes classifier, but any classifier that can deal with incomplete data tuples could be used, in principle. However, it is critical that we be able to compute the expected loss of a classifier, which in turn means that we must have a means of generating samples of the input distribution, or at least a model of the input distribution that allows expectations to be computed exactly. For the NB case, the classifier is also the generative model of the input distribution, but other non-statistical classifiers (like SVMs, say) would need a separate generative model to compute expected loss, and it is not clear what this generative model should look like, *e.g.*, how it should model dependencies, etc. Another problem when moving to more complicated classifiers that introduce dependencies between features is that the number of decisions when choosing which feature to query next will increase. In the Naïve Bayes case, we needed only to decide the *class label* of the instance we wished to purchase a feature from. For a more general Bayes net where we might want to purchase the value of a node, we would need to decide on the configuration of all parents of the node, and the number of configurations is exponential in the number of parents.

An immediate extension of our work is to handle the detection of dependencies among features and dropping redundant features in order to improve the performance of the learned classifier. This can be viewed as a special case of actively learning structure [TK01]. Dependency detection would remedy a problem on some of the UCI datasets such as votes, when a Naïve Bayes classifier using all features performs worse than using only the single best feature for classification. We are currently investigating logistic regression as a means of mitigating the problems caused by unmodeled dependencies. While better dependency

modeling is desirable, it may be difficult: The introduction of model dependencies among features will increase the space of actions because when purchasing a feature X_i, we will need to decide not only what the class label should be, but what values the other *features* should have in the instance from which we purchase X_i. This increase in the number of actions increases the size of the search space we examine when choosing an action.

The cost structure presented here is quite simple, but real data acquisition can have a very complex cost structure. One could imagine for example, extrapolating from the medical study that was our motivation, a situation with a fixed cost for obtaining a new (empty) data tuple with its corresponding class label, followed by incremental feature value costs. (Imagine it costs \$50 to have a patient come into a clinic, after which each individual test costs \$10.) Concerning the budget, one could consider the scenario where we have a "soft" budget, perhaps with an increasing cost per feature after we have expended our initial b. If there are major differences in costs or if features are expensive, then the goal should be to learn a *cost-sensitive* or *active* classifier [GGR96, Tur00] still in this budgeted framework.

Though it is too computationally expensive to solve optimally, our problem does have some structure that may be exploitable in its MDP form. We are also interested in the suitability various approximate methods for solving MDPs (*i.e.*, [Duf02]) for use on our problem.

5.2 Contributions

We have formulated the general "budgeted learning problem" as a Markov Decision Process and shown that its optimal solution appears to require computation time exponential in the number of features. (In fact it is NP-hard under different feature costs, see [MLG03].) In examining the simpler but closely related 'Coins Problem,' we have seen that uniform strategies are suboptimal, even for seemingly simple cases with uniform and identical priors.

We have also shown in empirical settings that simple policies such as round-robin and greedy loss reduction can be problematic on certain datasets, and propose two alternatives (Biased-Robin and Single Feature Lookahead) that can perform significantly better than these simple policies in certain situations. Empirical performance results both on synthesized data and on parts of the UCI dataset support our claim that the budget-aware policies we have proposed are preferable when faced with a budgeted learning problem.

Bibliography

[Ang92] D. Angluin. Computational learning theory: survey and selected bibliography. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 351–369. ACM Press, New York, NY, 1992.

[BF85] D.A. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments*. Chapman and Hall, New York, NY, 1985.

[BM98] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[Dev95] J. Devore. *Probability and Statistics for Engineering and the Sciences*. Duxbury Press, New York, NY, 1995.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, NY, 2001.

[Duf02] M. Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusets, Amherst, 2002.

[GGR96] R. Greiner, A. Grove, and D. Roth. Learning active classifiers. In *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

[GGR02] R. Greiner, A. Grove, and D. Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2), 2002.

[Gre96] Russell Greiner. PALO: A probabilistic hill-climbing algorithm. *Artificial Intelligence*, 84:1 – 2:177 – 204, 1996.

[Hec95] David Heckerman. *A Tutorial on Learning With Bayesian Networks*. Microsoft Research, Redmond, WA, 1995.

