University of Alberta

Library Release Form

Name of Author: Varun Grover

Title of Thesis: Active Learning and its Application to Heteroscedastic Problems

Degree: Master of Science

Year this Degree Granted: 2009

Varun Grover
4071 - 33A Street
Edmonton, Alberta
Canada, T6T-1R4

Date: _____

University of Alberta

Active Learning and its Application to Heteroscedastic Problems

by

Varun Grover

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
Spring 2009

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 **Active Learning and its Application to Heteroscedastic Problems** submitted by Varun Grover in partial fulfillment of the requirements for the degree of **Master of Science**.

Csaba Szepesvári

Dale Schuurmans

Edit Gombay

Date: _____

To my family

Abstract

This thesis presents Active Learning algorithms for heterogeneous distributions. Active Learning is a vast and growing sub-field of Machine Learning, where many significant contributions have been made. This thesis makes two contribution to the Active Learning field. First contribution is a broad survey of Active Learning literature and second contribution is two new Active Learning algorithms for the multi-armed bandit and stratified sampling problems. Both of these algorithms learn about the distributions of the underlying problem based on the observed samples. Then using the knowledge gained, the algorithms allocate samples to appropriate areas of the problem domain. Our algorithms suffers an excess loss compared to an oracle with full knowledge of the problems that scales at a rate of $n^{-3/2}$. Empirical results confirm this rate and demonstrate that the active algorithm perform exceptionally well compared to a passive algorithm.

Acknowledgements

I would like to deeply thank my supervisor Csaba Szepesvári for his help and support throughout this program. I would also like to thank my lab mates for their help and guidance. A special thanks to Barnabás Póczos and András Antos. Also I want to thank my family, without their support this would have been impossible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine Learning is the research of developing smart algorithm that can learn unknown patterns from the available data. Since the resources are always limited, one necessary quality of a smart algorithm is that it knows where to spend its efforts. This thesis explores a sub-field of Machine learning called Active Learning which focuses on building algorithms that minimize the number of samples required to achieve a desired level of accuracy. This thesis makes two major contributions. First is a fairly broad survey of the Active Learning literature. A variety of algorithms from historic to most current algorithms are examined for the regression, classification and bandit problems.

The second contribution of this thesis is a novel Active Learning algorithm for each of the two problems: multi-armed bandits and stratified sampling. Both of these problems have a number of practical applications in the field of quality control, clinical trials, sampling surveys etc. The excess loss suffered by this algorithm as compared with an oracle with knowledge, apart for some logarithmic factors, scales as $n^{-3/2}$. The performance of the algorithm is illustrated on three real-world based problems.

1.1 Active Learning

An important Machine Learning technique is Supervised Learning. A Learner, under Supervised Learning setting, learns a predictor or a model by observing value for samples provided by the Environment (also called supervisor/teacher if the response comes from a human user). The training data consists of input samples and their corresponding values. A common approach to learn a predictor, under supervised learning setting, is to requests the value for a sample drawn at random from a pre-determined distribution. The Learner's choice of a sample is not influenced by the previously observed samples. This approach is referred to as Passive Learning.

In Active Learning, on the other hand, the Learner utilizes the information gained from previous observation to choose which sample to observe next. Under the Active Learning

setting, the Learner is provided with either a large pool of samples (without their corresponding value) or the Learner has the flexibility to sample any point form the input domain. The Learner must then decide which samples it should request a value for from the Environment. The Learner makes this decision based on the values for samples it has seen so far and the samples (without their corresponding values) available to it. The goal of the Learner is to minimize the number of observed samples required to achieve a certain level of accuracy.

By requesting values for selected samples, an Active Learning algorithm can achieve huge performance gains. For example consider the problem of learning a function that maps the $[0, 1]$ interval to the reals and that takes the value 0 everywhere to the left of some point, while on the other side it takes the value of 1. More precisely, the input samples are $x \in [0, 1]$ and the value (which will be called a "label") returned by the Environment is $y = 0$ when $x < a$ and is $y = 1$ when $x \geq a$, where $y \in \{0, 1\}$. The goal of the Learner is to minimize the number of labeled samples required to learn this function with error rate at most ε. Now consider a Passive Learning approach and assume that the samples are chosen uniformly at random over the domain of the input samples. The number of samples required by a passive learner will be $\Theta(1/\varepsilon)$. An Active Learner, on the other hand, uses a binary search to identify the boundary where the value of the function changes from 0 to 1. This approach requires $O(\log 1/\varepsilon)$ samples, which is significantly less than the number of samples required by a Passive Algorithm. This exponential decrease in the *sample complexity*, number of samples required to achieve desired accuracy level, is the motivation behind the Active Learning research.

An Active Learning algorithm requires either access to a pool of input samples or the flexibility to query the Environment, while a Passive Learning algorithm does not. Either of these requirements is not limiting. Often, obtaining the labels for an input sample has some associated cost, while the input samples by themselves are readily available. For example consider the problem of classifying documents on the world-wide-web. The number of documents available (without their label) is enormous, while obtaining a label for a document requires some effort. Therefore, the requirement of a pool of input samples is generally not a limitation. Similarly, asking the value for a generated sample is not a problem for the Environment as long as the sample is within the domain of input samples.

1.2 Current Active Learning Algorithms

Regression and classification are the two major problem domains of Supervised Learning. Regression problems were first considered by statisticians. Hence early Active Learning algorithms were developed by statisticians. These approaches are still quite popular and used in the current work. Statisticians referred to Active Learning as Optimal Experiment

Design (OED). We review popular OED optimality conditions such as D-optimality, A-optimality and E-optimality for the linear regression problem.

The Active Learning community have spent a considerable effort on the classification problem. There are many approaches present in the literature. Some algorithms request labels for samples for which they are most uncertain of, for example Lewis and Gale's uncertainty reduction algorithm (Lewis and Gale, 1994). Other algorithms keep a list of valid predictors and use their confidence to decide the next sample to be labeled. The works of Cohn et al. (1994), Seung et al. (1992) and Dasgupta et al. (2007) are examples of this approach. Yet others choose samples that reduce the expected error (Roy and McCallum, 2001). Recent work by Dasgupta et al. (2007) presented an algorithm whose sample complexity is always within a constant factor of the sample complexity of passive algorithms and in many cases may perform better than passive algorithms. Their work also highlight the conditions under which an Active Learning algorithm has a strict advantage over the Passive Learning algorithm.

We argue that Active Learning can essentially be modeled under the Reinforcement Learning(RL) framework. The essence of RL algorithms (and hence the Active Learning algorithms) is balancing the tradeoff between exploration and exploitation. We present a work by Auer et al. (2002) that solves this problem for a specific setting of finite-armed bandit problem.

1.3 Active Learning under Heteroscedasticity

This thesis presents a novel algorithm for each of the two following problems: multi-armed bandits and stratified sampling. In the multi-armed bandits problem we have a set of bandit arms, each with their own distribution. In the problem studied here the goal is to estimate the mean of each bandit arm with equal precision by allocating samples to each arm. The Learner samples each arm to learn about the underlying distribution. Using the knowledge gained, the Learner samples those arms more frequently for which it is unsure of their mean performance, i.e., those arms that have high variance. This problem is an abstraction of quality control and clinical trials problems where inspections are performed on different groups (machines, parts, target population) to measure the mean response. Every sample has a cost (cost of inspection, destruction of part), and thus in order to reduce cost the algorithm should sample those groups for which it has the least knowledge about, more frequently.

In the second problem (stratified sampling) we deal with estimating the expected value of a function of a random variable. The Learner estimates the mean of each stratum and then combines them to obtain the mean of the entire function. Stratified sampling is used in many problems such as reliability estimation or surveys. Similar to the multi-armed bandits

problem the Learner is faced with the problem of deciding the number of samples from each stratum. We employ a similar solution to the first problem: the Learner samples each stratum and learns its distribution. Using this knowledge, the Learner then allocates more samples to the stratum with higher variance.

The algorithm incrementally estimates the variance of the underlying distributions and allocate samples accordingly. At each time step, the algorithm estimates the respective variances of the underlying distributions from the previous values seen from them. The algorithm then greedily samples the arm (or stratum) with the most expected loss. However, if there is an arm, which is severally under-sampled then the algorithm samples that arm, instead, in order to avoid errors caused due to the estimation error. This algorithm asymptotically achieves the same loss as the loss suffered by an optimal allocation rule that has full knowledge of the variance of each arm. Furthermore, the excess loss suffered by our algorithm converges to zero at a rate of $\tilde{O}(n^{-3/2})$.[1]

The performance of the algorithm is illustrated on three real-world based problems. The first problem is an abstract problem based on clinical trials and quality assurance. The second problem deals with estimating the value of the Canadian CPI (Consumer Price Index) for the year 2008. The problem is modeled based on survey data available through Statistics Canada's online reports. The final problem deals with pricing an Asian option in the Black-Scholes setting. The results for all three problems show a strict improvement over a passive approach to these problems.

1.4 Roadmap for the Rest of the Thesis

This thesis starts off by presenting a wide range of Active Learning algorithms and their properties (Chapter 2). In Chapter 3 we introduce the multi-armed bandits problem and present an incremental Active Learning algorithm to solve this problem effectively. Chapter 4 then presents the stratified sampling problem along with another incremental algorithm as an active solution for this problem. Experimental results are presented in Chapter 5, where the performance of the discussed algorithm is illustrated on three real-world based problems. Final remarks are provided in Chapter 6.

[1] A nonnegative sequence (a_n) is said to be $\tilde{O}(f(n))$, where $f : \mathbb{N} \to \mathbb{R}^+$, $\lim_{n\to\infty} f(n) = \infty$, if $a_n \leq Cf(n)\log(f(n))$ with a suitable constant $C > 0$.

Chapter 2

Related Work

Active Learning is referred by many names in the literature. Most people refer to it as Active Learning (Cohn et al., 1996) or Selective Sampling (Freund et al., 1997). Active Learning originated from the statistics literature where researchers were looking at designing experiments to gather data in an optimal way. Their task is to design an experiment such that your objective (modeling or prediction error) is reduced. The statisticians referred to this area of research as "Optimal Experiment Design" (Emery and Nenarokomov, 1998). Later it became popular in Machine Learning as a framework for reducing the sample complexity, the number of samples required by an algorithm to learn a model or hypothesis, of the learning algorithm. Many researchers have developed algorithms that use the Active Learning framework and have been shown to reduce sample complexity by a substantial factor. In this chapter we will review some of these algorithms and their properties.

2.1 Active Learning Framework

A learning algorithm is considered an Active Learning algorithm if it has the ability and intelligence to choose which samples to request a value for. In Passive Learning the training examples come from a source (i.e., a fixed distribution) that is not controlled by the learner. In the Active Learning framework, on the other hand, the learning algorithm chooses the next sample based on the past samples. This difference can lead to an improved performance of a learner for a given task.

Many learning situations can in general be described as the problem of finding a model (or hypothesis) given some data from an unknown data generating source ("Environment") that, in some sense, matches the source the best. The success of a Learner is measured by the quality of the model produced by the Learner as a function of the number of values (in regression setting) or labels (in classification setting) requested by the Learner. The quality of the model can be measured by either the prediction error (how well the model is doing in predicting the values or labels) or the related model identification error (how close

Query-based Active Learning
repeat
L computes $x \in X$
L requests value for sample x from E
E provides a value y
L updates its model with (x, y)
end

Pool-based Active Learning
L receives a pool of samples $\{x_1, \ldots, x_p\} \subset X$
repeat
L selects $i \in \{1, \ldots, p\}$
L requests value for sample x_i
E provides value y
L updates its model with (x_i, y)
end

Table 2.1: Protocols of Active Learning. The Learner L has control over which sample to request a value for from the Environment E. The two protocols differ in whether L can request the value for any sample or just for one of the samples in a finite set, the pool.

the estimated model is to the "true" model), the latter measure being used when one is interested in "understanding" how the data is generated. By allowing the Learner to choose the samples, the hope is that better quality models can be produced with fewer requests for values or labels.

All Active Learning algorithms follow a the general framework described in Table 2.1. Active learning algorithms can differ in a few characteristics:

(i) The Learner could be restricted to samples from a fixed set (this is called pool-based learning) or could ask for a sample with certain properties (query-based learning), cf. Table 2.1; *(ii)* Further, the training examples could be requested in batches or one-by-one (sequentially).

In this thesis we will not consider the query-based Active Learning algorithm and only focus on pool-based Active Learning algorithms. Under the pool-based setting there are a variety of algorithms for selecting a sample to be evaluated by the Environment. Some algorithms focus on samples which help reduce the uncertainty/entropy of their estimate of the model, while others select samples that directly minimize the expected error of their prediction. Yet others select samples that cause most conflict among the current set of valid hypotheses. We review some of the common Active Learning algorithms that have been tried on a wide variety of tasks in the next section.

2.2 Active Learning for Regression Problems

Regression is the problem of finding a model that can be used to predict some numerical-valued response variable given some inputs. The response variable is called the dependent variable, while the inputs are assumed to be numerical-valued vectors whose components are called the independent variables. Regression problems are an essential part of Machine Learning. We concentrate on the most common regression method - linear regression in this section. We, first, formalize the linear regression problem in the next subsection and then

look at how this problem can be solved under the Active Learning framework.

2.2.1 Linear Regression

Given an input vector $x \in \mathbb{R}^d$, the Environment computes its response by first drawing a random variable $\epsilon \sim N(0, \sigma^2)$ and then returning

$$Y = \theta^T x + \epsilon,$$

where $\theta \in \mathbb{R}^d$ is a parameter vector, known to the Environment, but unknown to the Learner. The goal of the Learner is to approximate this parameter vector θ. As we discussed, the Learner can either focus on reducing the model identification error or the prediction error. The model identification error is often measured as the distance between the "true" parameter θ and the parameter $\hat{\theta}$ picked by the Learner. The actual distance is typically the Euclidean distance or a weighted Euclidean distance. The prediction error is generally measured by the mean-square error. When it is measured at a point $x \in \mathbb{R}^d$ of the input space, for $\hat{\theta}$ we get

$$L(\hat{\theta}, x) = \mathbb{E}\left[(Y - \hat{\theta}^T x)^2 | \hat{\theta}\right] = ((\hat{\theta} - \theta)^T x)^2 + \sigma^2 = x^T(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T x + \sigma^2.$$

The Learner has access to samples of the form $\{(x_1, Y_1), \ldots, (x_n, Y_n)\}$ from which the Leaner must learn the parameter vector θ for reducing either of the above mentioned errors. The standard method to learn a good parameter vector $\hat{\theta}$ given these samples is to compute the least-squares estimate of θ:

$$\hat{\theta} = (X^T X)^{-1} X^T Y,$$

where $X^T = (x_1, \ldots, x_n) \in \mathbb{R}^{d \times n}$, $Y = (Y_1, \ldots, Y_n)^T \in \mathbb{R}^{n \times 1}$, and the $d \times d$ information matrix $X^T X$ is assumed to be of full rank. (Note that here we abuse notation in that Y is used to denote a vector here, while above it was used to denote a scalar. We hope that the meaning of Y will be clear from the context.) The estimate $\hat{\theta}$ is both the maximum-likelihood estimate and the estimate that minimizes the total squared error on the training data. Further, by the Gauss-Markov theorem this is the minimum variance unbiased estimator. Note that $\hat{\theta}$ is random, because the column-vector Y is random. Further, since we assumed that $(\epsilon_1, \ldots, \epsilon_n) \sim N(0, \sigma^2 I)$, the distribution of $\hat{\theta}$ is also normal. In particular,

$$\hat{\theta} \sim N(\theta, \sigma^2 M^{-1}), \tag{2.1}$$

thus

$$\mathbb{E}[\,\hat{\theta}\,] = \theta$$

and

$$\mathrm{Cov}[\,\hat{\theta}\,] = \mathbb{E}\left[(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T\right] = \sigma^2 M^{-1}.$$

Here $M = (X^T X)$ is the same matrix that is used to compute $\hat{\theta}$. Whenever we want to emphasize the dependence of M on X, we will use $M(X)$ to denote M. Similarly, we will use $\hat{\theta}(X)$ to emphasize the dependence of $\hat{\theta}$ on X.

Note that the distribution of $\hat{\theta}$ fully determines any error measure of interest, let it be a measure of the model estimation or the prediction error. Further, in this case the distribution of the error $\hat{\theta} - \theta$ is independent of the values or the unknown parameter θ, but depends only on the choice of the inputs $x_1, \ldots, x_n$. This means that one can optimize any error measure of interest that depends only on $\hat{\theta} - \theta$ before seeing any training examples. The various methods of Optimal Experiment Design, detailed in the next subsection, do this and differ only in the error measure to be optimized.

2.2.2 Optimal Experiment Design

Optimal Experiment Design (OED) methods choose an error measure of interest and aim to find the placement of $X^T = (x_1, \ldots, x_n)$ so as to minimize this error measure. When $x_1, \ldots, x_n$ can take any value in the domain and the domain is not further restricted these optimization problems might be difficult to solve (the optimization problems will be specified later). A special case of this problem is when $x_1, \ldots, x_n$ must be selected from a fixed, finite pool of examples (i.e., the domain is restricted to the examples in the pool). If $P = \{z_1, \ldots, z_m\}$ is this pool ($z_i \in \mathbb{R}^d$) then the design problem becomes to choose the multiplicity n_i of each z_i element of the pool such that $\sum_{i=1}^m n_i = n$, $n_i \in \mathbb{N}$. Equivalently, the problem can be described as choosing the allocation ratios $0 \leq \lambda_i \leq 1$ such that $\sum_{i=1}^m \lambda_i = 1$ and $n\lambda_i \in \mathbb{N}$. By dropping the constraint that restricts $n\lambda_i$ to natural numbers one can obtain tractable approximations to the original optimization problems. This is the formulation that will be used below to describe the optimization problems associated with the various optimality criterion.

D-Optimality and G-optimality

Consider the random ellipsoid $\mathcal{C}_q = \{x : (x-\hat{\theta})^T M^{-1}(x-\hat{\theta}) \leq q\}$, where $q > 0$ (the ellipsoid is random as it depends on $\hat{\theta}$ which is random). In fact, for any $0 < \alpha < 1$, by tuning q, we can achieve $\mathbb{P}(\theta \in \mathcal{C}_q) = \alpha$. Thus, the sets $\mathcal{C}_q$ are confidence regions for θ for some fixed level α.

Imagine that one is interested in achieving the best concentration of $\hat{\theta}$ in the vicinity of θ. An error measure that captures this desideratum is to minimize the volume of any of the confidence regions $\mathcal{C}_q$. Since the volume of any of these ellipsoids is proportional to $(\det M^{-1})^{1/2}$, this criterion is equivalent to minimizing the determinant of $M^{-1} = M^{-1}(X)$. The solution of the corresponding optimization problem is called a D-optimal design ('D' because determinant is optimized).

The celebrated Generalized Equivalence Theorem of Kiefer and Wolfowitz (1960) states that a D-optimal design is also the one that minimizes

$$L_\infty(X) \stackrel{\text{def}}{=} \sup_{x \in A} \mathbb{E}\left[L(\hat{\theta}(X), x)\right],$$

i.e., the worst expected squared prediction loss of the least-squares estimator over a region $A \subset \mathbb{R}^d$, and vice versa, provided that the span of the vectors in A is $\mathbb{R}^d$. The points X that minimize $L_\infty(X)$ are said to be G-optimal ('G'-optimality stands for global optimality).[1]

When we have a finite pool $\{z_1, \ldots, z_p\}$, the problem of optimizing the allocation ratios (λ_i) can be written as the convex optimization problem

$$\begin{aligned} \text{minimize} \quad & \log\det \quad (\sum_{j=1}^{p} \lambda_j z_j z_j^T)^{-1} \\ \text{subject to} \quad & \lambda_j \geq 0, \quad \sum_{j=1}^{p} \lambda_j = 1. \end{aligned} \tag{2.2}$$

Note that the equivalence of G- and D-optimality relies crucially on the assumption that the variance of a response does not depend on the input, i.e., that $\sigma^2 \equiv \text{const}$, or in other words that the model is *homoscedastic* as opposed to being *heteroscedastic*.

E-Optimality

E-optimal design minimizes the diameter of the confidence ellipsoids formed by the underlying M matrix ('E' stands for ellipsoid). Equivalently, an E-optimal design minimizes

$$\max_{\|a\|_2^2=1} \mathbb{E}\left[(a^T(\hat{\theta} - \theta))^2\right] = \max_{\|a\|_2^2=1} a^T M^{-1} a,$$

i.e., the worst projected squared error of estimating θ with $\hat{\theta}$. Since $\psi_{\max}(M^{-1}) = \|M^{-1}\|_2$, where ψ are the eigenvalues of a matrix, this is equivalent to minimizing $\|M^{-1}\|_2$, the spectral norm of M^{-1}.

In terms of the allocation ratios, for a finite pool, the solutions of the following convex optimization problem gives approximate E-optimal designs:

$$\begin{aligned} \text{minimize} \quad & \|(\textstyle\sum_{j=1}^{p} \lambda_j z_j z_j^T)^{-1}\|_2 \\ \text{subject to} \quad & \lambda_j \geq 0, \qquad \sum_{j=1}^{p} \lambda_j = 1. \end{aligned} \tag{2.3}$$

[1] More precisely, this result holds only when the design problems are defined for distributions over the input space. Clearly, choosing the points $x_1, \ldots, x_n$ corresponds to a distribution that places masses of weight $1/n$ at the points $x_1, \ldots, x_n$. More generally, a distribution μ gives rise to the dispersion matrix $D(\mu) = \int x x^T \mu(dx)$ and thus $E(\mu) = D(\mu)^{-1}$. Measure-valued design are called approximate as in general they may not be realizable exactly in practice. An approach between the two extremes of working with point sets and measures is to fix a finite number of points $u_1, \ldots, u_p$ in the input space and define the optimization problems in terms of non-negative weights assigned to them that are restricted to sum to one (if all the independent variables are categorial, the input space is finite and this set of vectors can be chosen to be the set of all possible inputs). This leads to optimization problems that can typically be solved numerically in a computationally efficient manner.

A-Optimality

An A-optimal design minimizes the expected (or average) deviation of $\hat{\theta}$ from θ measured by the squared 2-norm, i.e.,

$$\mathbb{E}\left[\|\hat{\theta}-\theta\|_2^2\right].$$

Since $\mathbb{E}\left[\|\hat{\theta}-\theta\|_2^2\right] = \sum_{i=1}^{d}\mathbb{E}\left[(\hat{\theta}_i-\theta_i)^2\right] = \sigma^2\mathrm{tr}(M^{-1})$, an A-optimal design minimizes the trace of the matrix M^{-1}.

In the case of a finite pool, the optimization problem becomes the following convex problem:

$$\begin{aligned} &\text{minimize} && \mathbf{tr}(\textstyle\sum_{j=1}^{p}\lambda_j z_j z_j^T)^{-1} \\ &\text{subject to} && \lambda_j \geq 0, \qquad \sum_{j=1}^{p}\lambda_j = 1. \end{aligned} \tag{2.4}$$

2.3 Active Learning for Classification

Now we move on to the classification problem and present Active Learning algorithms for binary classification problems.

2.3.1 Classification Problems

Let $\Omega \subset \mathbb{R}^d$ be the domain of a classification problem. If the Learner asks for the label of a point $x \in \Omega$, the response of the Environment will be random. In particular, we will assume that there exists a function $\eta : \Omega \to [0,1]$ and if Y_x denotes the response of the Environment for the sample x then $Y_x \in \{-1,1\}$ and

$$\mathbb{P}(Y_x = 1) = \eta(x).$$

In other words, for any point x, the posterior function η specifies the probability that the Environment responds with 1.

A binary classifier f maps the domain of the classification problem to the possible labels: $f : \Omega \to \{-1,1\}$. The prediction error of f is typically measured by the misclassification loss:

$$L_{0/1}(x) = \mathbb{P}(Y_x \neq f(x))$$

Note that

$$\begin{aligned} L_{0/1}(f;x) &= \mathbb{E}\left[\mathbb{I}_{\{Y_x \neq f(x)\}}\right] \\ &= \{1-\mathbb{E}[Y_x f(x)]\}/2 \\ &= \{1-f(x)\mathbb{E}[Y_x]\}/2 \\ &= \{1-f(x)(2\eta(x)-1)\}/2 \\ &= f(x)(1/2-\eta(x))+1/2. \end{aligned}$$

The function f^* that minimizes this loss at every point $x \in \Omega$ is called the *Bayes classifier*.[2] From the above formulation, it is clear that $f^*(x)$ should be 1 if and only if $1/2 - \eta(x) < 0$, i.e., iff $1/2 < \eta(x)$ (when $\eta(x) = 1/2$, both choices are equally good). Further, $L_{0/1}(f^*; x) = \min(1 - \eta(x), \eta(x))$.

One can also measure the error with the squared loss:

$$L^2(f; x) = \mathbb{E}\left[(Y_x - f(x))^2\right].$$

Interestingly, since $\mathbb{E}\left[(Y_x - f(x))^2\right] = 2(1 - f(x)\mathbb{E}\left[Y_x\right])$, we get that $L^2(f; x) = 4L_{0/1}(f; x)$ holds for any $x \in \Omega$ and classifier f. Thus, the two loss functions can be used in an interchangeable manner.

Some algorithms work with probabilistic classifiers. Such a classifier can be described by a function $\pi : \Omega \to [0, 1]$. The interpretation is that given some x and π, the predictor using π chooses label $+1$ with probability $p(x)$ and chooses the label -1 with probability $1 - p(x)$. The natural extension of the 0/1 loss to such probabilistic classifiers is $L_{0/1}(\pi; x) = \mathbb{P}\left(Y_x \neq Z_x\right)$, where Z_x is a $\{-1, +1\}$-valued random variable such that $\mathbb{P}\left(Z_x = +1\right) = p(x)$. Still, $L_{0/1}(p; x) = \{1 - \mathbb{E}\left[Y_x Z_x\right]\}/2$. The extension of the squared loss is $L^2(\pi; x) = \mathbb{E}\left[(Y_x - Z_x)^2\right]$. Since $L^2(\pi; x) = 2 - 2\mathbb{E}\left[Y_x Z_x\right]$, we still get that $L^2(\pi; x) = 4L_{0/1}(\pi; x)$.

A special case of this framework is when the output labels are a deterministic function of the inputs, i.e., when $\eta(x) \in \{0, 1\}$. Another way of saying this is that there exists a function $h^* : \Omega \to \{-1, 1\}$ such that $Y_x = h^*(x)$ holds for any $x \in \Omega$. This is called the "noise free setting". Another assumption that is common is that the learner has access to a set of classifiers, $\mathcal{F}$, that includes the Bayes classifier f^*. When this is not assumed, the Learner is said to be *agnostic*. These are strong assumptions. Note that the two assumptions are independent (i.e., four cases are possible). Early work in Active Learning for classification has mostly focused on the noise free setting and the non-agnostic case. More recent work eliminates these assumptions.

There are four main approaches that have been applied to the classification task using the Active Learning framework:

1. Uncertainty reduction
2. Version Space reduction
3. Expected Error reduction
4. Logistic Regression with A-optimality

[2]The word "Bayes" is used for historic reasons. In fact, "Bayes" does not imply that there was any prior that we took the posterior with respect to. There is no Bayesian computation done here.

Still, the algorithms that we consider are pool based.

The efficiency of an Active Learning algorithm can be measured by the so-called *labeled sample complexity*. Fix $\varepsilon > 0$, a class of classification problems $\mathcal{C}$ and a pool $P \subset \Omega$. The labeled sample complexity of an Active Learning algorithm $\mathcal{A}$ with respect to $(\varepsilon, \mathcal{C}, P)$ is the number of examples in P for which $\mathcal{A}$ requests a label for before the loss of the classifier (measured e.g. using the 0/1-loss) produced by $\mathcal{A}$ drops below ε. The probabilistic counterpart of this definition requires that the algorithm outputs a classifier such that with "high probability" the loss of the classifier drops below ε.

2.3.2 Uncertainty Reduction

Lewis and Gale (1994) introduce the idea of uncertainty reduction for text classification task. Uncertainty reduction is also sometimes referred to as entropy reduction. The algorithm is very intuitive and simple to follow. The algorithm uses a probabilistic classifier. The central idea of the algorithm is to choose the sample with most uncertainty and request labeling for it. Thus, for the binary classifier the sample for which this probability equals 0.5 for class **-1** and similarly 0.5 probability for class **+1** is when our classifier is most unsure. Lewis and Gale (1994) applied this approach to the task of text classification. They empirically showed that the classifier trained with uncertainty reduction objective learned much faster than a classifier with random sampling. However, further work showed that the uncertainty reduction can perform poorly (Roy and McCallum, 2001). This poor performance is highlighted when there is noise in the data labels or when there are many outliers. Therefore, uncertainty reduction criterion is not well accepted in the literature.

2.3.3 Version Space Reduction

Version space reduction algorithms try to pick a good classifier from a set of classifiers, $\mathcal{F}$, called the hypotheses space, known to the algorithm. The version space refers to the set of all hypothesis from $\mathcal{F}$, which are consistent with the training data set $\mathcal{D}$ presented to the agent. Seung et al. (1992) presented a popular algorithm called Query by Committee (QBC) based on previous work by Cohn et al. (1994). In their work Seung et al. (1992) considered the noise-free, non-agnostic case. That is, it is assumed that the hypothesis space $\mathcal{F}$ chosen by the Learner is such that the labels (Y_x) generated by the Environment satisfy $Y_x = h^*(x)$ for all $x \in \Omega$, where $h^* \in \mathcal{F}$ is the unknown function that the Learner actually tries to identify.

The *version space* $\mathcal{F}_n \subset \mathcal{F}$ after n examples is the set of all classifiers $f \in \mathcal{F}$ that are currently consistent with the labeled samples seen so far. If $\mathcal{D} = \{(x_i, Y_i) : 1 \leq i \leq n\}$ represents the labeled sample, then

$$\mathcal{F}_n = \{f \in \mathcal{F} : f(x_i) = Y_i, 1 \leq i \leq n\}.$$

The idea of the algorithm is to request label, Y_i, for a sample $x_i \in P$ in the pool P, for which the hypotheses in the version space is most split. To understand the meaning of this, let $\mathcal{F}_{n,x,y} = \{f \in \mathcal{F}_n : f(x) = y\}$. Assume we have a way to measure the "size" of a set of functions. To be specific, let us call $m(\mathcal{G})$ the size of $\mathcal{G} \subset \mathcal{F}$. (If all these sets are finite, we could e.g. simply count the number of functions in these sets. We shall comment on further possibilities for infinite sets later.) A query then is said to induce a balanced split if $m(\mathcal{F}_{n,x,+1}) \approx m(\mathcal{F}_{n,x,-1}) \approx m(\mathcal{F}_n)/2$. The measure of the balancedness of a split on x can e.g. be $b_n(x) = 4p_n(x)(1 - p_n(x))$, where $p_n(x) = m(\mathcal{F}_{n,x,+1})/m(\mathcal{F}_n)$. Given a finite set U_n of yet unlabeled examples the algorithm can search for $\operatorname{argmax}_x p_n(x)$.

The next detail is what is a good measure of the size of a hypothesis space. Given a finite pool $P = \{x_1, \ldots, x_m\} \subset \Omega$, one idea is to let $m(\mathcal{G}) = |\{(f(x_1), \ldots, f(x_m)) : f \in G\}|$, i.e., count the equivalence classes of G where two functions in G are called equivalent when they agree on all the examples in the pool P. Sometimes it is possible to put a probability measure m on the set of functions in $\mathcal{F}$ even though $\mathcal{F}$ is infinite (e.g., when the classifiers in $\mathcal{F}$ can be uniquely identified with a finite dimensional parameter, such as e.g. when $\mathcal{F}$ contains neural nets of a certain size). Still, if the hypotheses space is infinite, most of the time the version space will be infinite, too. Then one needs to resort to approximating the measure of the function spaces by sampling. If, for example, one samples K functions following m restricted to $\mathcal{F}_n$, say $f_1, \ldots, f_K$, then $m(\mathcal{F}_{n,x,y})$ can be approximated by counting how many of $f_1, \ldots, f_K$ satisfy $f(x) = y$ and dividing this count by K. The QBC algorithms does exactly this. In each iteration, the functions $f_1, \ldots, f_K$ are sampled from the version space using Gibbs sampling. Seung et al. (1992) showed empirically that the labeled sample complexity of the QBC algorithm is exponentially less than that of any passive learner's.

Freund et al. (1997) further explored the QBC algorithm and proved that the findings of Seung et al. (1992) regarding the exponential speed-up hold provided some conditions are true, at least in a Bayesian setting for a simplified version of the above algorithm. In the Bayesian setting the true hypothesis is drawn from a known (prior) distribution over the class of hypotheses. That is the distribution is assumed to be known to the Learner, just like a distribution over Ω. The goal is to minimize the average loss, where the average is taken with respect to both the distribution generating the hypotheses and the distribution over the domain. The main result of this paper is as follows: Assume that the expected information gain of QBC is lower bounded by γ independently of what labeled examples were shown to QBC. Then, for any $\epsilon > 0$, $0 < \delta < 1$, it holds with probability $1 - \delta$ that the number of labeled samples required by QBC to achieve ϵ-accuracy on the average is bounded by

$$\tilde{O}\left(\frac{d \log \frac{1}{\epsilon}}{g}\right), \tag{2.5}$$

where d is the VC-dimension of $\mathcal{F}$. They proved that the expected information gain, g, from a single query is at least equal to 0.672 bits for the case when $\mathcal{F}$ is the set of homogeneous linear separators and Ω is the d-dimensional unit ball and when the priors over the homogeneous linear separators and Ω are both close to uniform. This bound is quite an improvement compared to the bound of the Passive Learning algorithm:

$$O\left(\frac{d}{\epsilon}\right). \tag{2.6}$$

Hence the QBC algorithm is a huge improvement over the random sampling algorithm.

Although the theoretical results for QBC are impressive in practice however, the assumptions underlying this approach seriously limit its applicability. The analysis of the QBC algorithm makes three assumptions:

1. Uniform distribution over the domain and the hypotheses space
2. Our hypothesis space contains the true hypothesis f^*
3. There is no-noise present in the data labels

These assumptions are required for the theoretical bounds developed by Freund et al. (1997). The most notable extension of this algorithm comes from Dasgupta et al. (2007), where all three assumptions are relaxed. However, when the first assumption is broken the exponential gains can disappear.

Extensions

Current work by Dasgupta et al. (2007) have eliminated all three assumptions mentioned above and proved that the performance of an active learner is never worse than that of a passive learner and in some cases an active learner's labeled samples complexity is exponentially less than a passive learner's. Their algorithm is also a pool-based algorithm. We describe their algorithm next.

Let P be the pool of unlabeled samples provided for the algorithm. Furthermore, the labels presented by the Environment may be noisy. The Learner maintains two sets of data: $\hat{S}$ contains samples with imputed labels. For the elements of this set the Learner did not request a label for, but during learning the Learner became convinced of the label and so recorded the example with the "imputed label" in $\hat{S}$. The set T contains the labeled samples for which the Learner requested a label from the Environment.

For each new unlabeled sample the Learner learns two classifiers, f_{+1} and f_{-1} (for binary classification):

1. f_{+1} is consistent with the Learner's best guess of labels for $\hat{S} \cup \{(x, +1)\}$ and has minimal empirical error on T

Agnostic Selective Sampling algorithm
Input: $P = \{x_1, x_2, \cdots, x_m\}$
Initialization: $\hat{S}_0 = 0$ and $T_0 = 0$
for $i \in \{1, 2, \cdots, m\}$

1. $f_{+1} := \text{argmin}\{\hat{L}(f, T_{i-1}) : f \in F \text{ and } \hat{L}(f, \hat{S}_{i-1} \cup \{(x, +1)\}) = 0\}$
2. $f_{-1} := \text{argmin}\{\hat{L}(f, T_{i-1}) : f \in F \text{ and } \hat{L}(f, \hat{S}_{i-1} \cup \{(x, -1)\}) = 0\}$
3. $y := \text{argmin}_{z \in \{-1,+1\}} \hat{L}(f_z, \hat{S}_{i-1} \cup T_{i-1})$
4. **if** $\hat{L}(f_y, \hat{S}_{i-1} \cup T_{i-1}) < \hat{L}(f_{-y}, \hat{S}_{i-1} \cup T_{i-1}) - \Delta_{i-1}$
   then $\hat{S}_i := \hat{S}_{i-1} \cup \{(x, y)\}$ and $T_i := T_{i-1}$
   else request the label Y_i for x_i and
   let $\hat{S}_i := \hat{S}_{i-1}$, $T_i := T_{i-1} \cup \{(x, Y_i)\}$

Table 2.2: Agnostic Selective Sampling algorithm. The algorithm calculates empirical error for f_{+1} and f_{-1}. If the difference between the two empirical errors is greater than a threshold then the algorithm knows the label for a sample. Otherwise, the algorithm requests a label from the Environment.

2. f_{-1} is consistent with the learner's best guess of labels for $\hat{S} \cup \{(x, -1)\}$ and has minimal empirical error on T

If an appropriate hypothesis does not exist then the algorithm just returns a special object whose empirical error is defined to be maximal.

If the empirical error, measured on $\hat{S} \cup T$, for f_{+1} is greater than the empirical error for f_{-1} by some threshold, Δ, then the Learner will feel confident about the label of the new sample (and vice-a-versa). Otherwise, it will request a label for the sample.

This algorithm, called Agnostic Selective Sampling, is presented Table 2.2. In the algorithm's description $\hat{L}(f, S)$ denotes the number of classification errors on the labeled set S by f.

Dasgupta et al. (2007) show that if the optimal hypothesis, $f^* \in F$ achieves an error of ν then with high probability the maximum number of labeled sample required by the Agnostic Selective Sampling algorithm can be bounded by $\tilde{O}((d/\varepsilon)(1 + \nu/\varepsilon))$, where d is the VC dimension of the hypothesis space F, which is similar to the (2.6) bound.

Dasgupta et al. (2007) also highlighted the cases where an active learner performs exponentially better than a passive learner. These cases are defined in terms of the so-called *disagreement coefficient* $\mathcal{Z}(\mathcal{P}, F, \varepsilon)$, which is defined as follows: Let $\mathcal{P}$ be a distribution over $\Omega \times \{-1, +1\}$, $(X, Y) \sim \mathcal{P}$ and $f^* = \text{argmin}_{f \in F} \mathbb{P}(f(X) \neq Y)$ be the best classifier in our hypothesis class. Let $B(f, r) = \{f' \in F : \mathbb{P}(f(X) \neq f'(X)) \leq r\}$ the "ball" centered around f of radius r in the pseudo-metric $d(f, f') = \mathbb{P}(f(X) \neq f'(X))$. The idea of the disagreement coefficient is to measure how quickly the disagreement grows in the surroundings of

f^* between f^* and classifiers in $B(f^*, r)$. Formally,

$$\mathcal{Z}(\mathcal{P}, F, \varepsilon) = \sup \left\{ \frac{\mathbb{E}\left[\sup_{h \in B(f^*,r)} \mathbb{I}_{\{h(X) \neq f^*(X)\}}\right]}{r} : r \geq \nu + \varepsilon \right\}. \tag{2.7}$$

Dasgupta et al. (2007) show that if a hypothesis class F and distribution $\mathcal{P}$ has a disagreement coefficient $\mathcal{Z}$ bounded independently of $1/(\varepsilon+\nu)$ then the label sample complexity of the Agnostic Selective Sampling algorithm is $\tilde{O}(\mathcal{Z} d \log^2(1/\varepsilon))$ to achieve an error $\varepsilon \approx \nu$. They argue that this is the case e.g. when F consists of all the homogeneous linear separators and when the marginal of $\mathcal{P}$ is the uniform distribution over the unit sphere of $\mathbb{R}^d$, in which case $\mathcal{Z}(\mathcal{P}, F, \varepsilon) = \Theta(\sqrt{d})$.

2.3.4 Expected Error Reduction

Roy and McCallum (2001) state that uncertainty reduction and version space reduction methods do not necessarily reduce the error on each step. They can in fact spend a majority of their time chasing outliers. Roy and McCallum (2001) also introduced an algorithm to directly reduce the prediction error of a classifier. The method is targeted to learning algorithms that search for probabilistic classifiers. Such classifiers, as described earlier, assign an estimate of the probability of the possible labels to each possible input. Let π be such a classifier. The expected prediction error when using π can e.g. be defined as follows:

$$E_\pi = \mathbb{E}\left[L(\mathbb{P}\left(Y|X\right), \pi(Y|X))\right],$$

where L is a loss function (e.g., $L(p, q) = (p-q)^2$), and (X, Y) are jointly distributed according some unknown distribution $\mathcal{D}$. The expectation with respect to X can be approximated by an average over the examples in the pool of examples available for training, assuming that these come from $\mathcal{D}_X$ the marginal of X. The idea is to pick the example x that gives the model with the lowest expected prediction error when the model π is retrained with (x, Y_x) added to the set of labeled examples where Y_x is such that $\mathbb{P}\left(Y_x = y\right) = \mathbb{P}\left(Y = y|X = x\right)$. Since $\mathbb{P}\left(Y = y|X\right)$ is unavailable it is replaced by $\pi(y|x)$, when computing the expected prediction error the target labels.

The straightforward implementation of this algorithm would lead to an inefficient method. Roy and McCallum (2001) propose to use sampling techniques (i.e., sample points from the data in order to estimate the average loss over the pool) and other tricks to speed up these calculations. The algorithm of expected error reduction has multiple steps therefore we give the pseudo code for the algorithm in Table 2.3

Implementing this algorithm is computationally expensive, since for each example you have to re-train the classifier and re-calculate the expected prediction error. If the number of unlabeled examples is really large then these computations become very expensive. Roy

Expected Error Reduction algorithm

1. Train the classifier using the current training set of m labeled examples and obtain π
2. For each example x_i in the unlabeled data set do:

   For each possible label y do:

   a Add the example (x_i, y) to the current training set

   b Retrain the classifier with the new example to get $\pi_{(x_i,y)}$

   c Estimate the prediction error $L_{x_i,y}$ of $\pi_{(x_i,y)}$

   Compute the expected prediction error after seeing the label of x_i as follows:

$$L_{x_i} = \sum_y \pi(y|x_i) L_{x_i,y}.$$

3. Request the labeling for the example x_i for which the L_{x_i} is minimum
4. Go to 1

Table 2.3: Expected Error Reduction Algorithm.

and McCallum (2001) acknowledge this fact and propose using sampling methods. Also, one can save computation time by choosing classifiers that can learn incrementally. Thus when a new example is added it takes a short amount of computational time to re-train the classifier.

Roy and McCallum (2001) empirically showed that the algorithm indeed performs well and the performance is as good as the QBC algorithm. They claim that the main advantage of their algorithm comes early on in the training, when we have only seen few labeled samples. This algorithm produces low prediction error with only a few samples, whereas the QBC and the uncertainty sampling algorithms take much longer. The reason for the slow performance of the QBC and the uncertainty sampling algorithms is that they focus on the outliers and thus need more samples to attain the same level of prediction quality.

Further research by Chapelle (2005) indicates the performance of this algorithm greatly depends on the estimate of the prediction error. If we have a close estimate of the prediction error then the algorithm performs well otherwise it performs poorly. Chapelle tested various techniques to measure the expected prediction error and then comparing the performance of the algorithm on test data. They found that the unlabeled examples far from the labeled examples, through some distance metrics, are classified with the same confidence as the unlabeled examples close to the labeled examples. That is the algorithm does not take advantage of its confidence about the unlabeled examples. Chapelle demonstrated that adding this information provides a better estimate of the prediction error.

Empirically this algorithm also performs better than the Passive Learning algorithm and outperforms some of the other Active Learning algorithm. However, theoretical results about the performance of this algorithm are still missing from the literature. We do not know whether the results presented about this algorithm generalize to other domains or not.

2.3.5 Logistic Regression with A-Optimality

Earlier we looked at A-optimal experiment design for the regression problem. The statistical principle underlying this criterion was to minimize the expected squared 2-norm parameter error. Schein (2005) have adapted this criterion for the classification settings. Similar to the regression setting, we have a parameterized function with parameter θ. However, the output is not a linear combination of the parameter vector θ and input vector x. Instead, the output is generated through the logistic function $f(x) = \frac{1}{1+\exp^{-\theta^T x}}$, which is intended to approximate the probability of seeing (say) label $+1$ given the input z. So far we have discussed binary classification where one has two classes $+1$ or -1. In a multi-class classification one has, generally, more than two classes. We set up the multi-class logistic regression in the following way: Let $\theta = (\theta_y)_{y \in \mathcal{Y}}$, where $\mathcal{Y}$ is the set of possible labels. Then

$$\pi(y, x, \theta) = \frac{\exp(\theta_y^T x)}{\sum_{y'} \exp(\theta_{y'}^T x)} \tag{2.8}$$

The interpretation is that $\pi(y, x, \theta)$ approximates the probability of seeing the label y given the input example x.

One possible goal is to minimize the squared loss of predicting the class label probabilities:

$$L(\pi) = \mathbb{E}\left[(\pi(Y, X, \hat{\theta}) - \mathbb{P}(Y|X))^2\right].$$

Here (X, Y) are jointly distributed over $\mathbb{R}^d \times \mathcal{Y}$ and $\hat{\theta}$ is the parameters tuned with a the dataset $\mathcal{D}$ available for training, which is assumed to have n independent copies of (X, Y). Then, the bias-variance rule gives

$$\begin{aligned} L(\pi) &= \mathbb{E}\left[(\mathbb{E}\left[\pi(Y, X, \hat{\theta})|\mathcal{D}\right] - \mathbb{P}(Y|X))^2\right] \\ &+ \mathbb{E}\left[(\mathbb{E}\left[\pi(Y, X, \hat{\theta})|\mathcal{D}\right] - \pi(Y, X, \hat{\theta}))^2\right]. \end{aligned}$$

Schein (2005) suggest to concentrate on the minimization of the second term and they argue that the minimization of this term can be considered to be a generalization of A-optimality. Assume that $\hat{\theta}$ is obtained by maximum likelihood estimation. Using the Taylor series expansion of π around $\theta = \mathbb{E}\left[\hat{\theta}\right]$, we can approximate the variance as follows

$$\begin{aligned} \mathrm{Var}\left[(\pi(y, x, \hat{\theta})\right] &\simeq \mathrm{Var}\left[g(y, x, \theta)(\hat{\theta} - \theta)\right] &(2.9)\\ &= g(x, y, \theta)^T M^{-1}(\theta) g(x, y, \theta), &(2.10) \end{aligned}$$

where g is the gradient of π with respect to θ and $M(\theta)$ is the Fisher information matrix underlying our model, where it was exploited that $\hat{\theta}$ is estimated using maximum likelihood so the inverse covariance matrix of $\hat{\theta}$ is equal to the Fisher information matrix. Now, in order to approximate $\text{Var}\left[(\pi(Y, X, \hat{\theta})\right]$, we can use a set of labeled examples, $\mathcal{D}' = \{(X'_1, Y'_1), \ldots, (X'_m, Y'_m)\}$ that is composed of i.i.d. copies of (X, Y), which are in fact independent of the training examples:

$$\begin{aligned}
\text{Var}\left[(\pi(Y, X, \hat{\theta})\right] &\approx \frac{1}{n}\sum_{i=1}^{m} g(X'_i, Y'_i, \theta)^T M^{-1}(\theta) g(X'_i, Y'_i, \theta) \\
&= \frac{1}{m}\sum_{i=1}^{m} \text{trace}\left\{g(X'_i, Y'_i, \theta) g(X'_i, Y'_i, \theta)^T M^{-1}(\theta)\right\} \\
&= \text{trace}\left\{A_m(\theta) M^{-1}(\theta)\right\},
\end{aligned}$$

where $A_m(\theta) = \frac{1}{m}\sum_{i=1}^{m} g(X'_i, Y'_i, \theta) g(X'_i, Y'_i, \theta)^T$. Denote the right hand side of the last displayed equation by $V(\theta)$. Implicit in Schein (2005) is that for computational purposes one should use $\hat{\theta}$ above instead of θ. Further, they propose to pick the next example x that minimizes

$$\sum_{y \in \mathcal{Y}} V(\hat{\theta}_{x,y}) \pi(y, x, \hat{\theta}),$$

where $\hat{\theta}$ is the current parameter estimate and $\hat{\theta}_{x,y}$ is the estimate obtained when (x, y) is added to the set of labeled examples $\mathcal{D}$. This average is intended to approximate

$$\mathbb{E}\left[V(\hat{\theta}_{X,Y}) | X = x, \mathcal{D}\right] = \sum_{y \in \mathcal{Y}} V(\hat{\theta}_{x,y}) \mathbb{P}\left(Y = y | X = x\right).$$

Thus, we see that just like in the expected prediction error minimization algorithm, the unknown label distribution $\mathbb{P}\left(Y = y | X = x\right)$ is optimistically replaced by its current estimate (i.e., by $\pi(y, x, \hat{\theta})$).