[Hol93] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 3:63 – 91, 1993.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, NY, 2001.

[Kae93] Leslie Pack Kaelbling. *Learning in embedded systems*. MIT Press, Cambridge, Mass., 1993.

[LMG03] Daniel J. Lizotte, Omid Madani, and Russell Greiner. Budgeted learning of naive-bayes classifiers. In *Proceedings of UAI-03*, 2003.

[LMRar] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, To appear.

[MCR93] R. Musick, J. Catlett, and S. Russell. Decision theoretic subsampling for induction on large databases. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 212 – 219, Amherst, MA, 1993.

[Mit97] Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[MLG03] O. Madani, D. Lizotte, and R. Greiner. Foundations budgeted learning: The multi-armed bandit case. submitted, 2003.

[MM97] O. Maron and A. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1-5):193–225, 1997.

[Put94] M. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, New York, NY, 1994.

[RM01] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, San Francisco, CA, 2001.

[RN95] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[SG95] Dale Schuurmans and Russell Greiner. Sequential PAC learning. In *Proceedings of COLT-95*, Stanford University, 1995.

[TK00] S. Tong and D. Koller. Active learning for parameter estimation in bayesian networks. In *NIPS*, pages 647–653, 2000.

[TK01] S. Tong and D. Koller. Active learning for structure in Bayesian networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2001.

[Tur00] P. Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, pages 15–21, 2000.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Web] http://www.cs.ualberta.ca/∼greiner/budget.html.

[Wei] Eric Weisstein. Faulhaber's formula. *Eric Weisstein's World of Mathematics*. http://mathworld.wolfram.com/FaulhabersFormula.html.

Appendix A

Proofs

Lemma 4. *(Page 17.) If $P_{\mathrm{B}}(x;\gamma_1,\gamma_2)$ is a Beta distribution with parameters γ_1 and γ_2, then*

$$P_{\mathrm{B}}(x;\gamma_1,\gamma_2) = \frac{\gamma_1}{\gamma_1+\gamma_2}\cdot P_{\mathrm{B}}(x;\gamma_1+1,\gamma_2) + \frac{\gamma_2}{\gamma_1+\gamma_2}\cdot P_{\mathrm{B}}(x;\gamma_1,\gamma_2+1) \tag{A.1}$$

Proof.

$$\begin{aligned}
P_{\mathrm{B}}(x;\gamma_1,\gamma_2) &= \frac{\gamma_1}{\gamma_1+\gamma_2}\cdot P_{\mathrm{B}}(x;\gamma_1+1,\gamma_2) + \frac{\gamma_2}{\gamma_1+\gamma_2}\cdot P_{\mathrm{B}}(x;\gamma_1,\gamma_2+1)\\
&= \frac{\gamma_1}{\gamma_1+\gamma_2}\cdot\frac{\Gamma(\gamma_1+\gamma_2+1)}{\Gamma(\gamma_1+1)\Gamma(\gamma_2)}x^{\gamma_1}(1-x)^{\gamma_2-1}\\
&\qquad + \frac{\gamma_2}{\gamma_1+\gamma_2}\cdot\frac{\Gamma(\gamma_1+\gamma_2+1)}{\Gamma(\gamma_1)\Gamma(\gamma_2+1)}x^{\gamma_1-1}(1-x)^{\gamma_2}\\
&= \frac{\Gamma(\gamma_1+\gamma_2)}{\Gamma(\gamma_1)\Gamma(\gamma_2)}x^{\gamma_1}(1-x)^{\gamma_2-1} + \frac{\Gamma(\gamma_1+\gamma_2)}{\Gamma(\gamma_1)\Gamma(\gamma_2)}x^{\gamma_1-1}(1-x)^{\gamma_2}\\
&= \frac{\Gamma(\gamma_1+\gamma_2)}{\Gamma(\gamma_1)\Gamma(\gamma_2)}\left[x^{\gamma_1}(1-x)^{\gamma_2-1} + x^{\gamma_1-1}(1-x)^{\gamma_2}\right]\\
&= \frac{\Gamma(\gamma_1+\gamma_2)}{\Gamma(\gamma_1)\Gamma(\gamma_2)}\left[x^{\gamma_1-1}\left(x(1-x)^{\gamma_2-1} + (1-x)^{\gamma_2}\right)\right]\\
&= \frac{\Gamma(\gamma_1+\gamma_2)}{\Gamma(\gamma_1)\Gamma(\gamma_2)}\left[x^{\gamma_1-1}(1-x)^{\gamma_2-1}\left(x+(1-x)\right)\right]\\
&= \frac{\Gamma(\gamma_1+\gamma_2)}{\Gamma(\gamma_1)\Gamma(\gamma_2)}\left[x^{\gamma_1-1}(1-x)^{\gamma_2-1}\right]\\
&= P_{\mathrm{B}}(x;\gamma_1,\gamma_2)
\end{aligned}$$