Results

Schein (2005) compared their algorithm with the extensions to the QBC algorithms by McCallum and Nigam (1998) and Abe and Mamitsuka (1998). They found that their algorithm competes well with the other algorithms and never performs worse than the Passive Learning algorithm. The advantage of this algorithm is that it can handle noise in the data labels unlike the uncertainty reduction and the QBC algorithms. Schein (2005) report that their algorithm performs better than the expected error reduction algorithm (Roy and McCallum, 2001), as the expected error algorithm's performance greatly depends on the quality of the approximation of the prediction error, which can be poor for some cases (for example, Hierarchical data in Schein (2005) experiment).

2.4 Active Learning in Reinforcement Learning

In Reinforcement Learning (RL) problems the Learner is faced with the problem of taking a sequence of actions to minimize the total cost (or maximize the total reward). The actions of the Learner influence the state of the Environment, causing it to make stochastic transitions. The set of actions available to the Learner can be discrete or continuous and the set can depend on the actual state of the Environment. The Learner is able to observe the state transitions and the costs associated with these. Formally, the problem is to find a policy (a mapping of the history to the set actions) that minimizes the total cost. The costs (or rewards) can depend on the state transitions (i.e., the current and the next state, as well as the action taken) and can be "noisy". The stochastic process associated with the execution of a policy is called a Markov Decision Process (MDP).

This framework is very powerful. In particular, all the previous pool-based active learning problems can be cast as a special case of this framework: If the pool size is m, the state of the environment could be an integer-valued m-dimensional vector, the ith component denoting the number of times the Learner requested the label of the ith example. The initial state is the one when all components are zero. The actions of the Learner are: request a label for an example or quit while specifying a predictor f.

Now, various ways of formalizing Active Learning with this framework are possible. One possibility is to assign a fixed cost $c > 0$ to querying the label of any example and to assign the prediction loss $L(f)$ to the transition when the Learner chooses to quit and the proposes f as the predictor. The optimal policy then minimizes the sum of the costs of querying the labels and the prediction loss. In another problem formulation the goal is to find a policy that maximizes the rate of decay of $L(f_t)$, where f_t is the predictor that the Learner would produce had he have to quite at time t.

Although these reductions are interesting, our purpose here is not to study these, but the general issue of exploration vs. exploitation, a common dilemma underlying learning to act optimally while controlling an unknown, stochastic system.

The exploration vs. exploitation dilemma refers to the fact that a Learner who learns about its Environment will only know some estimates of the long-term values of the actions available in some state (assuming rewards). If the Learner decides to go with the action whose value estimate is the best, we say that the Learner is making an *exploitation* step, while in the other case the Learner is said to *explore*. A Learner who always exploits can fail to identify the optimal actions if the good actions initially were observed with bad outcomes due to the noise in the system. Thus, a Learner needs to balance exploration and exploration in order not to miss the best actions.

In the next section we will focus on a very specific problem with one state and a discrete set of action, known as the finite-armed bandit problem. We also look at a popular algorithm

Algorithm UCB

Initialization: Play each arm once.
At time n do:

1. calculate a biased value for each arm k based on the average reward $\hat{\mu}_k$:

$$\hat{r}_k = \hat{\mu}_k + \sqrt{\frac{2b^2 \ln(n-1)}{T_{k,n-1}}}$$

2. play the arm with the highest $\hat{r}_k$ value

Table 2.4: UCB algorithm calculates the empirical expected reward, $\hat{\mu}_k$, for each bandit arm, k, and adds an upper confidence interval bonus to the expected value

by Auer et al. (2002) that minimizes the expected regret, the difference between optimal reward and the algorithm's reward.

2.4.1 Upper Confidence Bound (UCB)

In a multi-arm bandit problem, each arm k produces a reward X_k, which is randomly distributed and independent from the other $k-1$ arms reward. Let the unknown expected value of the kth arm be μ_k. Assume that the range of the rewards is known. We denote by T_{kn} the number of times arm k is pulled during the first n steps. The objective of the problem is to minimize the expected regret, which we define as:

$$\mathbb{E}[\hat{R}_n] = \mu^* n - \mu_k \sum_{k=1}^{K} \mathbb{E}[T_{kn}],$$

where $\mu^* = \max_k \mu_k$ is the best expected payoff across the arms.

This implies we want to minimize the expected loss suffered due to the fact that our algorithm may choose a sub-optimal arm at some step. Again, the Learner could choose to select the arm whose payoff estimate is the largest, but if he does so always, he might miss the best arm if the best arm is unlucky during the initial trials. Although the problem looks simple, it is sufficiently deep to illustrate the intricacies of the exploration vs. exploitation issue.

Algorithm

Auer et al. (2002) developed an algorithm to solve the bandit problem efficiently. The algorithm calculates the empirical expected reward, $\hat{\mu}_k$, for each bandit arm, k, and adds an upper confidence interval bonus to the expected value. The algorithm is presented in Table 2.4 and works as follows:

The algorithm tops the average reward of each arm by a confidence bounds, essentially building an upper confidence bound for the value of each arm. This confidence interval is based on Hoeffding (Hoeffding, 1963) bounds and it was assumed that the rewards are in an interval of size b. The confidence interval is such that the more samples you have from an arm, the more confident the algorithm is about the expected reward value of an arm. Whereas, if the number of samples is low for an arm, the confidence bound will be large. Notice that the confidence intervals grow as a function of the number of steps n. This makes sure that all the arms will be selected eventually infinitely often.

If the rewards are bounded by an interval of length b, then the following bound holds for the expected regret of the UCB algorithm:

$$\mathbb{E}[\hat{R}_n] \leq 8 \sum_{k:\mu_k<\mu^*} \left(\frac{b^2}{\mu^* - \mu_k} \right) \log(n) + O(1) \tag{2.11}$$

A Passive Learning algorithm will choose uniformly randomly among all the bandit arms. The expected regret in such case would be: $\mathbb{E}[\hat{R}_n] = \Omega(n)$. Therefore the UCB algorithm provides exponential gains over the Passive Learning algorithm.

Auer et al. (2002) also presented a variant of the UCB algorithm called UCB-Tuned in which b^2 in the confidence bound term is replaced by the estimate of the variance. Intuitively this is a good idea, since if a sub-optimal arm has a low variance then we do not need to try out that arm so often to estimate its expected reward. On the other hand, if we do not use variance then all arms will receive a bonus based on the range of the reward. Since the reward range could be very large, some of the sub-optimal arms could receive high bonus and thus misguide the algorithm. Experimentally Auer et al. (2002) reported that UCB-Tuned algorithm always performed better than the UCB algorithm.

Audibert et al. (2007) further studied the theoretical properties of the UCB-Tuned algorithm. They calculated the estimated variance of each arm and then used what they called "empirical Bernstein bounds" to calculate the confidence bounds on the mean value of each arm, similar to the original UCB algorithm. The new algorithm called UCB-V is given in Table 2.5. The algorithm takes one additional parameter, $\xi_{s,n}$, which is the exploration schedule taking two arguments: a non-negative real number s, and the current number of plays n.

The regret bound of this algorithm replaces the b^2 (in equation (2.11)) with the variance of the arm (equation (2.12)):

$$\mathbb{E}[\hat{R}_n] \leq 10 \sum_{k:\mu_k<\mu^*} \left(\frac{\sigma_k^2}{\mu^* - \mu_k} + 2b \right) \log(n). \tag{2.12}$$

This regret bound is for the exploration schedule, $\xi = 1.2 \log n$. In general the constant term is related to the constant term in the exploration schedule.

UCB-V

Input: $\xi_{s,n}$ - exploration schedule
Initialization: Play each arm twice.
At time n do:

1. calculate the biased value of each arm k based on the average reward $\hat{\mu}_k$ and the estimated variance V_k:

$$\hat{r}_k = \hat{\mu}_k + \sqrt{\frac{2V_k\xi_{T_{k,n-1},n}}{T_{k,n-1}}} + \frac{3b\xi_{T_{k,n-1},n}}{T_{k,n-1}}.$$

2. play the arm with the highest $\hat{r}_k$ value

Table 2.5: UCB-V algorithm calculate the biased value of each arm k based on the average reward $\hat{\mu}_k$ and the estimated variance V_k. The algorithm then plays the arm with the highest biased reward value.

This work is directly related to the work presented in this thesis. Our work also deals with the exploration versus exploitation tradeoff. In addition, we will study an upper confidence bound based variant of our algorithms.

2.5 Discussion and Summary

We have looked at various Active Learning approaches in the literature. There are cases (Freund et al., 1997; Dasgupta et al., 2007; Auer et al., 2002) where we know that Active Learning achieves exponentially better performance than the Passive Learning algorithm. However, is there any advantage of using the Active Learning framework in the other cases? Dasgupta (2005) have shown that in some cases, Active Learning algorithms can not perform better than the Passive Learning. These results hold for minimax bounds. Recently Balcan et al. (2008) looked at the individual rates for the Active Learning algorithms and prove that the Active Learning does have a strict advantage over the Passive Learning. Balcan et al. (2008) compared the labeled sample complexity of the Active Learning algorithms to that of the Passive Learning algorithms. For the noise-free classification problems with finite VC dimension, d, they showed that the active learner's labeled sample complexity, S_a is strictly better than the passive learner's labeled sample complexity, S_p. i.e.

$$S_a(\varepsilon, \delta, h) = o(S_p(\varepsilon/4, \delta, h)). \tag{2.13}$$

Balcan et al. (2008) present an algorithm that achieves (2.13). The algorithm requires that the hypothesis class, F, can be broken down into many sub-classes F_i, such that

$$F = \bigcup_{i=1}^{\infty} F_i \tag{2.14}$$

The algorithm takes two parameters: number of sample $t \geq S_a(\varepsilon, \delta, h)$ and δ. The algorithm works by identifying the hypothesis space F_i that contains the positive labels and then uses algorithms like QBC to find the boundary. The algorithm produces a classifier h_t that has prediction error $\hat{\varepsilon}$ and $P(\hat{\varepsilon} \leq \varepsilon) \geq 1 - \delta$. For further details on their algorithm, please refer to Balcan et al. (2008).

Their results show that asymptotically as the prediction error ε approaches 0 the Active Learning algorithm's performance far exceeds the Passive Learning algorithm's performance. It turns out though, that their algorithm can perform arbitrarily poor in the initial phase of searching for the hypothesis class F_i containing the positive labels, compared to the Passive Learner. Therefore, Balcan et al. (2008) algorithm has nice theoretical properties, but it does not seem to lead to a practically useful algorithm.

2.5.1 What's Missing

Balcan et al. (2008) work shows that Active Learning can be advantageous. However, what is still missing from the literature is a general algorithm that can be guaranteed to be not much worse than the Passive Learning algorithm in the initial stage and which is asymptotically better. We can formalize this as follows:

Given a hypothesis class F and δ, suppose after t labels the active learner prediction error is $\hat{\varepsilon}_a(t)$. Similarly, the prediction error for the passive learner is $\hat{\varepsilon}_p(t)$, then we want the Active Learning algorithm to satisfy the following two properties:

$$\hat{\varepsilon}_a(t) \ll \hat{\varepsilon}_p(t) \qquad \text{for some } t, \text{ and} \tag{2.15}$$

$$\hat{\varepsilon}_a(\hat{t}) \leq C\hat{\varepsilon}_p(\hat{t}) \quad \text{for any } \hat{t} < t \text{ and for some constant } C \tag{2.16}$$

Figure 2.1 illustrates this requirement.

2.5.2 Summary

In this chapter we looked at various Active Learning algorithms that currently exist in the literature. We looked at algorithms for three types of problems: regression, classification and bandit problems. We looked at several algorithms that use the Active Learning framework. These algorithms' performance is exponentially better than the Passive Learning algorithm, however this guarantee is limited to a specific set of cases only. In some cases, in fact, Active Learning has shown no performance gain over the Passive Learning in the minimax setting. However, Balcan et al. (2008) proved that the Active Learning algorithms do have an advantage over the Passive Learning algorithms when one looks at individual rates. The

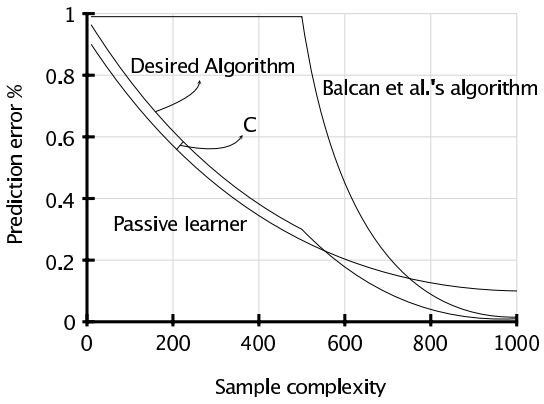

Figure 2.1: An illustration of the properties of the desired Active Learning algorithm.

algorithm proposed by Balcan et al. (2008) is general but can not be applied in practice as the initial phase of the algorithm can take arbitrarily long. Therefore, what is missing from this field is a general purpose algorithm which performs not much worse than the Passive Learning algorithm in the initial phase and considerably better in the later phase.

In the next two chapters we present algorithms that perform well in both phases for a restricted problem class. Our algorithm does not need to observe a lot of samples in order to show improvements. In fact the improvements can be seen from as little as few hundred samples.

Chapter 3

Active Learning for Multi-Armed Bandits

Consider the task of quality assurance in a factory equipped with a number of machines where each machine produces the product of different quality. The difference in quality can come due to various reasons, such as the age of the machine, the wear and tear on the various parts of the machine, the worker working on the machine etc. The quality can be monitored by inspecting the products produced. Multiple measurements are necessary to control the precision of the quality estimates. The objective is to keep the precision of the estimates equal across all the machines while minimizing the total cost associated with an inspection.

Similarly, clinical trials are conducted to test the effect of a drug on a number of patients. The patients can be grouped based on their gender, age group, and other medical conditions. One is interested in testing the effect of the drug on each of these groups. Again, the response has randomness due to each patients condition within each group. Thus, one must test the effect of the drug on a number of patients in each group to measure its effect properly. In order to minimize cost, one needs to minimize the total number of patients in any clinical trial.

An inspection or drug outcome can be modeled as a random number say between zero and one, one meaning the best, zero implying poor quality or no affect of the drug. Then the task is to measure the mean response of the random variable associated with each machine or group of patients. In order to estimate the mean value, one samples each random variable sequentially. In order to reduce our loss, we sample those machines more frequently that have high variance. However, the allocation is based on an estimate of the variance and the estimate could be wrong; therefore, one must not ignore other machines. Thus the problem faced here is similar to the exploration-exploitation dilemma in bandit problems where a greedy policy might incur a large loss if the payoff of the optimal arm is underestimated.

The general problem considered in this chapter is one where there are heterogeneous

groups and the goal is to measure the mean response of each group to equal precision. The problem is modeled as a multi-armed bandit problem. The Learner must decide which bandit arms it wants to sample at any given time. A formal definition of the problem is presented next. Then a Passive Learner algorithm for solving this problem is presented. Next, two algorithms that solve this problem using the Active Learning framework are presented. The first algorithm is a batch algorithm adapted from the algorithm presented by Etore and Jourdain (2007) for the stratified sampling problem. The second algorithm is a novice, incremental algorithm in joint work with András Antos and Csaba Szepesvári (Antos et al., 2008). The excess loss suffered by this algorithm, apart from logarithmic factors, scales as $n^{-3/2}$.

3.1 Problem Formulation

One is interested in estimating the expected values (μ_k) of some distributions ($\mathcal{D}_k$), each associated with an arm, k. If K is the number of arms then for any k, $1 \leq k \leq K$, the Learner can draw independent samples $\{X_{kt}\}_t$ from $\mathcal{D}_k$ where X_{kt} is the value obtained at time t from arm k. The samples are drawn sequentially: given the information collected up to trial n the Learner can decide which arm to choose next. The expected value, μ_k, can be estimated by computing the sample means of the respective arms:

$$\hat{\mu}_{kn} = \frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} X_{kt},$$

where T_{kn} is the number of samples for arm k. The error of estimate of arm k is measured with the expected squared error:

$$\begin{aligned} L_{kn} &= \mathbb{E}\left[(\hat{\mu}_{kn} - \mu_k)^2\right] \\ &= \frac{\sigma_k^2}{T_k n}, \end{aligned}$$

where $\sigma_k^2 = \mathrm{Var}\left[X_k\right]$. The overall loss is measured by the worst-case loss over the K arms:

$$L_n = \max_{1 \leq k \leq K} L_{kn}. \tag{3.1}$$

This expresses the desire that all estimates are equally important.

If one has full knowledge about the variance of each bandit arm, then the optimal allocation strategy would be to allocate in proportion to the variance of each arm. Thus:

$$T_{kn}^* = n\frac{\sigma_k^2}{\Sigma^2} = \lambda_k n.$$

Here $\Sigma^2 = \sum_{j=1}^K \sigma_j^2$ is the sum of the variances and

$$\lambda_k = \frac{\sigma_k^2}{\Sigma^2}.$$

The corresponding loss is

$$L_n^* = \frac{\Sigma^2}{n}.$$

The optimal allocation is easy to extend to the case when some arms have zero variance. Clearly, it is both necessary and sufficient to make a single observation on such arms. The case when all variances are zero (i.e., $\Sigma^2 = 0$) is uninteresting, hence we will assume from now on that $\Sigma^2 > 0$.

The goal of the Learner is to minimize this loss. An effective solution is an algorithm $\mathcal{A}$ such that the $L_n = L_n(\mathcal{A})$ is close to the optimal loss L_n^*.

3.2 Passive Approach

A Passive Learning algorithm will sample the arms based on some distribution. Without any prior knowledge about the arms, the most obvious choice is to use the uniform distribution over all the arms. Therefore, a Passive Learner would sample equally among all arms. The loss of this algorithm is:

$$L_n(\mathcal{P}) = \frac{(\max_{1 \le k \le K} \sigma_k^2)K}{n}.$$

The excess loss suffered by this algorithm converges to the optimal loss L^* at a rate of $\Theta(1/n)$ when the optimal allocation is non-uniform.

3.3 Active Algorithms for Multi-Armed Bandits

If one has full knowledge of the distributions, the optimal strategy would be to frequently sample those arms that have higher variance. An Active Learning algorithm can estimate the variance of each arm and allocates samples accordingly, thus having higher efficiency than a Passive Learner. Next subsections present two algorithm under the Active Learning framework for the multi-armed bandits problem. Both of these algorithms decide to sample an arm based on the sampling distributions they observed. We first develop the main idea of an Active Learning algorithm that can solve the multi-armed bandits problem and then look at two different ways of implementing this idea.

3.3.1 An Active Learning Approach

Since the loss of an arm k can only decrease if the algorithm requests a new sample from $\mathcal{D}_k$, one simple idea is to request the next sample from arm k whose estimated loss, $\hat{\sigma}_{kn}^2/T_{kn}$, is the largest amongst all estimated losses. Here $\hat{\sigma}_{kn}^2$ is an estimate of the variance of the k^{th} arm based on the history. The problem with this approach is that the variance might be underestimated in which case the arm will not be selected for a long time, which prevents refining the estimated variance, ultimately resulting in a large excess loss. One simple remedy is to make sure that the estimated variances converge to their true values. This can

GFSP-MAX

Input: $\xi(i)$ - a function returning length of phase i
Initialization: Sample each arm $\xi(1)/K$ times.
for $i \geq 1$

1. calculate empirical variance for each option k:

$$\hat{\sigma}^2_{kn} = \frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} X^2_{kt} - \left(\frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} X_{kt} \right)^2$$

2. sample each arm k once
3. let $M_i = \xi(i) - K = \sum_{k=1}^{K} m_{ki}$
4. calculate

$$m_{ki} = \begin{cases} M_i/K, & \text{if } \sum_{j=1}^{K} \hat{\sigma}^2_{jn} = \emptyset \\ \frac{\hat{\sigma}^2_{kn}}{\sum_{j=1}^{K} \hat{\sigma}^2_{jn}} M_i, & \text{otherwise,} \end{cases}$$

   for each arm k

5. Sample arm k, m_{ki} times

Table 3.1: GFSP-MAX algorithm estimates the variances of each arm at the beginning of each phase and then use those estimates to allocate samples for rest of the phase.

be ensured if the algorithm is forced to select all the arms indefinitely in the limit, which is often called the method of forced selections in the bandit literature. Two algorithms that deal with this tradeoff are presented next. The first algorithm is a phase based batch algorithm, while the second algorithm is incremental.

3.3.2 Phase-Based Allocation

Etore and Jourdain (2007) present an adaptive algorithm to solve the stratified sampling problem. We adapt their algorithm for the mutli-armed bandit problem. In order to deal with the exploration-exploitation dilemma the Etore et al. algorithm uses a phase-based technique. In the beginning of each phase the algorithm chooses all arms exactly once while in the rest of the phase it can sample all the arms proportional to their respective variance estimates computed at the beginning of the phase. Initially, we have very little knowledge about the variance of each arm; therefore, we start with small phase sizes encouraging exploration. However, as we gain more and more experience we want to reduce the exploration and thus increase subsequent phase sizes. GFSP-MAX (greedy with forced selection with Phases for the max-norm problem) algorithm is presented in Table 3.1.

The algorithm computes the sample variance for each option at the beginning of a phase. It then calculates the number of samples to be alloted to each option based on the estimated

variance proportion of each option. Note that m_{ki} as defined in the algorithm are not going to be an integer. (Etore and Jourdain, 2007) describe a complicated procedure to calculate integer values that sum to M_i. However, we conjecture that rounding down and allocating the remaining samples to the options with the largest estimated allocation ratios should be as good as their algorithm. This is what we used in our experiments.

Properties

The original algorithm by Etore and Jourdain (2007) is shown to converge to the optimal allocation based on the variance for the stratified sampling problem. We conjecture that the modified algorithm also converges to the optimal allocation, since in the limit this algorithm samples each arm infinitely many times and thus the estimated variance is guaranteed to converge to the true variance. This algorithm is a batch algorithm, and requires a function, ξ, specifying the length of each phase. The length of each phase must be at least equal to K. The performance of algorithm depends on the amount of exploration, which is controlled by ξ function. Therefore, ξ needs to be carefully picked.

3.3.3 Incremental Allocation

We now look at an algorithm which balances exploration-exploitation tradeoff and works incrementally. This algorithm is a joint work with András Antos and Csaba Szepesvári (Antos et al., 2008). The main idea of this algorithm is to select the arm with the largest estimated loss at any time step. However, if some of the arms are seriously under-sampled, then in this case the most under-sampled arm is selected. It turns out that a good definition for an arm being under-sampled is if $T_{kn} \leq c\sqrt{n}$ with some constant $c > 0$. The excess loss of this algorithm, compared to an optimal algorithm that knows the variance of each arm from the beginning, decreases at a rate of $\tilde{O}(n^{-3/2})$.[1] This algorithm is referred as GAFS-MAX (greedy allocation with forced selection for max-norm value estimation) as it allocates a sample greedily to the arm with the maximum expected loss but if there is a under-sampled arm then the algorithm is forced to select that arm. The algorithm is presented in Table 3.2:

The algorithm computes the variance of each arm on each time step.[2] Then, similar to the GFSP-MAX, computes the proportion of variance for each arm. Next the algorithm decides the arm to sample by examining all the arm and choosing the arm which has the least number of samples alloted to it compared to the variance proportion of that arm. This step has an exception when any arm is severely under-sampled. In such case the algorithm picks the arm which is most under-sampled.

[1] A nonnegative sequence (a_n) is said to be $\tilde{O}(f(n))$, where $f : \mathbb{N} \to \mathbb{R}^+$, $\lim_{n\to\infty} f(n) = \infty$, if $a_n \leq Cf(n)\log(f(n))$ with a suitable constant $C > 0$.

[2] We can compute the variance estimates incrementally as well.

Algorithm GAFS-MAX

Input: c - a constant > 0 to control the amount of exploration
Initialization: In the first K trials choose each arm once
Set $T_{k,K} = 1$ for each k $(1 \leq k \leq K)$, $n = K$
At time $n + 1$ do:

1. Compute $\hat{\sigma}^2_{kn} = \frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} X^2_{kt} - \left(\frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} X_{kt} \right)^2$
2. Let $\hat{\lambda}_{kn} = \hat{\sigma}^2_{kn} / (\sum_{j=1}^{K} \hat{\sigma}^2_{jn})$ if $\sum_{j=1}^{K} \hat{\sigma}^2_{jn} \neq 0$,
   otherwise let $\hat{\lambda}_{kn} = 1/K$.
3. Let $U_n = \text{argmin}_{1 \leq k \leq K} T_{kn}$
4. Let
$$I_{n+1} = \begin{cases} U_n, & \text{if } T_{U_n,n} < \sqrt{n} + 1 \\ \text{argmax}_{1 \leq k \leq K} \frac{\hat{\lambda}_{kn}}{T_{kn}}, & \text{otherwise,} \end{cases}$$
   where in the second case ties are broken in an arbitrary, but systematic manner.
5. Choose arm I_{n+1}, let $T_{k,n+1} = T_{k,n} + \mathbb{I}_{\{I_{n+1}=k\}}$
6. Observe the feedback $X_{I_{n+1},T_{I_{n+1},n+1}}$.

Table 3.2: GAFS-MAX algorithm estimates the variance of each arm. The algorithm then allocates the next sample to the arm with the maximum expected loss. However, if there is an arm that is severely under-sampled then the algorithm samples that arm instead.

Properties

The parameter, c, controls the amount of exploration by our algorithm. If we set the parameter too close to zero, then our algorithm will essentially act greedily. In such case, if the variance for an arm is over/under estimated then we will incur a huge loss. On the other hand, if one sets a high value for the parameter value then this algorithm essentially performs like the Passive Learning algorithm, i.e. it will sample each arm equally. GAFS-MAX algorithm works quite efficiently in the middle range. The experimental section 5.1 provides more insights into the parameter c and shows the performance of our algorithm on a 5-arm bandit problem.

This algorithm converges to the optimal solution as the estimate of the variances of the arms converge. The loss (equation 3.1) suffered by this algorithm decreases at the rate of $\tilde{O}(n^{-3/2})$. The detailed proof on the rate of convergence is provided in Antos et al. (2008). A sketch of the proof is provided in the next section.

GAFS-MAX Rate of Convergence

The main result for the GAFS-MAX is the following: There exists some integer N such that for any k, $n \geq N$,

$$L_n \leq L_n^* + \tilde{O}(n^{-3/2}).$$

The detailed proof is presented in the appendix. A sketch of the proof is presented here.

The proof is broken down into multiple parts. Firstly Antos et al. (2008) prove that if $T_{kn} \geq f(n)$ then $\hat{\lambda}_{kn}$ converges to λ_k at a rate of $O(1/f(n)^{1/2})$ (Lemma 2 in Antos et al. (2008)). The next result shows that we can replace T_{kn}/n for $\hat{\lambda}_{kn}$ in the first result and get a bound on the difference between T_{kn}/n and λ_k (Lemma 3). Since the GAFS-MAX algorithm is forced to select from each arm $\sqrt{n}$ times, then from the first result we can conclude that $\hat{\lambda}_{kn}$ converges to λ_k at a rate of at least $1/n^{1/4}$. From the second result it can be shown that T_{kn} grows at least as fast as $\lambda_k/2n$, i.e., linearly in n. Using again the first result one gets that $\hat{\lambda}_{kn} - \lambda_k$ decays at least as fast as $1/n^{1/2}$, which, using the second result, allows us to conclude that $T_{kn}/n - \lambda_k$ converges to zero at the rate of $1/n^{1/2}$. Resorting to Wald's second identity then allows us to prove that the excess loss $L_{kn} - L_n^*$ decays at the rate of $1/(n^{3/2})$ with high probability.

3.4 Relation to Optimal Experiment Design

The above problem is a special optimal experiment design problem for linear regression with heteroscedastic noise. To see this define e_i to be ith unit vector in $\mathbb{R}^K$: $e_i^T = (0, \ldots, 0, 1, 0, \ldots, 0)^T$, where 1 is at the ith position in e_i. Let $\theta \in \mathbb{R}^K$ be the vector of the means of distributions D_k: $\theta_k = \mu_k$. Let the domain of linear regression be

$\mathcal{X} = \{e_1, \ldots, e_K\}$. Then, upon choosing vector e_k, the response can be written as

$$Y_{e_k} = \theta^T e_k + Z_{e_k}, \tag{3.2}$$

where Z_{e_k} is zero-mean with $\mathrm{Var}[Z_{e_k}] = \sigma_k^2$. Hence, the noise depends on the choice of the input – the model is *heteroscedastic*.

Consider a sequential allocation and estimation strategy $\mathcal{A}$ to estimate θ. Assume that after seeing n samples $\mathcal{A}$ returns $\hat{\theta}_n = (\hat{\theta}_{n1}, \ldots, \hat{\theta}_{nK})^T$ as the estimate of θ. Consider the worst-case prediction loss:

$$L(\mathcal{A}) = \max_{x \in \mathcal{X}} \mathbb{E}\left[(\theta^T x - \hat{\theta}_n^T x)^2\right].$$

Clearly, by the choice of $\mathcal{X}$,

$$L(\mathcal{A}) = \max_{1 \le k \le K} \mathbb{E}\left[(\hat{\theta}_{nk} - \theta_k)^2\right].$$

Hence, the criterion to minimize this loss is equivalent to minimizing the loss considered above (equation 3.1). However, unlike in the homoscedastic case where although the variance of the response influences the loss, it does not influence the best allocation, there is no optimal passive (non-sequential) allocation strategy in this case.

Note that the loss function defined here is slightly different than the loss function considered in Section 2.2.2 for the G-optimality criterion, where we used

$$L' = \max_{x \in \mathcal{X}} \mathbb{E}\left[(Y_x - \hat{\theta}_n^T x)^2\right],$$

with $Y_x = \theta^T x + Z_x$ (Z_x uses the notation of (3.2)). In particular, note that

$$L' = \max_{x \in \mathcal{X}} \left(\mathbb{E}\left[(\theta^T x - \hat{\theta}_n^T x)^2\right] + \sigma_x^2\right),$$

where σ_x^2 is the variance of Z_x. Note that in the homoscedastic case ($\sigma_x^2 \equiv \text{const}$) the two loss functions are the same, while in the heteroscedastic case they are different. Note that the optimal allocations (assuming the knowledge of the variances) also differ for the two loss functions.

It would be interesting to further investigate the loss L' and the problem of finding good sequential algorithms aimed at minimizing this loss when $\mathcal{X}$ is restricted to the unit vectors. Another interesting open question is what are the good sequential allocation algorithms for L (or L) in the general case when $\mathcal{X}$ can be arbitrary (finite, or non-finite) set. However, these problems are out of the scope of this thesis.

3.5 Summary

This chapter examines the problem of estimating the mean response of heterogeneous groups to equal precision. This kind of problem arises in quality assurance and clinical trials. Two

Active Learning algorithms GFSP-MAX and GAFS-MAX for solving this problem. GFSP-MAX is a phase-based batch algorithm while the GAFS-MAX algorithm is incremental. Both algorithm converge to the optimal allocation. In addition Antos et al. (2008) shows that the rate of convergence of GAFS-MAX algorithm is $\tilde{O}(n^{-3/2})$.

Chapter 4

Active Learning for Stratified Sampling

The previous chapter presented two Active Learning algorithms for solving the multi-armed bandits problem. In this chapter we study a similar problem of estimating the mean of a stratified function. Stratified sampling is used to measure the mean value of a function that can be divided into regions (strata). We assume that a stratified function with full knowledge of the size of each stratum is provided. In order to compute the mean of such a function the Learner sample each stratum. The goal is to minimize the number of samples required to achieve a certain approximation error of the estimate of the mean. A Passive Learner will sample according to the proportion of each stratum and will not take advantage of the variance of a stratum. An Active Learner, on the other hand, can take advantage of the different variance in each stratum and decide the sampling proportion of the stratum in order to reduce the estimation error. Formulation and motivation of the problem is presented next followed by the two Active Learning algorithms.

4.1 Stratified Sampling

In Machine Learning and statistics one is frequently faced with the problem of estimating the expected value of a quantity or a function. However, the task becomes more challenging when one is not given the function and can only sample the function in desired locations. For example if one wants to do a poll to find out which Canadian political party is going to be win the popular vote in the federal election on 14 October, 2008. He/she does not have enough resources to ask every eligible Canadian of their voting preference, but can randomly sample some Canadians and ask for their voting preferences. The result of this sampling will provide an estimate of the true result. The accuracy of the estimate depends on the number of samples and how well the sample represents the entire population. The goal is to improve the estimate with a limited set of samples. Therefore, one needs to concentrate on picking those samples that well represent the population. If one has some additional

knowledge about the different population groups, for example geography, culture, ethnicity, religion, age etc., then one can use this knowledge to improve their estimate. One can identify people in a stratum based on knowledge available to them and then sample each stratum to calculate the estimated value for the entire population. For example in Canada we have geographic strata based on provinces and territories. In addition, french-speaking Canadians mostly reside in Quebec than any other province or territory of Canada. This information can provide us more ways of stratifying the population. We can even further divide the population on the basis of economy and urban and rural areas. This technique of sampling strata of a function to estimate the mean of an entire function is known as stratified sampling. Stratified sampling is an important variance reduction technique in Monte-Carlo estimation (Fishman, 1999). In the following section we formalize this problem.

4.1.1 Problem Formulation

Given a function f, we are interested in calculating the expected value $\mu = \mathbb{E}[f(X)]$ with n samples, where X is a random variable taking values in the domain of f. Let $\mathcal{D}$ denote the distribution of X and let $Y = f(X)$, the result of applying function f to the random variable X. Assume that the domain $\mathcal{X}$ of the function f is divided up into K disjoint strata, $(\mathcal{X}_k)$ (i.e., $\mathcal{X}_k \cap \mathcal{X}_j = \emptyset$ and $\cup_{1 \le k \le K} \mathcal{X}_k = \mathcal{X}$. Let $p_k = \mathbb{P}(X \in \mathcal{X}_k)$ be the probability mass of stratum k. We assume that p_k is known and that we can sample from $\mathcal{D}$ restricted to $\mathcal{X}_k$ $(k = 1, \ldots, K)$ in an efficient manner. Let us denote these distributions by $\mathcal{D}_k$ (hence, the support of $\mathcal{D}_k$ is $\mathcal{X}_k$).

Clearly,

$$\begin{aligned} \mathbb{E}[Y] &= \sum_{k=1}^{K} \mathbb{E}[Y|X \in \mathcal{X}_k] \, \mathbb{P}(X \in \mathcal{X}_k) \\ &= \sum_{k=1}^{K} p_k \, \mathbb{E}[Y|X \in \mathcal{X}_k], \end{aligned}$$

so we can estimate μ if we estimate each $\mu_k = \mathbb{E}[Y|X \in \mathcal{X}_k]$ and then take the weighted linear combination of these estimates with (p_k) used as weights.

Assume that we estimate μ_k by taking T_{kn} independent samples, $X_{kt} \sim \mathcal{D}_k$, and then computing the sample mean of f at these samples:

$$\hat{\mu}_{kn} = \frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} Y_{kt},$$

where Y_{kt} is the sample obtained from function f for sample X_k at time t. In line of what was said above we can then form

$$\hat{\mu}_n = \sum_{k=1}^{K} p_k \hat{\mu}_{kn}.$$

Our objective is to allocate samples such that the expected loss L_n between our estimated values, $\hat{\mu}_n$, and the true expected value, μ, is minimized

$$\text{minimize} \quad L_n = \mathbb{E}[(\hat{\mu}_n - \mu)^2].$$

In the sequential (active) problem the algorithm chooses which strata to allocate the next sample to based on the values already seen. Alternatively, we may consider the following non-sequential problem:

$$\begin{aligned} \text{minimize} \quad & L_n = \mathbb{E}[(\hat{\mu}_n - \mu)^2] \\ \text{w.r.t} \qquad & (T_{kn})_{1 \le k \le K} \qquad \text{s.t.} \sum_{k=1}^{K} T_{kn} = n, \end{aligned}$$

assuming that the distributions $\mathcal{D}_k$ are known.

Since the above algorithms estimate μ_k with the sample mean of the data originating from strata k, which is unbiased and since the sample means of different strata are independent of each other, the loss can be written as

$$L_n = \sum_{k=1}^{K} p_k^2 L_{kn}, \tag{4.1}$$

where $L_{kn} = \mathbb{E}\left[(\hat{\mu}_{kn} - \mu_k)^2\right]$ is the loss of estimating μ_k. In particular, if T_{kn} samples are allocated to stratum k , where T_{kn} is a deterministic number then

$$L_{kn} = \frac{\sigma_k^2}{T_{kn}}$$

and thus

$$L_n = \sum_{k=1}^{K} \frac{p_k^2 \sigma_k^2}{T_{kn}},$$

where $\sigma_k^2 = \text{Var}\left[Y_{kt}\right]$ is the variance associated with strata k.

4.2 Passive Approach

Similar to the multi-armed bandit problem, the passive solution (without any prior knowledge except for the strata proportions) is to allocate samples in proportion to p_k, which is referred to as *stratified random sampling with proportional allocation* in the literature. This solution is never worse than the solution obtained through sampling over the original function f with same number of samples n. The loss of this algorithm after n samples is:

$$\begin{aligned} L_n(\mathcal{P}) &= \sum_{k=1}^{K} \frac{p_k^2 \sigma_k^2}{T_{kn}} \\ &= \sum_{k=1}^{K} \frac{p_k^2 \sigma_k^2}{n\, p_k} \quad (\text{since } T_{kn} = n\, p_k) \\ &= \frac{1}{n} \sum_{k=1}^{K} p_k \sigma_k^2. \end{aligned}$$

On the other hand, if $X_1, \ldots, X_n \sim \mathcal{D}$ are independent, then

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{1}{n}\sum_{t=1}^{n} Y_t - \mu\right)^2\right] &= \mathbb{E}\left[\left(\frac{1}{n}\sum_{t=1}^{n} [Y_t - \mu]\right)^2\right] \\
&= \frac{1}{n}\,\mathbb{E}\left[(Y-\mu)^2\right] \\
&= \frac{1}{n}\sum_{k=1}^{K} p_k \mathbb{E}\left[(Y-\mu)^2 | X \in \mathcal{X}_k\right] \\
&= \frac{1}{n}\sum_{k=1}^{K} p_k \left(\mathbb{E}\left[(Y-\mu_k)^2 | X \in \mathcal{X}_k\right] + (\mu_k - \mu)^2\right) \\
&= \frac{1}{n}\sum_{k=1}^{K} p_k \left(\sigma_k^2 + (\mu_k - \mu)^2\right) \\
&\geq \frac{1}{n}\sum_{k=1}^{K} p_k \sigma_k^2,
\end{aligned}$$

showing that proportional stratified sampling can be better and is never worse than the "pure" Monte-Carlo approach.

The excess loss suffered by proportional sampling is $\Theta(1/n)$ compared with the optimal allocation when the optimal allocation is non-proportional.

4.3 Active Sample Allocation for Stratified Sampling

In order to construct an active solution for this problem, first consider the non-sequential version of the stratified sampling problem. We assume, for a moment, that we have full knowledge of the distribution of each stratum. We also assume that $\sigma_k > 0$ holds for all k, as the case of $\sigma_k = 0$ is uninteresting. Then the optimal allocation for stratum k would be:

$$T_{kn}^* = n\frac{p_k\sigma_k}{\sum_{j=1}^{K} p_j\sigma_j} = n\frac{p_k\sigma_k}{\overline{\sigma}} = n\lambda_k.$$

Here $\overline{\sigma} = \sum_{j=1}^{K} p_j\sigma_j$ and $\lambda_k = \frac{p_k\sigma_k}{\overline{\sigma}}$.

Analogous to the multi-armed bandit problem one can estimate the variances of a stratum and use those estimates allocate samples. We, again, look at two active algorithms for the stratified sampling problem. The first algorithm, is a phase-based batch algorithm presented by Etore and Jourdain (2007). The second algorithm is an incremental algorithm and it is again a joint work with András Antos and Csaba Szepesvári (Antos et al., 2008).

4.3.1 Adaptive Optimal Allocation

Etore and Jourdain (2007) present an adaptive algorithm to allocate samples in a stratified function. Their algorithm is a batch algorithm that works in phases with the size of phases chosen by some schedule. At the beginning of each phase, their algorithm estimates the

Adaptive Optimal Allocation

Input: probability mass p_k for each stratum and $\xi(i)$ - a function returning length of phase i
Initialization: Sample each stratum $\xi(1)/K$ times.
for $i \geq 1$

1. calculate empirical standard deviation for each stratum k:

$$\hat{\sigma}_{kn} = \sqrt{\frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} Y_{kt}^2 - \left(\frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} Y_{kt} \right)^2}$$

2. sample each stratum k once
3. let $M_i = \xi(i) - K = \sum_{k=1}^{K} m_{ki}$
4. compute

$$m_{ki} = \begin{cases} M_i/K, & \text{if } \sum_{j=1}^{K} p_j \hat{\sigma}_{jn} = \emptyset \\ \frac{p_k \hat{\sigma}_{kn}}{\sum_{j=1}^{K} p_j \hat{\sigma}_{jn}} M_i, & \text{otherwise,} \end{cases}$$

   for each stratum k

5. Sample stratum k, m_{ki} times

Table 4.1: A phase-based algorithm for efficiently solving the stratified sampling problem. The algorithm computes the estimate of standard deviation of each stratum, k, at the beginning of each phase and then uses those estimates to allocate samples.

variance for each stratum and then uses those estimates to allocate samples in each stratum. In order to get a close estimate of the variance of a stratum, the algorithm needs to sample each stratum enough times. Therefore, Etore and Jourdain (2007) force the algorithm to sample each stratum at least once in each phase. This requirement adds a constraint on the function specifying the phase length: the length of each phase must be at least equal to the number of strata. Initially the variance of each stratum is assumed to be equal and hence the samples for the first phase are equally divided into each stratum.

The Algorithm

In the first step this algorithm computes the estimated sample standard deviation for each stratum. Next it force samples each stratum once. Then in step three and four the algorithm computes the number of samples for each stratum from the current phase. This allocation number is calculated using the estimate of the standard deviation computed in the first step. The algorithm is presented in Table 4.1.

Properties

Etore and Jourdain (2007) show that their estimate of the standard deviation, $\hat{\sigma}_{kn}$, converges to the true standard devation, σ_k and their algorithm converges to optimal allocation. This algorithm requires a function that provides the length for each phase. The only constraint on this function is that the length of each phase must at least equal to the number of strata. However, since this function controls the exploration of this algorithm, it should be chosen carefully. In some situations one might not know the number of samples available to them in advance. For example a worker conducting a survey on the phone requires a varying amount of time to complete each survey. Therefore, the worker does not know how many people he/she will be able to sample in a given time interval. In such situations the phase-based algorithm is not applicable. In the next section we present an active incremental algorithm, which does not face the above mentioned problem.

4.3.2 Incremental Greedy Allocation

In joint work András Antos and Csaba Szepesvári (Antos et al., 2008), we developed an incremental algorithm for active allocation for stratified sampling problem. Similar to Etore and Jourdain (2007), we also estimate the standard deviation of each stratum. However, the difference is that our algorithm works sequentially. Initially the algorithm samples each stratum to get an estimate of the standard deviation. Then it allocates samples to the stratum that minimizes loss, $\text{argmax}_{1\leq k\leq K} \frac{\hat{\lambda}_{kn}}{T_{kn}}$. In order to make sure that our estimates of the standard deviation converges to the optimal standard deviation, the algorithm samples each stratum indefinitely in the limit. To achieve indefinite sampling for each stratum, the algorithm employs the same forcing technique as in the case of GAFS-MAX algorithm, i.e., it force samples a stratum if the number of samples, T_{kn}, is less than $c\sqrt{n}$ with some constant $c > 0$. We refer to this algorithm as GAFS-WL (greedy allocation with forced selection for weighted Loss).

The Algorithm

The algorithm is presented in Table 4.2. The algorithm computes the variance of each stratum in the first step. In the second step it computes the proportion of standard deviation for each stratum. In the following steps, the algorithm chooses the stratum to sample next. If a stratum is severely under-sampled, the algorithm samples the most under-sampled stratum. Otherwise, it samples the stratum with the highest proportion of the $\hat{\lambda}_{kn}$ and T_{kn}. That is the algorithm samples the stratum with the least number of samples allocated to it compared to its proportion of standard deviation.

Algorithm GAFS-WL

Input: c - a constant > 0 to control the amount of exploration
Initialization: In the first K trials choose each stratum once
Set $T_{k,K} = 1$ for each k $(1 \leq k \leq K)$, $n = K$
At time $n + 1$ do:

1. Compute $\hat{\sigma}^2_{kn} = \frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} Y^2_{kt} - \left(\frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} Y_{kt} \right)^2$
2. Let $\hat{\lambda}_{kn} = p_k \hat{\sigma}_{kn} / (\sum_{j=1}^{K} p_j \hat{\sigma}_{jn})$ if $\sum_{j=1}^{K} p_j \hat{\sigma}_{jn} \neq 0$, otherwise let $\hat{\lambda}_{kn} = 1/K$.
3. Let $U_n = \text{argmin}_{1 \leq k \leq K} T_{kn}$
4. Let
$$I_{n+1} = \begin{cases} U_n, & \text{if } T_{U_n,n} < \sqrt{n} + 1 \\ \text{argmax}_{1 \leq k \leq K} \frac{\hat{\lambda}_{kn}}{T_{kn}}, & \text{otherwise,} \end{cases}$$
where in the second case ties are broken in an arbitrary, but systematic manner.
5. Choose stratum I_{n+1}, let $T_{k,n+1} = T_{k,n} + \mathbb{I}_{\{I_{n+1}=k\}}$
6. Observe the feedback $Y_{I_{n+1},T_{I_{n+1},n+1}}$.