□

Note that by integrating both sides of Equation A.1, we can derive the same rule for Beta cumulative density functions, which is used in proving Theorem 1.

Theorem 1. *(Page 17.) For any policy π,*

$$\sum_{\mathbf{s}\in\mathcal{O}(\pi,\mathbf{s}_0)} P(\mathbf{s})\cdot\langle\Theta_{\max}|\mathbf{s}\rangle = \langle\Theta_{\max}|\mathbf{s}_0\rangle$$

Proof. First, we will show that

$$\sum_{\mathbf{s}\in\mathcal{O}(\mathrm{a},\mathbf{s}_0)} P(\mathbf{s})\cdot\langle\Theta_{\max}|\mathbf{s}\rangle = \langle\Theta_{\max}|\mathbf{s}_0\rangle$$

I.e., for any action a_k, the expected value of $\Theta_{\max}$ summed over all outcome states of a_k from $\mathbf{s}$ is the same as the expected value of $\Theta_{\max}$ evaluated in state $\mathbf{s}$. Using Lemmas 4 and 3 and Equation A.1, it is easy to show that

$$\begin{aligned}
\langle \Theta_{\max}|\mathbf{s}\rangle &= 1 - \int_0^1 \prod_i \mathrm{cdf}_i(x)\,\mathrm{d}x \\
&= 1 - \int_0^1 \prod_{i\neq k} \mathrm{cdf}_i(x) \cdot \left[\hat{\theta}_k \mathrm{cdf}_k(x|\gamma_{k1}\texttt{++}) + (1-\hat{\theta}_k)\mathrm{cdf}_k(x|\gamma_{k2}\texttt{++})\right] \mathrm{d}x \\
&= 1 - \hat{\theta}_k \int_0^1 \prod_{i\neq k} \mathrm{cdf}_i(x) \cdot \mathrm{cdf}_k(x|\gamma_{k1}\texttt{++})\,\mathrm{d}x \\
&\qquad - (1-\hat{\theta}_k) \int_0^1 \prod_{i\neq k} \mathrm{cdf}_i(x) \cdot \mathrm{cdf}_k(x|\gamma_{k2}\texttt{++})\,\mathrm{d}x \\
&= \hat{\theta}_k + (1-\hat{\theta}_k) - \hat{\theta}_k \int_0^1 \prod_{i\neq k} \mathrm{cdf}_i(x) \cdot \mathrm{cdf}_k(x|\gamma_{k1}\texttt{++})\,\mathrm{d}x \\
&\qquad - (1-\hat{\theta}_k) \int_0^1 \prod_{i\neq k} \mathrm{cdf}_i(x) \cdot \mathrm{cdf}_k(x|\gamma_{k2}\texttt{++})\,\mathrm{d}x \\
&= \hat{\theta}_k \left[1 - \int_0^1 \prod_{i\neq k} \mathrm{cdf}_i(x) \cdot \mathrm{cdf}_k(x|\gamma_{k1}\texttt{++})\,\mathrm{d}x\right] \\
&\qquad + (1-\hat{\theta}_k) \left[1 - \int_0^1 \prod_{i\neq k} \mathrm{cdf}_i(x) \cdot \mathrm{cdf}_k(x|\gamma_{k2}\texttt{++})\,\mathrm{d}x\right] \\
&= \hat{\theta}_k \cdot \langle \Theta_{\max}|(s|\gamma_{k1}\texttt{++})\rangle + (1-\hat{\theta}_k) \cdot \langle \Theta_{\max}|(s|\gamma_{k2}\texttt{++})\rangle \qquad \text{(A.2)}
\end{aligned}$$