Table 4.2: GAFS-WL algorithm estimates the variance of each stratum at each time step and then uses those estimate to allocate a new sample. If there is an under-sampled stratum, the algorithm samples that stratum instead.

Properties

The properties of this algorithm are similar to the GAFS-MAX algorithm (in the previous chapter). The parameter c controls the exploration and should be set between $(0, \sqrt{n}/K)$. Lower values, generally, provide a good tradeoff between exploration and exploitation. GAFS-WL algorithm also converges to the optimal allocation and suffers excess loss at a rate of $\tilde{O}(1/n^{3/2})$.GAFS-WL algorithm never performs worse than the passive algorithm, even when the optimal allocation is proportional allocation. The experimental results for our algorithm demonstrate huge improvements compared to the passive algorithm.

4.3.3 GAFS-WL Rate of Convergence

The main result for the GAFS-WL is the following: There exists some integer N such that for any k, $n \geq N$,

$$L_n \leq L_n^* + \tilde{O}(n^{-3/2}).$$

The proof is identical to the proof for the GAFS-MAX algorithm presented in the appendix. Besides the changes to some constants, the bounds do not change. A sketch for the GAFS-MAX algorithm is provided in section 3.3.3. The proof for the GAFS-WL algorithm changes in two places: Lemma 2 and Theorem 3.

Lemma 2, for the GAFS-MAX algorithm, shows a bound on $\hat{\lambda}$, where:

$$\hat{\lambda} = \frac{\hat{\sigma}_k^2}{\sum_{j=1}^K \hat{\sigma}_j^2}.$$

This bound needs to be modified for the definition of $\hat{\lambda}$ for the GAFS-WL algorithm:

$$\hat{\lambda} = \frac{p_k \hat{\sigma}_k}{\sum_{j=1}^K p_j \hat{\sigma}_j}.$$

Except for some changes to the constants, this bounds sticks to the same form.

The second change comes in the very last part of Theorem 3. The loss for the GAFS-MAX algorithm takes the maximum loss among all arms. Theorem 3 for the GAFS-MAX algorithm shows that the excess loss decays at the desired rate ($\tilde{O}(n^{-3/2})$) for each arm. This result can be easily extended to the case where the loss is the sum of losses of all strata.

4.4 Relation to Optimal Experiment Design

As in Section 3.2 we can view the problem as finding a sequential allocation strategy to minimize some loss function in the linear regression model (3.2) with heteroscedastic errors and when the domain of the regression problem is restricted to the unit vectors.

Here, the loss function that makes the two problems equivalent is

$$L(\mathcal{A}) = \mathbb{E}\left[\left\{\mathbb{E}(\hat{\theta}_n^T X|\hat{\theta}) - \mathbb{E}(\theta^T X)\right\}^2\right],$$

where $X \sim p$ is chosen independently of $\hat{\theta}$ from a probability distribution p over $\mathcal{X}$. Indeed, by the choice of $\mathcal{X}$ and if we define $p(e_k) = p_k$ then

$$L(\mathcal{A}) = \mathbb{E}\left[(\sum_{k=1}^{K} p_k \hat{\theta}_{nk} - \theta_k)^2\right],$$

showing the equivalence of the two problems.

Another way of making a connection to optimal experiment design is to study weighted prediction losses. Let

$$L_{p^2}(\mathcal{A}) \quad = \quad \mathbb{E}\left[\sum_{x \in \mathcal{X}} p^2(x)(\hat{\theta}_n^T x - \theta^T x)^2\right].$$

With this loss again we get a generalization of the problem studied in this chapter to linear regression problems. The unusual feature of this loss is that the weight factors are squared. When they are not squared, the loss becomes

$$\begin{aligned} L_p(\mathcal{A}) &= \mathbb{E}\left[(\hat{\theta}_n^T X - \theta^T X)^2\right] \\ &= \mathbb{E}\left[\left\|\hat{\theta} - \theta\right\|_{C,2}^2\right], \end{aligned}$$

where $X \sim \mathcal{D}$ and $C = \mathbb{E}\left[XX^T\right]$ and $\|v\|_{C,2}^2 = v^T C v$ is the weighted 2-norm of vector v, where it is assumed that C is a positive definite matrix. Remember that under A-optimality the goal was to design a strategy such that

$$\mathbb{E}\left[\left\|\hat{\theta} - \theta\right\|_2^2\right]$$

is minimized. Thus, this criterion can be viewed as a variant of the A-optimality criterion when a weighted 2-norm is used instead of the unweighted 2-norm (this criterion is actually called C-optimality in the literature). It remains for future work to see if the algorithmic ideas and the results presented in this chapter extend to these cases.

4.5 Summary

In this chapter we examined the problem of estimating the mean response of a stratified function. Stratified functions naturally occur in the environment. For example many survey problems use this technique to sample appropriately. Two Active Learning algorithms, Adaptive Optimal Allocation and GAFS-WL, are presented for solving this problem. Adaptive Optimal Allocation is a phase-based batch algorithm while the GAFS-WL algorithm is incremental. Both algorithm converge to the optimal allocation. In addition, based on the results in Antos et al. (2008) we argued that the rate of convergence of the excess loss of the GAFS-WL algorithm is $\tilde{O}(n^{-3/2})$.

Chapter 5

Experimental Results

This chapter presents an evaluation of active algorithms' performance on three real-world scenarios. We compared the performance of the active algorithms with a Passive Learning algorithm and show that in all cases the Active learner algorithms performed significantly better. In the first problem we modeled clinical trials problem, where the goal is to measure the mean response of each test group equally well. The GAFS-MAX algorithm was applied to solve this problem. The results showed that the algorithm is quite efficient and the rescaled excess loss suffered by this algorithm stayed bounded by a small constant. The second and third problems dealt with estimating the mean of a stratified function. The second problem dealt with estimating the Consumer Price Index (CPI), which is an instance of a survey sampling problem. We obtained data from Statistics Canada for the 2008 Canadian CPI and used that as a basis for our model. Again, the active algorithm provided improvements over the passive algorithm. The last problem dealt with pricing an Asian option. Glasserman et al. (1999) present a setting where one can use stratified sampling to estimate the price of an option. We replaced their passive sampling algorithm with our active algorithm for stratified sampling. The performance increased by more than 100 fold in some cases.

Before proceeding with the experiments, one should understand the effects of the exploration parameter c. In the following section we analyze the effect of the c parameter on the performance of our algorithm.

5.1 Effect of Exploration Parameter

The GAFS (MAX and WL) algorithm requires a parameter c that controls the amount of exploration. We wanted to find out how sensitive our algorithm is to the value of c. Setting c close to 0 will make the algorithm greedy with respect to the estimated loss, while setting it equal to $\sqrt{n}/K$ will make the algorithm close to the passive algorithm. A greedy approach can hugely under/over estimate the variance of an option and thus it can experience a large loss. We evaluated the performance of the GAFS-MAX algorithm on a two arm bandit

problem. We chose the Bernoulli distribution for each arm with a variety of mean values. The algorithm was run with c ranging from close to 0 to $\sqrt{n}/K$. We collected the data for the excess loss suffered by the algorithm after $n = 10,000$. The problems were sorted in decreasing order of $\lambda = \frac{\sigma_1^2}{\sigma_2^2}$. The variance proportion, λ, for each problem is given in Table 5.1. The results are presented in the Figure 5.1. This figure shows that the greedy approach can severely hurt the performance of the GAFS-MAX algorithm, which is what one expects. As c approached 50, the algorithm essentially behaved like a passive learner as it sampled each arm uniformly.

Problem#	λ	Problem#	λ
1	0.96	8	0.61
2	0.89	9	0.53
3	0.87	10	0.51
4	0.84	11	0.48
5	0.78	12	0.43
6	0.75	13	0.38
7	0.68	14	0.36

Table 5.1: Variance proportion, λ, for problem 1 thru 14

As depicted in Figure 5.1 the loss was relatively high when c is close to 50 compared with the loss when c was say 20. In addition the loss increased as the problem number increased. Since the problems were sorted in decreasing order based on the proportion of the variance, the optimal allocation proportion for the first arm also decreased. Thus, when the forced exploration rate (c) was set at a high value and the difference λ decreased the loss increased, as the algorithm sampled the first arm more often than it should have. Therefore, with higher c values one has the risk of exploring too much and thus incurring a higher loss.

Since our algorithm is an incremental algorithm, we were also interested in the performance of the algorithm as a function of n for different values of the c parameter. Figure 5.2 presents the sum of rescaled excess loss for $n = 1, 2, \cdots, 10,000$. Again, when c was close to 0 the loss was enormous and as c approached 50 the loss increased as well. In the middle of this range, larger value of c had larger sum of excess loss. This is due to the fact that larger c value caused the algorithm to behave similar to a passive algorithm initially and thus the algorithm incurred a higher loss in early part of the experiment. Since this graph shows the sum of excess loss, the extra loss suffered early on shows up in the graph.

These two experiments demonstrate that lower values of c provide a good balance between exploration and exploitations. Values close to 0 can lead to huge loses and thus should be avoided. Values close to $\sqrt{n}/K$ will essentially make the algorithm similar to the Passive Learning algorithm and hence take away all the advantage of Active Learning.

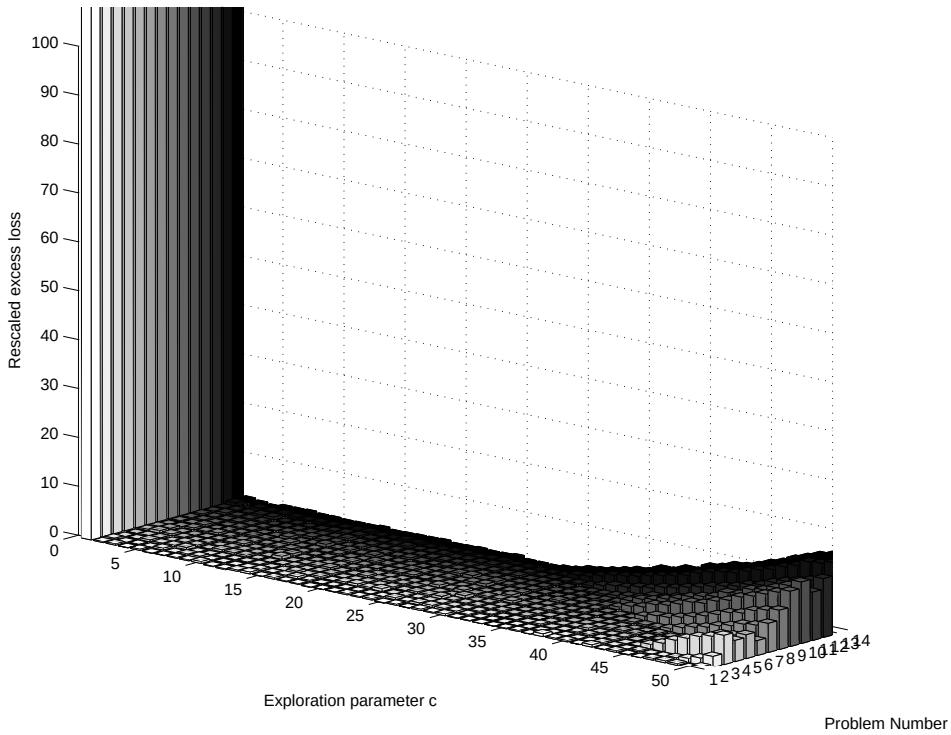

Figure 5.1: Rescaled excess loss for 2-arm bandit problems with the Bernoulli distribution. The algorithm is robust to the selection of c. However, if exploration is really limited then the algorithm's performance degrades.

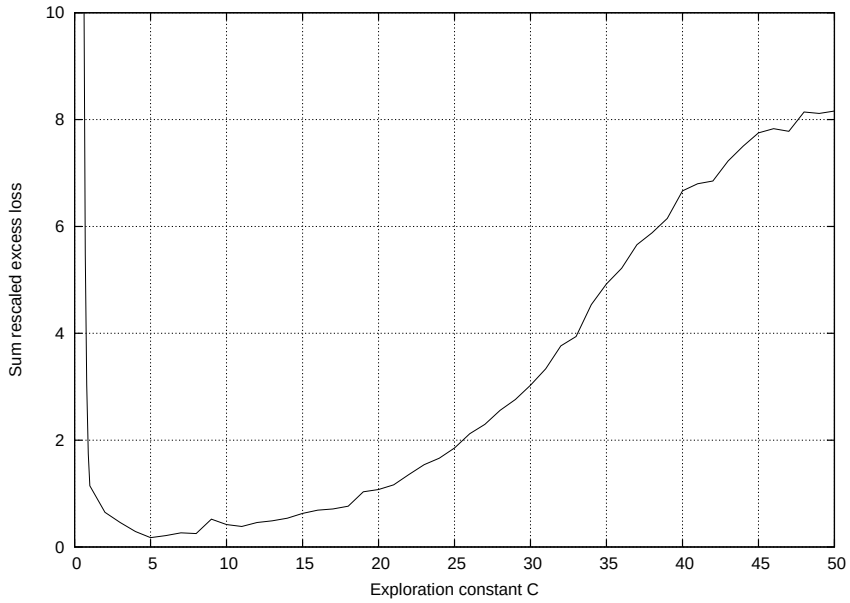

Figure 5.2: Sum of rescaled excess loss for 2-arm bandit problem with the Bernoulli distribution.

5.2 Experimental Results for Multi-Armed Bandits

In this section we empirically compare the Active Learning and Passive Learning algorithms on a 5-arm bandit problem with Bernoulli and truncated Normal distributions. This problem can be viewed as an abstraction for a problem of clinical trials where you have multiple groups of people, for example groups based on age or gender, and one is interested in accessing the expected affect of a drug for a typical member of a group. The response vary among any group due to distinct condition of patients in each group, for example their weight, diet, health condition, genetics etc. Under our representation the 5 bandit arms represent the groups of people and the response of a drug is modeled with the Bernoulli and truncated Normal distributions.

5.2.1 Experimental Setup

In the first experiment we setup a 5-arm bandit problem with the random responses modeled as Bernoulli random variables; in the second experiment we used truncated normal distribution. The means and corresponding variances for each arm is given in Table 5.2. We chose these values so there was a range of variances from small to large. The metric used for comparison is the rescaled excess loss, $n^{3/2}(L_n - L_n^*)$. Since we assume no prior knowledge about the distribution of the arms, the optimal passive strategy for minimizing this loss is to sample each arm uniformly. Each experiment was run for 100,000 runs and the averaged results are reported. We also computed the variance between the different runs of the experiment and present it as error bars in the graphs. There is no effort spent on optimizing the exploration parameter, rather a small value, $c = 1$ was chosen. We use this value for all our experiments. The ξ function was $\xi(i) = \xi(i-1) + 1$ with $\xi(0) = 5$ (number of arms) for the GFSP-MAX algorithm, which was optimized for this problem.

	Bernoulli Distribution		Truncated Normal Distribution	
Arm#	Mean	Variance	Mean	Variance
1	1/2	1/4	0.0	1/2
2	1/2	1/4	0.0	1/2
3	1/3	2/9	0.0	1/3
4	2/9	14/81	0.0	2/9
5	4/27	92/729	0.0	4/27

Table 5.2: Mean and Variance of Bandit Arms

Results for the Bernoulli Distribution

Figure 5.3 presents the average rescaled excess loss with error bars for the 5-arm bandit problem with the Bernoulli distribution. We noticed that the GAFS-MAX algorithm incurs

a huge loss before returning to small constant bounded loss. The GFSP-MAX algorithm does not depict this characteristic. We investigated this problem and found that this increase in loss was due to the under-sampling of arm 5. Arm 5 had a low mean value and thus an even smaller variance. If the variance of an arm is underestimated, say the variance is estimated to be zero, then only $\sqrt{n}$ samples are allocated to that arm, instead of $\lambda_k n$. When $\hat{\lambda}_k < 1/\sqrt{n}$, where $\hat{\lambda}_k$ is an estimate of λ_k, the algorithm will not allocate extra samples. This is a problem when in fact $\lambda_k n > \sqrt{n}$ (i.e. $\lambda_k > \sqrt{n}$).

Rather than optimize the exploration parameter c, which requires some prior knowledge of the problem, we added a bonus term to the next-arm-selection process for picking the arm with the most potential of minimizing the loss. This bonus term is based on the empirical Bernstein bounds (see (Audibert et al., 2007)). It reflects the algorithm's confidence in its estimate. If the algorithm is very confident, which happens when it has sampled many times, then its confidence is high and the bonus is low. On the other hand, if the algorithm has low confidence because it has only sampled this arm a few times then the bonus term is large. Therefore, arms that are under-sampled will receive a higher bonus. The confidence bound also depends on the range of the variance. We replaced the upper-bound of this range with the maximum estimated variance the algorithm has seen so far as arbitrary large ranges can hurt the performance of this algorithm. For this problem, the substitution of range with maximum estimated variance has shown some advantage. The GAFS-MAX algorithm with the added bonus term is presented in Table 5.3.

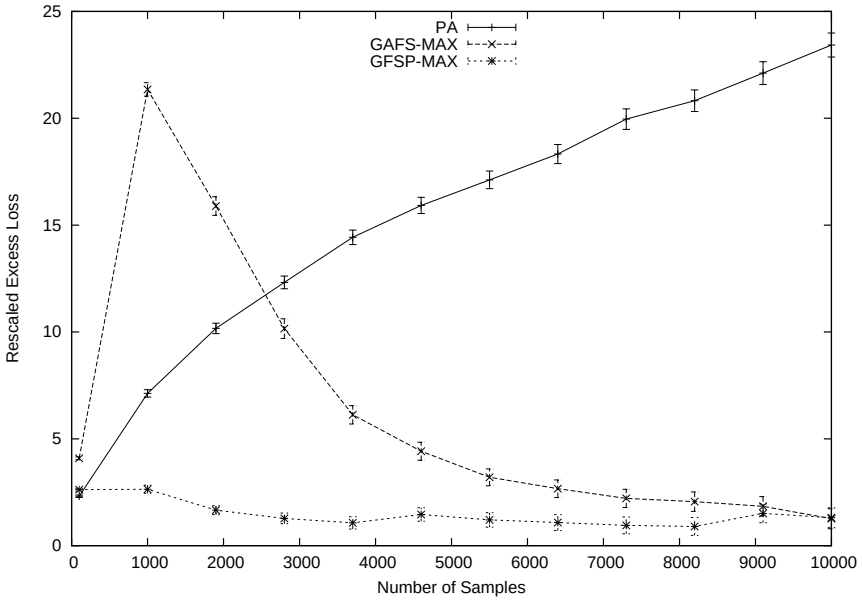

Figure 5.3: Rescaled excess loss for the Bernoulli distribution. The GAFS-MAX algorithm incurred a large loss early on. We investigated this problem and find that one of the bandit arms was severely under-sampled.

Algorithm GAFS-MAX with Bonus

Input: c - a constant > 0 to control the amount of exploration
Initialization: In the first K trials choose each arm once
Set $T_{k,K} = 1$ for each k $(1 \le k \le K)$, $n = K$
At time $n+1$ do:

1. Compute $\hat{\sigma}^2_{kn} = \frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} X^2_{kt} - \left(\frac{1}{T_{kn}} \sum_{t=1}^{T_{kn}} X_{kt} \right)^2$
2. Let $\hat{\lambda}_{kn} = \hat{\sigma}^2_{kn} / (\sum_{j=1}^{K} \hat{\sigma}^2_{jn})$ if $\sum_{j=1}^{K} \hat{\sigma}^2_{jn} \neq 0$,
   otherwise let $\hat{\lambda}_{kn} = 1/K$.
3. Let $U_n = \mathrm{argmin}_{1 \le k \le K} T_{kn}$
4. Let

$$I_{n+1} = \begin{cases} U_n, & \text{if } T_{U_n,n} < \sqrt{n} + 1 \\ \mathrm{argmax}_{1 \le k \le K} \frac{\hat{\lambda}_{kn}}{T_{kn}} + \hat{\sigma}^2_{kn} \sqrt{\frac{2 \log(4K3n)}{T_{kn}}} \; + & \\ \qquad \hat{\sigma}^2_{max,n} \frac{\log(4K3n)}{T_{kn}} \;, & \text{otherwise,} \end{cases}$$

   where in the second case ties are broken in an arbitrary, but systematic manner; $\hat{\sigma}^2_{max,n} = \max_{1 \le k \le K} \hat{\sigma}^2_{kn}$.
5. Choose arm I_{n+1}, let $T_{k,n+1} = T_{k,n} + \mathbb{I}_{\{I_{n+1}=k\}}$
6. Observe the feedback $X_{I_{n+1}, T_{I_{n+1},n+1}}$.

Table 5.3: GAFS-MAX algorithm with an added bonus term. This bonus term is based on the empirical Bernstein bounds.

Figure 5.4 shows a specific "bad" case for the GAFS-MAX algorithm. However, with this bonus term this case is not so daunting anymore. We also computed the rescaled excess loss for the bonus term added algorithm. The results are presented in Figure 5.5. The figure shows the rescaled excess losses of the active algorithms stayed bounded, as predicted by the theory, while the rescaled loss of the passive sampling strategy grew at a rate of $\sqrt{n}$. It is remarkable that the limit of the rescaled loss seems to be a small number, showing the efficiency of the algorithm.

Results for Truncated Normal Distribution

Figure 5.6 presents the average rescaled excess loss with error bars for the 5-arm bandit problem with truncated Normal distribution. This time there was no bump in the graph as was seen for the Bernoulli distribution. We conjecture that this bump could be a special case of the Bernoulli distribution, since the probability of underestimating is the largest amongst bounded random variables. Further work needs to be done to investigate the conditions

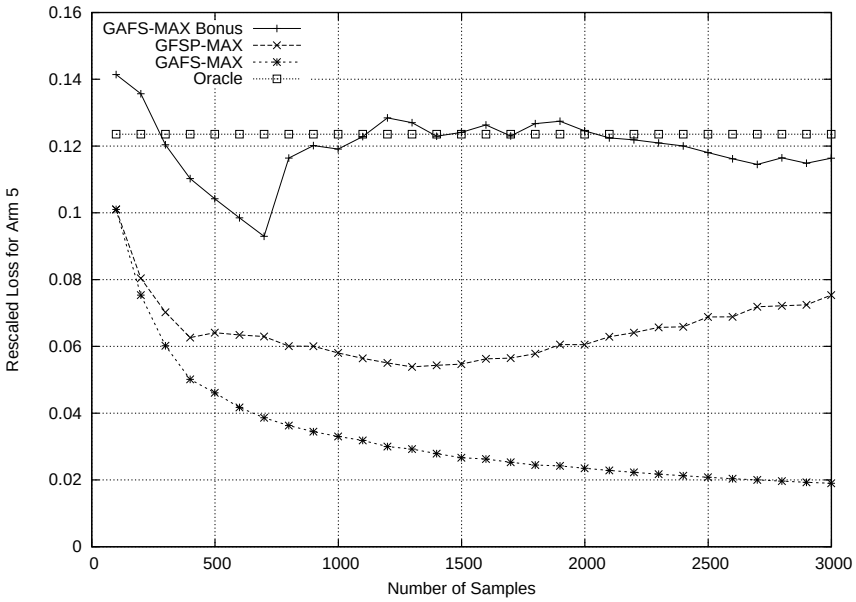

Figure 5.4: Allocation proportion for arm 5 for the Bernoulli distribution. The GAFS-MAX algorithm's allocation proportion was much lower than the true allocation proportion. This effected the loss suffered by the algorithm. Hence there is a large bump in Figure 5.3.