which means that $\langle \Theta_{\max}|\mathbf{s}\rangle$ does not change when averaged over the possible outcomes of any action. To see how this extends to a policy, we refer to Figure A.1, which shows a policy tree. Nodes in the tree represent belief states, with the leftmost in the figure being $\mathbf{s}_0$. (Nodes are labeled with the action taken next by the policy at that point.) From a given state $\mathbf{s}$, following an upward arc (*i.e.*, observing a 'heads') takes us to state $\mathbf{s}|\gamma_{k1}\texttt{++}$, and following a downward arc (*i.e.*, observing a tails) takes us to state $\mathbf{s}|\gamma_{k2}\texttt{++}$. Using Equation A.2 starting from the right side of the tree (leaf nodes,) we can 'collapse' the value of $\langle \Theta^{(s)}_{\max}\rangle$ into earlier states, all the way back to $\mathbf{s}_0$. (*N.b.*, though Figure A.1 shows an optimal policy tree, the same induction applies to general policy trees.)

□

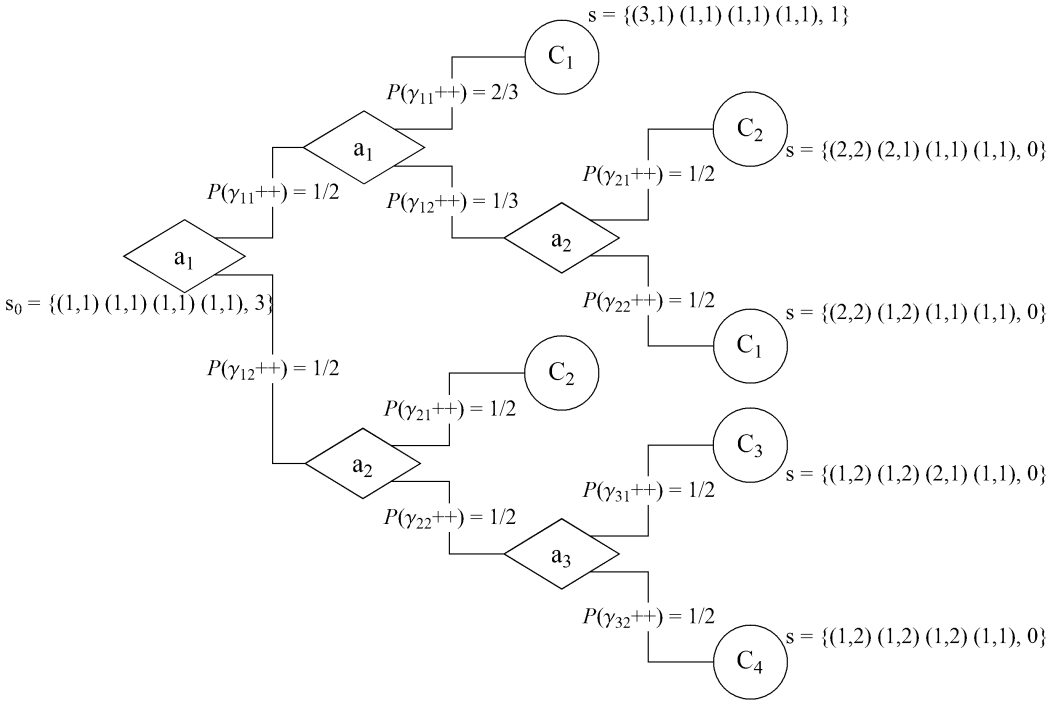


Figure A.1: *(Repeated from page 10.)* An optimal policy tree for budget $b = 3$ on identical uniform priors, where $n = 4$. Diamond boxes are actions, with 'head' outcomes leading up, and 'tail' outcomes leading down. Transition probabilities are indicated. Some branches terminate early as the coin to report (circled) is already determined.