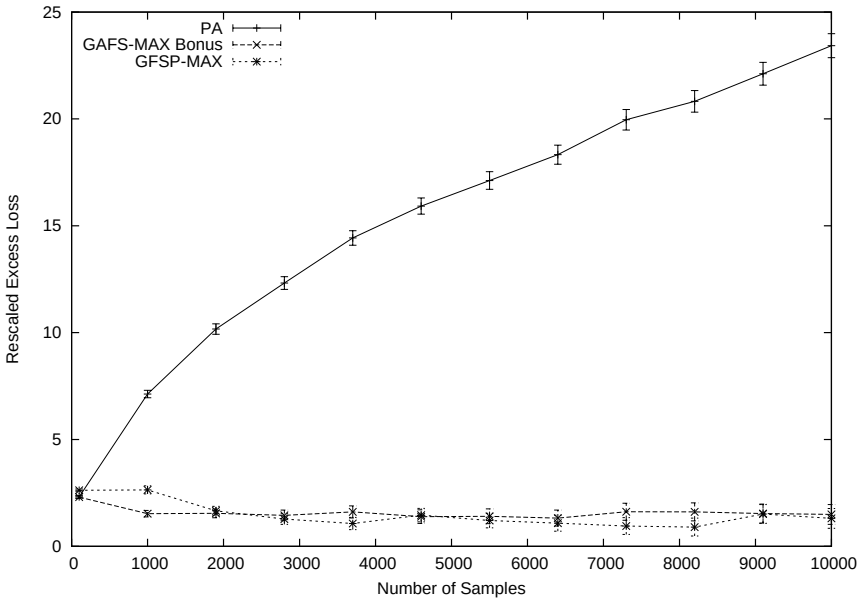

Figure 5.5: Rescaled Excess Loss for the Bernoulli distribution. The added bonus fixed the problem of under-sampling of an arm.

under which we see this surge in the loss. Similar to the previous result, the rescaled excess loss for the active algorithms stayed bounded while the passive algorithm's loss grew with the number of samples.

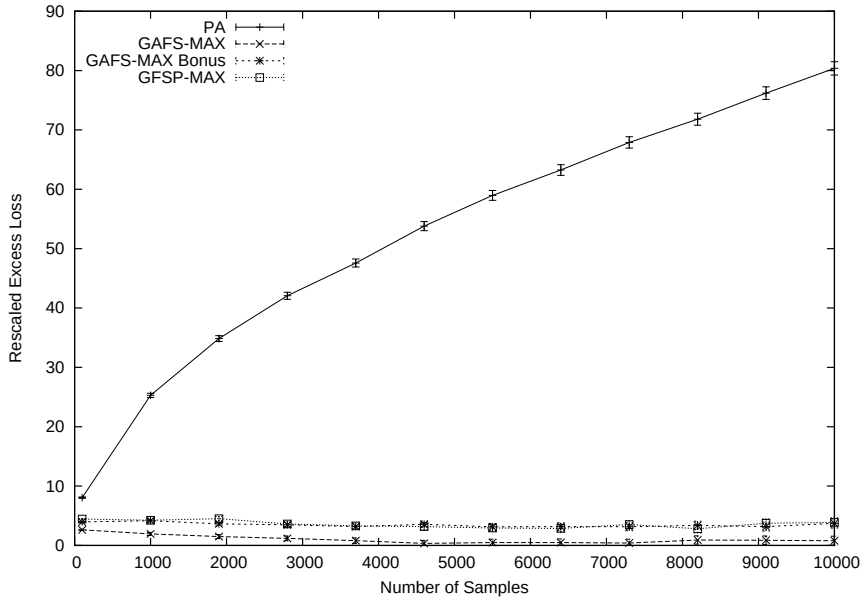

Figure 5.6: Rescaled Excess Loss for the truncated Normal distribution. Active Learning algorithms performed better than the passive algorithm. This time there was no bump in the graph as was seen for the Bernoulli distribution. We suspect that the surge in loss is a special case for the Bernoulli distribution.

These two experiments demonstrate the efficiency and the advantage of using the Active Learning algorithm for two different distributions. The algorithms' excess rescaled loss stayed bounded by a small constant, further detailing the efficiency of the algorithms.

5.3 Experimental Results for Stratified Sampling

Similar to the multi-armed Bandit case, we compared the performance of the two Active Learning algorithms against a Passive Learning algorithm for the stratified sampling problem domain. We test the performance on two problems. The first problem is an instance of a survey sampling. A survey is conducted on a small proportion of the entire population to identify or analyze general trends about the entire population. We chose a particular problem of estimating the Canadian Consumer Price Index (CPI) for the year 2008. This problem is a simplification of the CPI survey conducted by Statistics Canada. The second problem dealt with pricing an Asian option in the Black-Scholes model based on the details provided by Glasserman et al. (1999). We compared the performance of Adaptive Optimal Allocation (AOA) and the GAFS-WL algorithm with a passive algorithm. The passive

algorithm was the stratified random sampling with proportional allocation (PA). In both of these examples the Active Learning algorithms perform better than the passive algorithm.

5.3.1 Estimating CPI

The Consumer Price Index (CPI) measures the rate of price change for goods and services bought by consumers and "is the most widely used indicator of price changes in Canada" (Canada, 1996). The CPI is measured by observing the price change for a basket of goods which consists of a small number of goods and services. Statistics Canada estimates price for about 600 items by using approximately 60,000 samples every year. These 600 items are grouped into 8 major components (Table 5.4). Statistics Canada provides average CPI for each of the 8 major components each year for general public use. This problem is a natural stratified sampling problem with 600 strata. Since we only had the data for the 8 major components, we used the major components as our strata as an illustration of the application of our algorithm. We modeled the price change of each component as a truncated normal distribution. Statistics Canada provided only the averages for each component of the CPI; therefore, we guessed the variance of each component based on personal experience. The data provided by Statistics Canada and our guess of the variance is given in Table 5.4.

The unadjusted CPI for 2008 is 115.7% relative to the 2002 prices.			
Major Components	Proportion	Avg. value (base year 2002)	Variance
Food	17.04%	117.1%	0.7
Shelter	26.62%	123.1%	0.5
Household Renovation	11.1%	105.6%	0.3
Clothing & Footwear	5.36%	96.1%	0.5
Transportation	19.88 %	122.4%	0.7
Health & Personal care	4.73%	109.4%	0.1
Rec, Education & Reading	12.2%	103.9%	0.2
Alcohol & Tobacco	3.07%	128.0%	0.2
Total	100%	115.7%	
Source http://www.statcan.ca/english/Subjects/Cpi/cpi.pdf			

Table 5.4: 2008 Canadian Consumer Price Index

The goal was to estimate the CPI by sampling each of the 8 components modeled as a truncated normal distribution with mean provided from Statistics Canada and our guess of the variance. We took in total $5,000$ samples and ran each experiment for $100,000$ times to collect the average absolute error. The exploration parameter c for the GAFS-WL algorithm was set to $c = 1$ and the ξ function was $\xi(i) = \xi(i-1) + 5$ with $\xi(0) = 8$(number of strata) for the AOA algorithm (again we optimized the ξ function but not the c value).

Results

Figure 5.7 presents the average absolute error with error bars for the estimate of the change in the CPI for GAFS-WL, AOA and PA algorithms. The absolute error is the difference between the estimated mean, $\hat{\mu}$, and the true mean, μ, obtained from Table 5.4.

$$absolute\ error = |\hat{\mu} - \mu|$$

The two Active Learning algorithm GAFS-WL and AOA performed better than the Passive Learning algorithm PA. The active algorithms needed about 200 less samples to achieve the same level of accuracy as a passive learner. These gains are not huge but if one looks at Figure 5.8, which presents the allocation proportion for the three algorithm under consideration here and the optimal allocation, the active algorithms allocation proportion was similar to the optimal allocation proportion while the passive algorithm was far from optimal allocation. The optimal allocation proportion, λ^*_{kn}, is calculated as follows:

$$\lambda^*_{kn} = \frac{p_k \sigma_k}{\sum_{j=1}^{K} p_j \sigma_j}.$$

The allocation proportion for the two active learning algorithms use the estimated standard deviation

$$\lambda_{kn} = \frac{p_k \hat{\sigma}_k}{\sum_{j=1}^{K} p_j \hat{\sigma}_j}.$$

The Passive Learning algorithm pays no attention to the variance in any strata and only uses the proportion, p_k, for allocation. Although the active algorithms were allocating samples close to the optimal allocation, the performance gain was not immense. This was possibly due to the fact that we were dealing with a limited number of strata with weights for these strata biased by Statistics Canada based on past fluctuations in price. Nevertheless, this example shows the application of our algorithm to a vast area of survey sampling and the algorithm achieved strict improvement over the passive algorithm.

5.3.2 Pricing an Asian Option

Glasserman et al. (1999) presented a variance reduction technique for Monte Carlo simulations applied to the pricing of an Asian option. Their method combined importance sampling based on a change of drift with stratified sampling along a small number of key dimensions. They used proportional allocation for the stratified sampling, i.e., their approach used a passive algorithm. We replace their passive algorithm and used an active algorithm for this problem to price the option and compared the variance reduction achieved by active algorithms. The results show that in some cases the active algorithms were able to get about 100 times improvements.

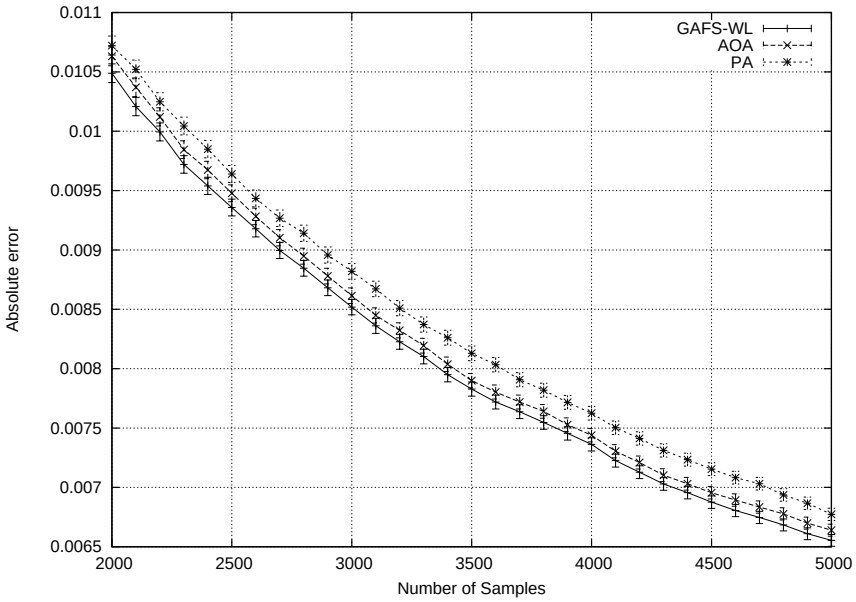

Figure 5.7: Absolute Error for the estimate of the 2008 Canadian CPI. Active algorithms performed strictly better than the passive algorithm. The performance gain was not immense, perhaps due to the limited data available to us.

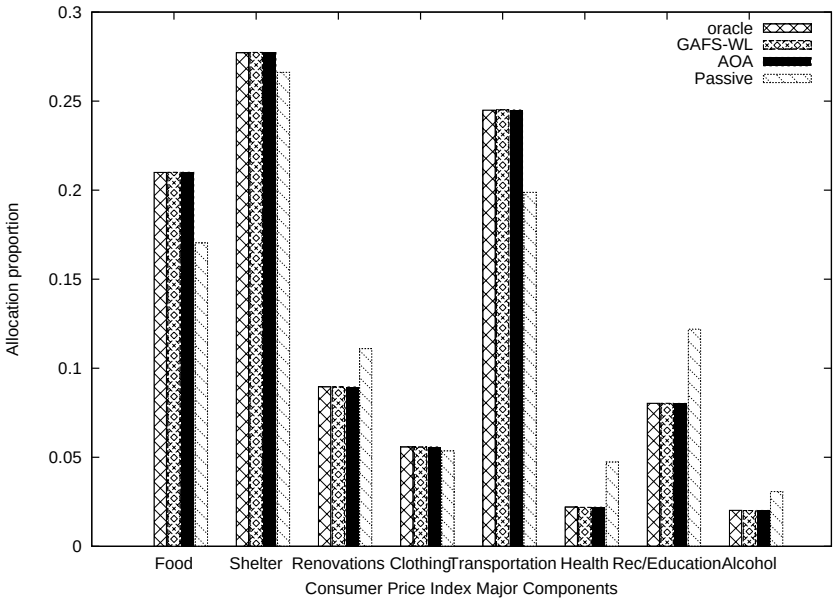

Figure 5.8: Allocation proportion to estimate the 2008 Canadian CPI. The allocation proportion of active algorithms matches optimal allocation proportions.

Setup

The price of an option at time t for a particular asset, under the Black-Scholes model can be described by the following stochastic differential equation

$$dS_t = \nu S_t dW_t + rS_t dt,$$

where r is the risk free, continuously compounded interest rate, ν is the asset's volatility, W_t is a standard Wiener process and S_0 fixed. The solution can be simulated exactly for a discrete set of time points $0 = t_0 < t_1 < \cdots < t_d = T$ by computing:

$$S_{t_i} = S_{t_i} \exp\left([r - 1/2\nu^2](t_i - t_{i-1}) + \nu\sqrt{(}t_i - t_{i-1})X_i\right), i = 1, \cdots, d,$$

where $X_i, \cdots, X_d$ are $\mathcal{N}(0, 1)$ and independent of each other. The discounted payoff of the arithmetic Asian call option with strike K is given by:

$$G(X_1, \cdots, X_d) = \exp(-rT)\left(\frac{1}{d}\sum_{i=1}^{d} S_{t_i} - K\right)^+, \tag{5.1}$$

and similarly the put option payoff is given by:

$$G(X_1, \cdots, X_d) = \exp(-rT)\left(\frac{1}{d}\sum_{i=1}^{d} K - S_{t_i}\right)^+. \tag{5.2}$$

We are interested in computing the expectation of this payoff:

$$a = \mathbb{E}[G(X)].$$

The payoff of a typical option is nonnegative hence the integrand in the above equations (Equation (5.1) and (5.2)) only considers values greater than 0. We can rewrite this equation as follows:

$$a = \mathbb{E}[\exp(f(X))\mathbf{1}_D(X)]$$

where $f(X) = \log G(X)$ on D and $D = \{X \in \mathbb{R}^d : G(X) > 0\}$. A naïve Monte Carlo simulation would average over random replications of $G(X)$. Glasserman et al. (1999) instead, used important sampling guided by a change of drift to identify key dimensions. With importance sampling the expectation of the payoff is as follows:

$$a = \mathbb{E}[f_\mu(X)]$$

where $f_\mu(X) = G(X + \mu)\exp(-\mu^T X - 1/2\mu^T\mu)\mathbf{1}_D(X + \mu)$, μ is the drift vector $\mu = \text{argmax}_{X\in D}(G(X) - 1/2X^T X)$. Glasserman et al. (1999) provide details on calculating the drift vector. Then they stratified $f_\mu(X)$ into 100 equally probable strata and used proportional stratified sampling to estimate the price, a, of the option. They used $1,000,000$

samples to calculate the price of the option and reported a significant reduction in the variance of the estimate.

We were interested in using our Active Learning algorithm in place of the Passive Learning algorithm, proportional allocation, and compare the performance. We used the same setting as Glasserman et al. (1999), i.e. we also sampled $1,000,000$ times with 100 equal probable strata. Following Etore and Jourdain (2007) we set $S_0 = 50, \nu = 0.1, r = 0.05$ and $T = 1.0$ for all our experiments. The exploration parameter c for the GAFS-WL algorithm was set to $c = 1$ and the ξ function was $\xi(0) = 100,000, \xi(1) = 400,000, \xi(2) = 500,000$. for the AOA algorithm, which is the same function used by Etore and Jourdain (2007). We computed the option price by repeating the experiment $1,000$ times and then took the average.

Results

The results for the call option pricing problem are presented in Table 5.5 and the results for the put option pricing problem are presented in Table 5.6[1]

d	K	Price	Variance AOA	Variance GAFS-WL	ratio GHS/GAFS-WL
16	45	\$ 6.06	2.83×10^{-9}	2.64×10^{-9}	3.31
	50	\$ 1.92	1.15×10^{-9}	9.88×10^{-10}	1.97
	55	\$ 0.20	1.27×10^{-8}	1.19×10^{-8}	60.94
64	45	\$ 6.00	2.32×10^{-9}	1.99×10^{-9}	4.35
	50	\$ 1.85	6.73×10^{-10}	5.49×10^{-9}	1.99
	55	\$ 0.17	9.04×10^{-9}	8.60×10^{-9}	70.01

Table 5.5: Call option Price and Variance

d	K	Price	Variance AOA	Variance GAFS-WL	ratio GHS/GAFS-WL
16	45	\$ 0.01	7.93×10^{-10}	7.96×10^{-10}	89.15
	50	\$ 0.63	6.94×10^{-8}	6.52×10^{-8}	59.69
	55	\$ 3.67	1.41×10^{-9}	1.35×10^{-9}	2.00
64	45	\$ 0.01	5.56×10^{-10}	5.50×10^{-10}	105.19
	50	\$ 0.62	5.61×10^{-8}	5.54×10^{-8}	66.17
	55	\$ 3.70	1.39×10^{-9}	1.14×10^{-9}	2.54

Table 5.6: Put option Price and Variance

Etore and Jourdain (2007) reported, and was confirmed by our experiments, that the reason their algorithm achieved such great performance gains in some cases was due to the fact that in those cases the standard deviation of almost all strata was close to zero. The

[1]NOTE: our results do not match the results published in Etore and Jourdain (2007) due to a programming bug in their implementation.

Passive Learning algorithm allocates equal number of samples for strata with zero and non-zero standard deviation, whereas the Active Learning algorithm allocates most samples to strata with non-zero standard deviation and a few samples to the strata with zero standard deviation to ensure that the standard deviation is really zero. For example from Table 5.5 and 5.6 we see that the put option performance increase is much higher than the call option performance increase for the case where $K = 45$ and $d = 64$. Figure 5.9 shows that for the put option most of the strata had standard deviation equal to zero, whereas for the call option every strata had some standard deviation. Therefore the performance gain was not as considerable in the call option case as it was for the put option case.

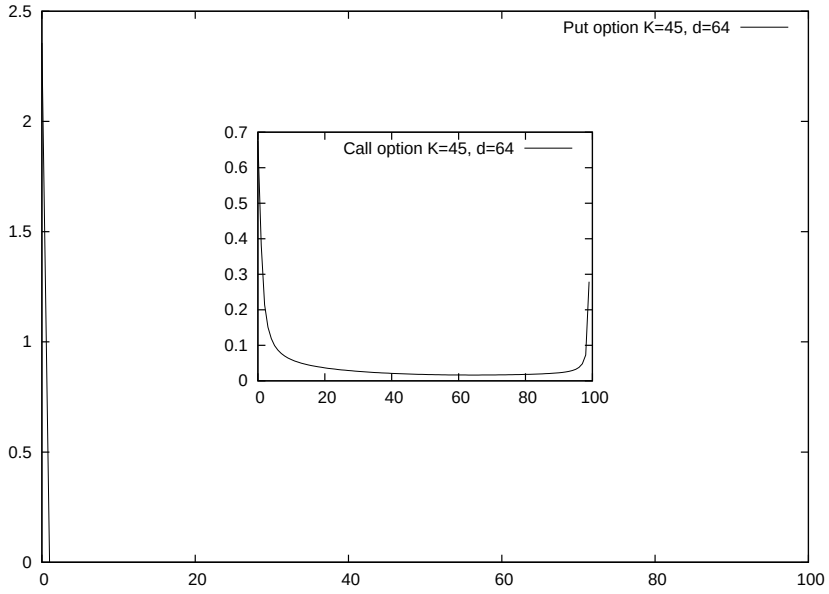

Figure 5.9: Estimated standard deviation of stratum 1 to 100 for put and call option. Most of the strata (99 out of 100), had zero standard deviation for the put option. The call option, on the hand, had some standard deviation in all of its strata. For the put option, the active algorithms can focus on just 1 stratum and reduce the variance of the option price significantly.

5.4 Incremental versus Phase-Based

These experiments demonstrate that our incremental algorithm performance is close to the phase-based batch algorithm. In some case the incremental algorithm performed slightly better than phase-based. The main distinction and advantage of our algorithm is that it is completely sequential. The phase based algorithm allocates samples for the entire phase. The performance of this algorithm is only optimal at the end of the phase. During the phase the samples are non-optimally allocated and thus the algorithm suffers non-optimal

loss. Figure 5.10 illustrates this behavior of the phase-based algorithm. This figure shows the rescaled excess loss suffered by both the GAFS-MAX algorithm and the GFSP algorithm for the 5-Arm bandit problem modeled with truncated Normal random variables (see section 5.2.1). The incremental algorithms, GAFS-MAX and GAFS-MAX Bonus, attain a constant loss for any new sample. However, the phase based algorithm, GFSP-MAX, has a zic-zac loss pattern. The loss is near optimal near the boundaries of the phase and non-optimal during the rest of the phase indicating non-optimal allocation during the phase.

In addition to this non-optimal behavior during the phase, the phase-based algorithm requires at least K samples before the algorithm can be executed. Our sequential algorithm, on the other hand, does not have to wait for a certain minimum number of samples before the algorithm can be applied. Therefore, the GAFS algorithm can be thought of as an improvement over the phase-based GFSP algorithm.

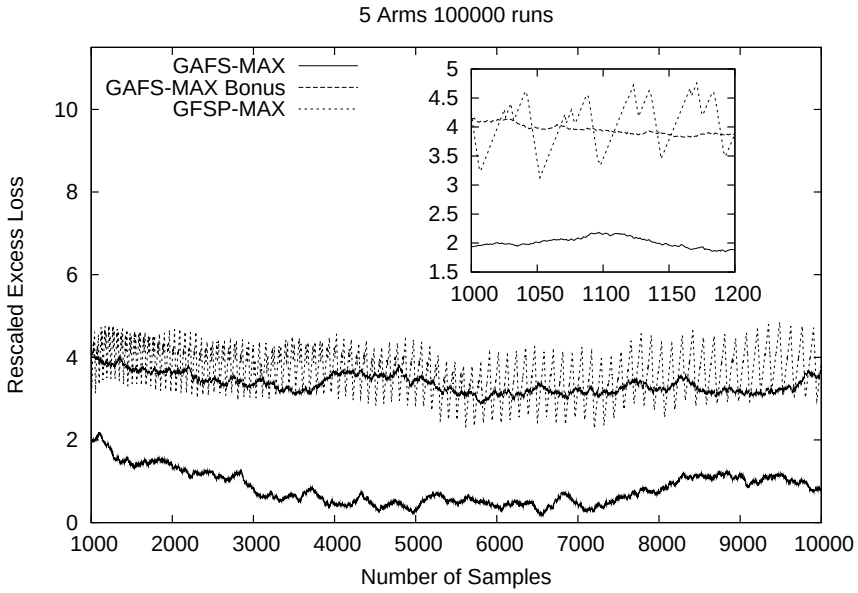

Figure 5.10: Rescaled excess loss for the 5-Arm Truncated Normal random variable problem. The GFSP algorithm is a phase based algorithm while the GAFS algorithm is incremental. The loss suffered by the phase based has a zic-zac pattern. The loss increases during the phase and then is minimized when the phase finishes. The incremental algorithm has a constant loss, on the other hand.

5.5 Summary

In this chapter we compared the performance of active and passive algorithm on three different examples motivated from the real-world problems. The first example was based on the clinical trials problem, which we modeled as a multi-armed bandit problem. The second

example was an instance of survey sampling problem where we focused on estimating the Canadian CPI for the year 2008. The last example was pricing as Asian option under the Black-Scholes model. The results showed that the active algorithms outperforms the passive algorithm in all instances. A passive learning algorithm always allocated equal amount of samples to each arm/stratum, whereas active algorithms' allocation strategy was very close to the optimal strategy. Therefore, active algorithms outperformed the passive algorithm. The performance of our sequential algorithm matches up with the phase-based batch algorithm presented by Etore and Jourdain (2007). One advantage of our algorithm is that it does not compromise performance and it does not require a batch of certain minimum sample size. However, in some cases phase-based batch algorithms may also make sense as they could be faster or when incremental approaches are (physically) not possible. Clearly, it is possible to combine the two algorithms and create algorithms that are half-batch, half-incremental.

Chapter 6

Conclusions

Active Learning uses past observed data and a pool of input samples to reduce the sample complexity of a problem. Getting a value for an input sample is costly and generally there is an abundance of input samples available. Therefore, Active Learning is applicable in a variety of settings. This thesis provides a survey of Active Learning algorithms and a novel Active Learning algorithm for each the multi-armed bandit problem and stratified sampling problem is presented.

The current Active Learning literature is extensive. Most techniques aim to reduce the uncertainty of the problem either directly or indirectly. In this thesis we examined a few of these techniques. The sample complexity of Active Learning has been found to be strictly greater than Passive Learning for a number of problems. In other cases, the minimax bounds for Active Learning are never worse than Passive Learning. Balcan et al. (2008) have shown that Active Learning has strict advantage over Passive Learning asymptotically when individual performance is considered. Their algorithm is, however, impractical due to its arbitrarily poor initial performance. The research continues on in the hunt for a more practical algorithm.

This thesis also presents two Active Learning algorithm for multi-armed bandit problem and for a related problem of stratified sampling. Both algorithms are sequential and use the samples to learn the corresponding distributions and allocate samples accordingly. The algorithms performed strictly better than a Passive Learning algorithm on three real-world based problems. Theoretically both of these active algorithms suffer an excess loss that decreases at a rate of $\tilde{O}(n^{-3/2})$.

Appendix A

Proof of Bound on Excess Loss for the GAFS-MAX Algorithm

This chapter presents a proof showing that loss L_n suffered by the GAFS-MAX algorithm after n samples is $\leq L_n^* + \tilde{O}(n^{-3/2})$, where L_n^* is the loss for an algorithm that knows the variance of each random variable after n iterations. The proof is developed through a series of Lemmata. First, we state Hoeffding's inequality in a form that suits the best our needs:

Lemma 1 (Hoeffding's inequality) *Let Z_t be a sequence of zero-mean, i.i.d. random variables, where $a \leq Z_t \leq b$, $a < b$ reals. Then, for any $0 < \delta \leq 1$,*

$$\mathbb{P}\left(\frac{1}{n}\sum_{t=1}^{n} Z_t > \sqrt{\frac{1}{2}\frac{(b-a)^2}{n}\log(1/\delta)}\right) \leq \delta.$$

Let

$$\Delta(R, n, \delta) = R\sqrt{\frac{\log(1/\delta)}{2n}}.$$

Let $\mu_k^{(2)} = \mathbb{E}\left[X_{kt}^2\right]$, R_k be the size of the range of the random variables $\{X_{kt}\}_t$ (i.e., $|\mathrm{supp}(X_{kt})| \leq R_k$), S_k be the size of the range of the random variables $\{X_{kt}^2\}_t$, and B_k be the essential supremum of the random variables $\{|X_{kt}|\}_t$. Note that $R_k \leq 2B_k$ and $S_k \leq B_k^2$. Let

$$\begin{aligned} A_\delta \quad = \quad & \left(\bigcap_{1\leq k\leq K, n\geq 1}\left\{\left|\frac{1}{n}\sum_{t=1}^{n} X_{kt}^2 - \mu_k^{(2)}\right| \leq \Delta(S_k, n, \delta_n)\right\}\right)\bigcap \\ & \bigcap_{1\leq k\leq K, n\geq 1}\left\{\left|\frac{1}{n}\sum_{t=1}^{n} X_{kt} - \mu_k\right| \leq \Delta(R_k, n, \delta_n)\right\}, \end{aligned}$$

where $\delta_n = \delta/(4K(n(n+1))$. Note that δ_n is chosen such that $\sum_{k=1}^{K}\sum_{n=1}^{\infty}\delta_n = \delta/4$. Hence, we observe that by Hoeffding's inequality

$$\mathbb{P}\left(A_\delta\right) \geq 1 - \delta.$$

The sets $\{A_\delta\}$ will play a key role in the proof: Many of the statements will be proved on these set.

Our first result connects a lower bound on T_{kn} to the rate of convergence of $\hat{\lambda}_{kn}$. Let $a_k = |\mu_k| + B_k$, $b_k = S_k + a_k R_k (\leq 5B_k^2)$, $a'_k = \sigma_k^4/(4b_k^2)$, and $\ell_{K,\delta} = \log(4K/\delta)$. Note that

$$\log(\delta_n^{-1}) = \log(n(n+1)) + \ell_{K,\delta} \leq 2\log n + 1 + \ell_{K,\delta}. \tag{A.1}$$

Lemma 2 *Fix $0 < \delta \leq 1$ and $n_0 > 0$, and assume that for $n \geq n_0$, $1 \leq k \leq K$, $T_{kn} \geq f_n \geq 2$ holds on A_δ, and that for $n \geq n_0$, for each $1 \leq k \leq K$ such that $\sigma_k \neq 0$*

$$f_n \geq \frac{1}{2a'_k}\left(2\log f_n + 1 + \ell_{K,\delta}\right). \tag{A.2}$$

Then there exists a constant $c > 0$ such that for any $n \geq n_0$, $1 \leq k \leq K$, on A_δ

$$\left|\hat{\lambda}_{kn} - \lambda_k\right| \leq c\sqrt{\frac{\log(\delta_n^{-1})}{f_n}} \tag{A.3}$$

holds. In particular, $c = \sqrt{2}(\max_{1\leq k\leq K} b_k + \lambda_k \sum_{j=1}^K b_j)/\Sigma^2 \leq 5\sqrt{2}(\max_{1\leq k\leq K} B_k^2 + \sum_{j=1}^K B_j^2)/\Sigma^2$.

Remark 1 *If $f_n = \beta n^p$ (p,$n > 0$) then* (A.2) *can be written as*

$$\log n \leq \frac{\beta a'_k}{p} n^p - \frac{1 + \ell_{K,\delta} + 2\log\beta}{2p}. \tag{A.4}$$

Proof First, we develop a bound on $|\hat{\sigma}_{kn}^2 - \sigma_k^2|$. Let $\hat{\mu}_{kn}^{(2)} = 1/T_{kn}\sum_{t=1}^{T_{kn}} X_{kt}^2$ and $\hat{\mu}_{kn} = 1/T_{kn}\sum_{t=1}^{T_{kn}} X_{kt}$. Consider any element of A_δ. Then by the definition of A_δ, $|1/m\sum_{t=1}^m X_{kt}^2 - \mu_k^{(2)}| \leq \Delta(S_k, m, \delta_m)$ holds simultaneously for any $m \geq 1$. Hence, for $n \geq n_0$ it also holds that

$$\left|\frac{1}{T_{kn}}\sum_{t=1}^{T_{kn}} X_{kt}^2 - \mu_k^{(2)}\right| \quad \leq \quad \Delta(S_k, T_{kn}, \delta_{T_{kn}}) \leq \Delta(S_k, f_n, \delta_{f_n}),$$

where we have used that $\log(x(x+1)/\delta)/x$ is monotonically decreasing when $x \geq 2$ and $T_{kn} \geq f_n \geq 2$. Similarly, we get that

$$\left|\frac{1}{T_{kn}}\sum_{t=1}^{T_{kn}} X_{kt} - \mu_k\right| \quad \leq \quad \Delta(R_k, f_n, \delta_{f_n}).$$

Using $\hat{\sigma}_{kn}^2 = \hat{\mu}_{kn}^{(2)} - \hat{\mu}_{kn}^2$ and $\sigma_k^2 = \mathbb{E}\left[X_{kt}^2\right] - (\mathbb{E}\left[X_{kt}\right])^2 = \mu_k^{(2)} - \mu_k^2$, we get

$$\begin{aligned}\left|\hat{\sigma}_{kn}^2 - \sigma_k^2\right| &\leq \left|\hat{\mu}_{kn}^{(2)} - \mu_k^{(2)}\right| + \left|\hat{\mu}_{kn}^2 - \mu_k^2\right| \\ &\leq \Delta(S_k, f_n, \delta_{f_n}) + \Delta(R_k, f_n, \delta_{f_n})(|\mu_k| + B_k),\end{aligned} \tag{A.5}$$

where we used $|x^2 - y^2| \leq |x - y|\,(|x| + |y|)$.

Denote the right-hand side of (A.5) by $\Delta_k(n, \delta)$. Now, let us develop a lower bound on

$\hat{\lambda}_{kn}$ in terms of λ_k. Then, for $n \geq n_0$,

$$\begin{aligned}
\hat{\lambda}_{kn} &= \frac{\hat{\sigma}^2_{kn}}{\sum_{j=1}^K \hat{\sigma}^2_{jn}} \geq \frac{\sigma_k^2 - \Delta_k(n,\delta)}{\Sigma^2 + \sum_{j=1}^K \Delta_j(n,\delta)} \\
&= \frac{\sigma_k^2}{\Sigma^2}\left(1 + \frac{\sum_{j=1}^K \Delta_j(n,\delta)}{\Sigma^2}\right)^{-1} - \frac{\Delta_k(n,\delta)}{\Sigma^2 + \sum_{j=1}^K \Delta_j(n,\delta)} \\
&\geq \lambda_k\left(1 - \frac{\sum_{j=1}^K \Delta_j(n,\delta)}{\Sigma^2}\right) - \frac{\Delta_k(n,\delta)}{\Sigma^2},
\end{aligned}$$

where we used $1/(1+x) \geq 1-x$ that holds for $x > -1$.

An upper bound can be obtained analogously: For $n \geq n_0$, if

$$\Sigma^2 \geq 2\sum_{j=1}^K \Delta_j(n,\delta) \tag{A.6}$$

then

$$\begin{aligned}
\hat{\lambda}_{kn} &= \frac{\hat{\sigma}^2_{kn}}{\sum_{j=1}^K \hat{\sigma}^2_{jn}} \leq \frac{\sigma_k^2 + \Delta_k(n,\delta)}{\Sigma^2 - \sum_{j=1}^K \Delta_j(n,\delta)} \\
&= \frac{\sigma_k^2}{\Sigma^2}\left(1 - \frac{\sum_{j=1}^K \Delta_j(n,\delta)}{\Sigma^2}\right)^{-1} + \frac{\Delta_k(n,\delta)}{\Sigma^2 - \sum_{j=1}^K \Delta_j(n,\delta)} \\
&\leq \lambda_k\left(1 + 2\frac{\sum_{j=1}^K \Delta_j(n,\delta)}{\Sigma^2}\right) + 2\frac{\Delta_k(n,\delta)}{\Sigma^2},
\end{aligned}$$

where we used $1/(1-x) = 1 + x/(1-x) \leq 1+2x$ that holds for $0 \leq x \leq 1/2$. This constraint follows from (A.6), that is implied if n is big enough so that

$$2\Delta_j(n,\delta) \leq \sigma_j^2, \quad 1 \leq j \leq K. \tag{A.7}$$

The upper and lower bounds above together give

$$|\hat{\lambda}_{kn} - \lambda_k| \leq \frac{2}{\Sigma^2}\left(\lambda_k \sum_{j=1}^K \Delta_j(n,\delta) + \Delta_k(n,\delta)\right).$$

Noting that $\Delta_j(n,\delta)$ equals to

$$(S_j + R_j(|\mu_j| + B_j))\sqrt{\frac{\log(\delta_{f_n}^{-1})}{2f_n}} = b_j\sqrt{\frac{\log(\delta_{f_n}^{-1})}{2f_n}}, \tag{A.8}$$

where $b_j = S_j + R_j(|\mu_j| + B_j)$, we get

$$|\hat{\lambda}_{kn} - \lambda_k| \leq \frac{\sqrt{2}}{\Sigma^2}\left(\lambda_k \sum_{j=1}^K b_j + b_k\right)\sqrt{\frac{\log(\delta_{f_n}^{-1})}{f_n}}.$$

Since $f_n \leq T_{kn} \leq n$, $\delta_{f_n}^{-1}$ can be upper bounded by δ_n^{-1} leading to (A.3).

At last, to satisfy (A.7), by (A.8), it suffices if $\sigma_j^4 f_n \geq 2b_j^2 \log(\delta_{f_n}^{-1})$, $1 \leq j \leq K$. Note that if $\sigma_j = 0$ then $R_j = S_j = 0$, and so $b_j = 0$ and both sides above are 0. Otherwise we need

$$f_n \geq \frac{2b_j^2}{\sigma_j^4}\log(\delta_{f_n}^{-1}) = \frac{1}{2a_j'}\log(\delta_{f_n}^{-1})$$

that is guaranteed by (A.1) and (A.2) provided that $n \geq n_0$. ∎

Now we show how a rate of convergence result for $\hat{\lambda}_{kn}$ can be turned into bounds on $T_{kn}/n - \lambda_k$. Note that this lemma holds pointwise, i.e., for any element ω of the probability space Ω underlying the random variables considered. For brevity, we write below $\hat{\lambda}_{kn}$ instead of $\hat{\lambda}_{kn}(\omega)$, T_{kn} instead of $T_{kn}(\omega)$, etc.

Let $\lambda_{\min} = \min_{1\le j\le K} \lambda_j$. In what follows, unless otherwise stated, we will assume that $\lambda_{\min} > 0$. or $K = 1$ the results are obvious, so without the loss of generality (w.l.o.g.) we can also assume that $K \ge 2$.

Lemma 3 *Fix $n_0 > 0$. Assume that g_n is such that for $n \ge n_0$, ng_n is monotone increasing, $5ng_n \ge \lceil \sqrt{n} \rceil$, and*

$$g_n \le \lambda_{\min}/2, \tag{A.9}$$

$$|\hat{\lambda}_{kn} - \lambda_k| \le g_n, \qquad 1 \le k \le K \tag{A.10}$$

hold. Let $\rho = 1 + 2/\lambda_{\min}$. Then the following inequalities hold for $n \ge 1$ and $1 \le k \le K$:

$$-(K-1)\max\left(\frac{n_0}{n}, \frac{1}{n} + \rho g_n\right) \le \frac{T_{kn}}{n} - \lambda_k \le \max\left(\frac{n_0}{n}, \frac{1}{n} + \rho g_n\right).$$

Proof By definition $T_{k,n+1} = T_{kn} + \mathbb{I}_{\{I_{n+1}=k\}}$. Let $E_{kn} = T_{kn} - n\lambda_k$ with $E_{k0} = 0$. Note that $E_{kn} \le n(1-\lambda_k)$ and

$$\sum_{k=1}^{K} E_{kn} = 0 \tag{A.11}$$

holds for any $n \ge 0$. Notice that the desired result can be stated as bounds on E_{kn}. Hence, our goal now is to study E_{kn}. If b_{jn} is an upper bound for E_{jn} $(1 \le j \le K)$ then from (A.11) we get the lower bound $E_{kn} = -\sum_{j\ne k} E_{jn} \ge -\sum_{j\ne k} b_{jn} \ge -(K-1)\max_j b_{jn}$. Hence, we target upper bounds on $\{E_{kn}\}_k$.

Assume now that $n \ge n_0$. Note that (A.9) and (A.10) imply $\lambda_k - \hat{\lambda}_{kn} \le |\hat{\lambda}_{kn} - \lambda_k| \le \lambda_k/2$, and thus $\hat{\lambda}_{kn} \ge \lambda_k/2 > 0$ for each k.

From the definition of E_{kn} and T_{kn} we get

$$E_{k,n+1} = E_{kn} - \lambda_k + \mathbb{I}_{\{I_{n+1}=k\}}.$$

By the definition of the algorithm

$$\mathbb{I}_{\{I_{n+1}=k\}} \le \mathbb{I}_{\{T_{kn}\le\lceil\sqrt{n}\rceil \text{ or } k=\operatorname{argmin}_{1\le j\le K} \frac{T_{jn}}{\hat{\lambda}_{jn}}\}},$$

Assume now that k is an index where $\{\frac{T_{jn}}{\hat{\lambda}_{jn}}\}_j$ takes its minimum, that is,

$$\frac{T_{kn}}{\hat{\lambda}_{kn}} \le \min_j \frac{T_{jn}}{\hat{\lambda}_{jn}}.$$

Using $T_{jn} = E_{jn} + n\lambda_j$ and reordering the terms gives

$$E_{kn} + n\lambda_k \quad \le \quad \hat{\lambda}_{kn} \min_j \frac{E_{jn} + n\lambda_j}{\hat{\lambda}_{jn}} \le \hat{\lambda}_{kn}\left(\min_j \frac{E_{jn}}{\hat{\lambda}_{jn}} + n \max_j \frac{\lambda_j}{\hat{\lambda}_{jn}}\right).$$

By (A.11), there exists an index j such that $E_{jn} \leq 0$. Since $\hat{\lambda}_{jn} > 0$ for any j, it holds that $\min_j \frac{E_{jn}}{\hat{\lambda}_{jn}} \leq 0$. Hence,

$$E_{kn} + n\lambda_k \leq n\hat{\lambda}_{kn} \max_j \frac{\lambda_j}{\hat{\lambda}_{jn}}. \tag{A.12}$$

Using (A.10) and (A.9), we get

$$\frac{\lambda_j}{\hat{\lambda}_{jn}} \leq \frac{\lambda_j}{\lambda_j - g_n} = \frac{1}{1 - g_n/\lambda_j}.$$

This is upper bounded by

$$1 + \frac{2g_n}{\lambda_j}$$

using $1/(1-x) \leq 1 + 2x$ for $0 \leq x \leq 1/2$, where the latest constraint follows from (A.9). Using (A.12), $\hat{\lambda}_{kn} \leq 1$, and (A.10) again,

$$\begin{aligned} E_{kn} &\leq n\hat{\lambda}_{kn} \max_j \frac{\lambda_j}{\hat{\lambda}_{jn}} - n\lambda_k \\ &\leq n(\hat{\lambda}_{kn} - \lambda_k) + \frac{2ng_n}{\lambda_{\min}} \\ &\leq \left(1 + \frac{2}{\lambda_{\min}}\right) ng_n = \rho n g_n. \end{aligned}$$

Denote that the right-hand side by H_n. Hence,

$$\mathbb{I}_{\{I_{n+1}=k\}} \leq \mathbb{I}_{\{T_{kn} \leq \lceil\sqrt{n}\rceil \text{ or } E_{kn} \leq H_n\}}.$$

We show that $T_{kn} \leq \lceil\sqrt{n}\rceil$ implies $E_{kn} \leq H_n$. By the definition of E_{kn}, from $T_{kn} \leq \lceil\sqrt{n}\rceil$ it follows that $E_{kn} = T_{kn} - n\lambda_k \leq \lceil\sqrt{n}\rceil \leq 5ng_n$. The bound $\lambda_{\min} \leq 1/K \leq 1/2$ imply $5ng_n \leq H_n$. Hence, $E_{kn} \leq H_n$ follows. Therefore

$$\mathbb{I}_{\{I_{n+1}=k\}} \leq \mathbb{I}_{\{E_{kn} \leq H_n\}}.$$

Now we need the following technical lemma:

Lemma 4 *Let $0 \leq \lambda \leq 1$. Consider the sequences $E_n, \tilde{E}_n, I_n, \tilde{I}_n$ ($n \geq 1$) where $I_n, \tilde{I}_n \in 0, 1$, $E_{n+1} = E_n + I_n - \lambda$, $\tilde{E}_{n+1} = \tilde{E}_n + \tilde{I}_n - \lambda$, $\tilde{E}_1 = E_1$ and assume that $I_n \leq \tilde{I}_n$ holds whenever $E_n = \tilde{E}_n$. Then $E_n \leq \tilde{E}_n$ holds for $n \geq 1$.*

Proof Consider the difference sequence $P_n = \tilde{E}_n - E_n$. The goal is to show that $P_n \geq 0$ holds for any n. It holds that $P_1 = 0$. Since

$$P_{n+1} - P_n = (\tilde{E}_{n+1} - \tilde{E}_n) - (E_{n+1} - E_n) = \tilde{I}_n - I_n \in \{-1, 0, +1\},$$

P_n is always an integer. Hence, it suffices to show that $P_{n+1} \geq 0$ if $P_n = 0$. However, this holds because if $P_n = 0$ then $I_n \leq \tilde{I}_n$. ∎

Now, returning to the proof of Lemma 3, define $\{\tilde{E}_{kn}\}_{n\geq n_0}$ by

$$\begin{aligned}\tilde{E}_{k,n_0} &= E_{k,n_0},\\ \tilde{E}_{k,n+1} &= \tilde{E}_{kn} - \lambda_k + \mathbb{I}_{\{\tilde{E}_{kn}\leq H_n\}}, \quad n \geq n_0.\end{aligned}$$

The conditions of Lemma 4 are clearly satisfied from index n_0. Consequently $E_{kn} \leq \tilde{E}_{kn}$ holds for any $n \geq n_0$. Further, since H_n is monotone increasing in n, $\tilde{E}_{kn} \leq \max(E_{k,n_0}, 1 + H_n) \leq \max(n_0(1-\lambda_k), 1 + H_n), \qquad n \geq n_0$, and so $E_{kn} \leq \max(n_0(1-\lambda_k), 1 + H_n) \leq \max(n_0, 1 + H_n)$ for $n \geq 0$, finishing the upper-bound. ∎

Corollary 1 *Fix $0 < \delta \leq 1$, $c \geq 1/5$, and $n_0 > 0$. Assume that $f_n > 0$ is such that for $n \geq n_0$, f_n/n^2 is monotone decreasing, and $f_n \leq n$,*

$$f_n \geq \frac{4c^2}{\lambda_{\min}^2}(2\log n + 1 + \ell_{K,\delta}), \tag{A.13}$$

$$|\hat{\lambda}_{kn} - \lambda_k| \leq c\sqrt{\frac{\log(\delta_n^{-1})}{f_n}}, \qquad 1 \leq k \leq K \tag{A.14}$$

hold. Let

$$H_n(\delta) = c\left(1 + \frac{2}{\lambda_{\min}}\right) n\sqrt{\frac{\log(\delta_n^{-1})}{f_n}}.$$

Then the following inequalities hold for $n \geq 0$ and $1 \leq k \leq K$:

$$-(K-1)\max(n_0, 1 + H_n(\delta)) \leq T_{kn} - n\lambda_k \leq \max(n_0, 1 + H_n(\delta)).$$

Moreover, if $n_0 \geq 6$ then $c \geq 2/19$ is sufficient for this.

Remark 2 *If $f_n = \beta n^p$ (p,$n > 0$) then* (A.13) *can be written as*

$$\log n \leq \frac{\beta\lambda_{\min}^2}{8c^2}\, n^p - \frac{1 + \ell_{K,\delta}}{2}. \tag{A.15}$$

Proof Assume that $n \geq n_0$. Defining

$$g_n = c\sqrt{\frac{\log(\delta_n^{-1})}{f_n}},$$

ng_n is monotone increasing, (A.1) and (A.13) imply (A.9), and (A.14) implies (A.10). The bounds on f_n, K, and δ imply

$$5ng_n = 5nc\sqrt{\frac{\log(4Kn(n+1)/\delta)}{f_n}} \geq 5c\sqrt{n\log(8n_0(n_0+1))},$$

that is at least $\sqrt{n\log(16)} > \sqrt{2n} \geq \lceil\sqrt{n}\rceil$ by the bounds on c and n_0. Moreover, if $n_0 \geq 6$ then $c \geq 2/19$ is enough to assure that the right side above is at least $10\sqrt{n\log(336)}/19 > \sqrt{8n/5} \geq \lceil\sqrt{n}\rceil$ ($n \geq 6$). Thus Lemma 3 gives the result. ∎

Using the previous results we are now in the position to prove a linear lower bound on T_{kn}:

Lemma 5 *Let $0 < \delta \leq 1$ arbitrary. Then there exists an integer N_1 such that for any $n \geq N_1$, $T_{kn} \geq n\lambda_k/2$ holds on A_δ.*

In particular,

$$N_1 = \max\left(\frac{2(K-1)}{\lambda_{\min}}\max(3, n_0'), D_2^4\left[\log D_2^4 + \frac{1}{2}\left(\ell_{K,\delta} + 1.01\right)\right]^2\right), \tag{A.16}$$

where $n_0' = \max(K(K+1), n_1, n_2)$, $n_1 = \max_{1\leq k\leq K}\left[2\log(1/a_k') + 1 + \ell_{K,\delta}\right]^2/a_k'^2$, $n_2 = 4\left(\frac{2c}{\lambda_{\min}}\right)^4\left[8\log\left(\frac{4c}{\lambda_{\min}}\right) + 1 + \ell_{K,\delta}\right]^2$, and $D_2 = 4c(2K-1)/\lambda_{\min}^2$.

For the proof we need the following two lemmata. Lemma 6 quantifies the point when for $a > 0$ the function $at^{1/2} + b$ overtakes $\log t$. Lemma 7 proves that $T_{kn} \geq \sqrt{n}$.

Lemma 6 *Let $a > 0$. For any $t \geq (2/a)^2\left[\log((2/a)^2) - b\right]^2$, $at^{1/2} + b \geq \log t$.*

We develop the proof by presenting some propositions relating $\log t$ and $at^{1/2} + b$.

Let us look at the solution for the following:

$$\log(t) = at^p + b, \tag{A.17}$$

where $a, p, t > 0$.

Let

$$\begin{aligned} \ell(t) &= \log t, \\ q(t) &= at^p + b, \text{ and} \\ t_0 &= (pa)^{-1/p}. \end{aligned}$$

Here t_0 is the point where ℓ and q have the same growth rate, i.e., where $\ell'(t_0) = q'(t_0)$. Note that for $t \geq t_0$, $q'(t) \geq \ell'(t)$. Hence, if $q(t_0) > \ell(t_0)$ then (A.17) has no solutions on $[t_0, \infty)$. Similarly, $q'(t) \leq \ell'(t)$ holds when $t \leq t_0$. Hence, if $q(t_0) > \ell(t_0)$ then (A.17) has no solutions on $(0, t_0]$. Now, consider the case when $q(t_0) \leq \ell(t_0)$. Since for $t \geq t_0$, $q'(t) \geq \ell'(t)$ and $q(t)/\ell(t) \to \infty$, (A.17) will have exactly one solution in $[t_0, \infty)$.

These findings are summarized in the next proposition:

Proposition 1 *Consider $t_0 = (pa)^{-1/p}$, $q(t) = at^p + b$ and $\ell(t) = \log(t)$, where $a, p, t > 0$. Then $q(t_0) \leq \ell(t_0)$ is a sufficient and necessary condition for the existence of a solution to $q(t) = \ell(t)$. When a solution exists, it is unique. Further, t_0 is a lower bound on it.*

Remark 3 *Note that $q(t_0) \leq \ell(t_0)$ is equivalent to $1 + bp \leq -\log(pa)$, which is thus a sufficient and necessary condition for the existence of a solution to $q(t) = \ell(t)$.*

In the sequel we will derive upper bounds on the solution of (A.17) by picking some t^* such that $q(t^*) \geq \ell(t^*)$ and $q'(t^*) \geq \ell'(t^*)$. In doing so we will first consider the homogeneous version of (A.17),

$$\log u = a' u^p. \tag{A.18}$$

The following proposition gives the link between the solutions of the homogeneous and inhomogeneous equations.

Proposition 2 *Any solution of* (A.17) *can be obtained by solving* (A.18) *with* $a' = ae^{pb}$ *and then using* $t = e^b u$ *and vice versa. Further, if* u^* *is an upper bound on the solution of* (A.18) *then* $t^* = e^b u^*$ *is an upper bound on the solution of* (A.17).

Now, consider the linear case when $p = 1$. Our goal is to obtain an upper bound on the solution of (A.17). First, we consider the homogeneous case.

Proposition 3 *Let* $t^* = 2/a \log(1/a)$, $q(t) = at$, $\ell(t) = \log t$, *where* $a > 0$. *Then for any positive* t *satisfying* $t \geq t^*$, $q(t) \geq \ell(t)$ *holds.*

Proof We may assume that $\log(1/a) \geq 1$, or by Remark 3 $q(t) = \ell(t)$ does not have a solution and the statement follows trivially. It suffices to verify that $\ell(t^*) \leq q(t^*)$ and $\ell'(t^*) \leq q'(t^*)$. The second inequality follows from $\log(1/a) \geq 1$, the first follows from the inequality $\log(z^2) \leq z$ which holds for any $z > 0$. ∎

Proposition 4 *Consider* $q(t) = at+b$, $\ell(t) = \log(t)$, *where* $a > 0$. *Let* $t^* = 2/a\,[-b + \log(1/a)]$. *Then for any positive* t *such that* $t \geq t^*$ *it holds that* $q(t) \geq \ell(t)$.

Proof The statement follows immediately from Propositions 2 and 3. ∎

Proposition 5 *Consider* $q(t) = at^{1/2}$, $\ell(t) = \log t$. *Let* $t^* = (2/a)^2 \log^2 (2/a)^2$. *Then for any positive* t *such that* $t \geq t^*$, $q(t) \geq \ell(t)$ *holds.*

Proof By Remark 3 $q(t) = \ell(t)$ has a solution iff $\log(2/a) \geq 1$. Hence, we shall assume w.l.o.g. that this holds. It is easy to show then that $\ell(t^*) \leq q(t^*)$ and $\ell'(t) \leq q'(t)$ hold for $t \geq t^*$ (the first inequality follows from $\log(2z) \leq z$ which holds for any $z > 0$, while the second inequality follows from $\log(2/a) \geq 1$). ∎

Lemma 6 can be stated in terms of $q(t)$ and $l(t)$ as follows:
Let $q(t) = at^{1/2} + b$, $\ell(t) = \log(t)$, where $a > 0$. Let $t^* = (2/a)^2 \left[-b + \log (2/a)^2\right]^2$. Then for any positive t such that $t \geq t^*$ it holds that $q(t) \geq \ell(t)$.

Proof (*Lemma 6*) The statement follows immediately from Propositions 2 and 5. ∎

Lemma 7 *For* $1 \le k \le K$, $n \ge K(K+1)$

$$T_{kn} \ge \sqrt{n} \tag{A.19}$$

holds.

Proof For a positive integer l, let $C_l = \left\{(l-1)^2+1, (l-1)^2+2, \ldots, l^2\right\}$, a partition of $\{1,2,\ldots\}$. Observe that if (A.19) holds for some $n = n' \in C_l$, then it holds also for any $n = n'' \in C_l$, $n'' > n'$, since $T_{k,n''} \ge T_{k,n'} \ge \sqrt{n'} > l-1$ which implies $T_{k,n''} \ge l \ge \sqrt{n''}$. Thus, it is enough to prove (A.19) for $n = K(K+1)$ and then for $n = l^2+1$, $l = K+1, K+2, \ldots$.

By a careful analysis of the algorithm, we see that only forced selection steps happen till $n = K(K+2)$ in a uniform manner, during which each option is selected $K+2$ times. This implies that $T_{k,K(K+1)} = K+1 > \sqrt{K(K+1)}$ and that $T_{k,(K+1)^2+1} \ge T_{k,K(K+2)} = K+2 > \sqrt{(K+1)^2+1}$, i.e., (A.19) holds for $n = K(K+1)$ and $(K+1)^2+1$, for all k. Now we use induction for l. Assume that (A.19) holds for all k, for some $n = (l-1)^2+1$ ($l \ge K+2$), i.e., $T_{k,(l-1)^2+1} \ge \sqrt{(l-1)^2+1} > l-1$ implying $T_{k,(l-1)^2+1} \ge l$. Now at times $(l-1)^2+2, \ldots, l^2+1$ (which total up to $|C_l| = 2l-1 (\ge 2K-3)$ steps), one of those arms for which $T_{k,(l-1)^2+1} = l$ holds is forced to be selected exactly once. Hence each such arm is selected at least once in this phase, assuring $T_{k,l^2+1} \ge l+1 > \sqrt{l^2+1}$ for all k, i.e., (A.19) holds for $n = l^2+1$. ∎

Proof *(Lemma 5)* By Lemma 6, for

$$n \ge n_1 = \max_{1 \le k \le K} \left[2\log(1/a'_k) + 1 + \ell_{K,\delta}\right]^2 / a'^2_k,$$

(A.4) holds with $p = 1/2$, $\beta = 1$ for each k. Due to the forced selection of the options built into the algorithm, $T_{kn} \ge \sqrt{n}$ holds for $n \ge K(K+1)$ (Lemma 7).

Hence, we can apply Lemma 2 and Remark 1 following it with $f_n = n^{1/2}$ and $n_0 = \max(K(K+1), n_1) (\ge 6)$, and get that

$$\left|\hat{\lambda}_{kn} - \lambda_k\right| \le c\sqrt{\frac{\log(\delta_n^{-1})}{n^{1/2}}} \tag{A.20}$$

on A_δ for $n \ge n_0$, $1 \le k \le K$, and $c > 0$ as defined in Lemma 2. Possibly replacing c with $\max(c, 2/19)$, we can assume that $c \ge 2/19$. By Lemma 6 again, for

$$n \ge n_2 = 4\left(\frac{2c}{\lambda_{\min}}\right)^4 \left[8\log\left(\frac{4c}{\lambda_{\min}}\right) + 1 + \ell_{K,\delta}\right]^2,$$

(A.15) holds with $p = 1/2$, $\beta = 1$. Now, we can apply Corollary 1 and Remark 2 following it on A_δ with $f_n = n^{1/2}$ and $n'_0 = \max(n_0, n_2) = \max(K(K+1), n_1, n_2) (\ge 6)$, and get that on A_δ for $n \ge 0$, $1 \le k \le K$, $T_{kn} \ge n\lambda_k - (K-1)\max(n'_0, 1 + H_n(\delta))$, where $H_n(\delta) = D_1 n^{3/4}\sqrt{\log(\delta_n^{-1})}$, and $D_1 = c\left(1 + \frac{2}{\lambda_{\min}}\right) \le 3c/\lambda_{\min}$. Hence, $T_{kn} \ge n\lambda_k/2$ by the time when $n \ge 2n'_0(K-1)/\lambda_{\min}$ and $n \ge 2(K-1)(1+H_n(\delta))/\lambda_{\min}$. Lemma 6 and some

tedious calculations then show that these two constrained are satisfied when $n \geq N_1$, where N_1 is defined as in equation (A.16). ∎

With the help of this result we can get better bounds on T_{kn}, resulting in our first main result:

Theorem 1 *Let $0 < \delta \leq 1$ be arbitrary. Then there exists an integer N_2 and a positive real number D_3 such that for any $n \geq N_2$,*

$$-(K-1)\frac{\max(N_2, 1+G_n(\delta))}{n} \leq \frac{T_{kn}}{n} - \lambda_k \leq \frac{\max(N_2, 1+G_n(\delta))}{n}$$

holds on A_δ, where

$$G_n(\delta) = D_3\sqrt{n\log(\delta_n^{-1})}. \tag{A.21}$$

Here $D_3 \leq 3\sqrt{2}\,c/\lambda_{\min}^{3/2}$,

$$N_2 = \max\left(N_1, \left(\frac{4}{\lambda_{\min}a'_k}\right)\left[\log\left(\frac{2}{\lambda_{\min}a'_k}\right) + \frac{1}{2} + \frac{1}{2}\ell_{K,\delta}\right]\right),$$

where N_1 is defined in Lemma 5.

The theorem shows that asymptotically the GAFS-MAX algorithm behaves the same way as an optimal allocation rule that knows the variances. It also shows that the deviation of the proportion of choices of any option from the optimal value decays as $\tilde{O}(1/\sqrt{n})$.

For the proof we need the counterpart of Lemma 6 for linear functions. The proof is in the Appendix.

Lemma 8 *Let $a > 0$. Then for any $t \geq (2/a)(\log((1/a)) - b)$, $at + b \geq \log t$.*

Proof This statement is restating proposition 3. ∎

Proof *(Theorem 1)* The proof is almost identical to that of Lemma 5. The difference is that now we start with a better lower bound on T_{kn}. In particular, by Lemma 5 $T_{kn} \geq n\lambda_k/2$ holds whenever $n \geq N_1$. By Lemma 2, for some $N_2 \geq N_1$, $c \geq 1$,

$$\left|\hat{\lambda}_{kn} - \lambda_k\right| \leq \frac{c}{\lambda_k^{1/2}}\sqrt{\frac{\log(\delta_n^{-1})}{n}} \tag{A.22}$$

holds for all $n \geq N_2$. In particular, solving (A.4) for n_1 with $f_n = n\lambda_k/2$ and Lemma 8 give that

$$N_2 = \max\left(N_1, \frac{4}{\lambda_{\min}a'_k}\left[\log\left(\frac{2}{\lambda_{\min}a'_k}\right) + \frac{1}{2} + \frac{1}{2}\ell_{K,\delta}\right]\right)$$

will suffice. By Corollary 1, for $n \geq \max(N_2, \lambda_{\min}^{-1}) = N_2$,

$$\begin{aligned} T_{kn} &\leq n\lambda_k + \max(N_2, 1+G_n(\delta)), \quad \text{and} \\ T_{kn} &\geq n\lambda_k - (K-1)\max(N_2, 1+G_n(\delta)), \end{aligned}$$

where $G_n(\delta)$ is given by (A.21) and $D_3 = \sqrt{\frac{c}{\lambda_{\min}}}\,c\left(1 + \frac{2}{\lambda_{\min}}\right)$. ∎

This result yields a bound on the expected value of $\mathbb{E}[T_{kn}]$:

Theorem 2 *Let N_2' be such that $N_2 \leq N_2' \ell_{K,\delta}^2$ holds for any $\delta > 0$, where N_2 is defined in Theorem 1. Then, there exists and index N_3 that depends only on N_2', D_3 and K, such for any $n \geq N_3$,*

$$\mathbb{E}[T_{kn}] \leq n\lambda_k + D_3\sqrt{n(1+\log(4Kn(n+1)))} + 2. \tag{A.23}$$

Proof First note that N_2' exists and $N_2 \leq N_2' \log^2(\delta_n^{-1})$ holds for any $n \geq 2$. Fix $0 < \delta \leq 1$. If $n \geq N_2^2/(D_3^2 \log(\delta_n^{-1}))$, then $1 + G_n(\delta) \geq N_2$, thus it follows from Theorem 1 that for $n \geq \max(N_2, N_2^2/(D_3^2 \log(\delta_n^{-1})))$,

$$\mathbb{P}\left(\frac{T_{kn} - n\lambda_k - 1}{D_3 n^{1/2}} > \sqrt{\log(\delta_n^{-1})}\right) \leq \delta$$

where we used $\mathbb{P}(A_\delta) \geq 1 - \delta$. Let $Z = (T_{kn} - n\lambda_k - 1)/(D_3 n^{1/2})$ and $\varepsilon = \sqrt{\log(\delta_n^{-1})}$. he above inequality is equivalent to

$$\mathbb{P}(Z > \varepsilon) \leq 4Kn(n+1)\, e^{-\varepsilon^2}.$$

By the constraints that connect n and δ, this inequality holds for any pair (n, ε) that satisfy

$$n \geq \max(N_2' \log^2(\delta_n^{-1}), {N_2'}^2 \log^3(\delta_n^{-1})/D_3^2) = \max(N_2' \varepsilon^4, {N_2'}^2 \varepsilon^6/D_3^2),$$

that is, for any (n, ε) such that

$$\varepsilon \leq \min((n/N_2')^{1/4}, (nD_3^2/{N_2'}^2)^{1/6}).$$

Also, since $Z \leq n^{1/2}/D_3$ is always true, $\mathbb{P}(Z > \varepsilon) = 0$ holds for $\varepsilon \geq n^{1/2}/D_3$. We need the following technical lemma, a variant of which can be found, e.g., as Exercise 12.1 in **?**:

Lemma 9 *If $\mathbb{P}(Z > \varepsilon) \leq C \exp(-c\varepsilon^2)$ for any $\varepsilon \leq a$, $a > 0$, and $\mathbb{P}(Z > \varepsilon) = 0$ for any $\varepsilon \geq b\,(\geq a)$, then*

$$\mathbb{E}[Z] \leq \sqrt{(1 + \log C)/c + Cb^2 e^{-ca^2}}. \tag{A.24}$$

Proof By the monotonicity of $\mathbb{P}(Z > \varepsilon) \leq 1$, for any $u > 0$,

$$\begin{aligned}
\mathbb{E}\left[Z^2\right] &= \int_0^\infty \mathbb{P}\left(Z^2 > \varepsilon\right)\, d\varepsilon = \int_0^u + \int_u^{a^2} + \int_{a^2}^{b^2} + \int_{b^2}^\infty \\
&\leq u + \int_u^{a^2} Ce^{-c\varepsilon}\, d\varepsilon + \int_{a^2}^{b^2} \mathbb{P}(Z > a)\, d\varepsilon + 0 \\
&\leq u + \frac{Ce^{-cu}}{c} - \frac{Ce^{-ca^2}}{c} + (b^2 - a^2)Ce^{-ca^2}.
\end{aligned}$$

This gives

$$\mathbb{E}\left[Z^2\right] \leq \frac{1 + \log C}{c} + (b^2 - a^2 - 1/c)Ce^{-ca^2} \leq \frac{1 + \log C}{c} + Cb^2 e^{-ca^2}$$

with the choice $u = (\log C)/c$. Now,

$$\mathbb{E}[Z] \leq \sqrt{\mathbb{E}[Z^2]} \leq \sqrt{\frac{1 + \log C}{c} + Cb^2 e^{-ca^2}}. \qquad \blacksquare$$

Applying Lemma 9 with $a = \min((n/N_2')^{1/4}, (nD_3^2/{N_2'}^2)^{1/6})$, $b = n^{1/2}/D_3$, $C = 4Kn(n+1)$, and $c = 1$,

$$\mathbb{E}[Z] \leq \sqrt{1 + \log(4Kn(n+1)) + 4Kn^2(n+1)e^{-\min((n/N_2')^{1/2}, (nD_3^2/{N_2'}^2)^{1/3})}/D_3^2}.$$

Equation (A.23) then follows by straightforward algebra. ∎

In order to develop a bound on the loss $L_{n,k}$ we need Wald's (second) identity:

Lemma 10 (Wald's Identity, Theorem 13.2.14 of ?) *Let $\{\mathcal{F}_t\}$ be a filtration and let Y_t be an $\mathcal{F}_t$-adapted sequence of i.i.d. random variables. Assume that $\mathcal{F}_t$ and $\sigma(\{Y_s : s \geq t+1\})$ are independent and T is a stopping time w.r.t. $\mathcal{F}_t$ with a finite expected value: $\mathbb{E}[T] < +\infty$. Consider the partial sums $S_n = Y_1 + \ldots + Y_n$, $n \geq 1$. If $\mathbb{E}\left[Y_1^2\right] < +\infty$ then*

$$\mathbb{E}\left[(S_T - T\mathbb{E}[Y_1])^2\right] = \mathrm{Var}[Y_1]\,\mathbb{E}[T]. \tag{A.25}$$

The following theorem is the main result of the paper:

Theorem 3 *Fix k, $n \geq N_2$, where N_2 is as in Theorem 1. Then*

$$L_n \leq L_n^* + \tilde{O}(n^{-3/2}).$$

Proof Let $S_{kn} = \sum_{t=1}^n X_{kt}$, $\hat{L}_{kn} = (S_{k,T_{kn}} - T_{kn}\mu_k)/T_{kn}$, $G'(n,\delta) = (K-1)\max(N_2, 1 + G_n(\delta))$ and

$$G''(n) = D_3\sqrt{n(1 + \log(4Kn(n+1)))} + 2.$$

Note that by Theorem 1,

$$\mathbb{P}(T_{kn} \leq n\lambda_k - G'(n,\delta)) \leq P(n,\delta) \stackrel{\text{def}}{=} \mathbb{I}_{\{n<N_2\}} + \mathbb{I}_{\{n\geq N_2\}}\delta \tag{A.26}$$

holds for any $n \geq 1$ and $0 < \delta \leq 1$. Then, for any $0 < \delta \leq 1$,

$$\begin{aligned}
L_{kn} &= \mathbb{E}\left[\hat{L}_{kn}^2\right] \\
&= \mathbb{E}\left[\hat{L}_{kn}^2 \mathbb{I}_{\{T_{kn} > n\lambda_k - G'(n,\delta)\}}\right] + \mathbb{E}\left[\hat{L}_{kn}^2 \mathbb{I}_{\{T_{kn} \leq n\lambda_k - G'(n,\delta)\}}\right] \\
&\leq \frac{\mathbb{E}\left[(S_{k,T_{kn}} - T_{kn}\mu_k)^2\right]}{(n\lambda_k - G'(n,\delta))^2} + R^2\,\mathbb{P}(T_{kn} \leq n\lambda_k - G'(n,\delta)) \\
&= \frac{\sigma_k^2\mathbb{E}[T_{kn}]}{(n\lambda_k - G'(n,\delta))^2} + R^2\,\mathbb{P}(T_{kn} \leq n\lambda_k - G'(n,\delta)) \qquad \text{(by Lemma 10)} \\
&= \frac{\sigma_k^2\mathbb{E}[T_{kn}]}{(n\lambda_k - G'(n,\delta))^2} + R^2\,P(n,\delta) \qquad \text{(by (A.26))} \\
&\leq \frac{\sigma_k^2(n\lambda_k + G''(n))}{(n\lambda_k - G'(n,\delta))^2} + R^2\,P(n,\delta) \qquad \text{(by A.23)} \\
&= \frac{\sigma_k^2}{n\lambda_k}\frac{1}{(1 - G'(n,\delta)/(n\lambda_k))^2} + \frac{\sigma_k^2 G''(n)}{(n\lambda_k - G'(n,\delta))^2} + R^2\,P(n,\delta).
\end{aligned}$$

Now choose $\delta = n^{-3/2}$. Then, for n sufficiently large, $G'(n, n^{-3/2})/(n\lambda_k) \leq 1/2$. Further, since $N_2 \leq N_2'\ell_{K,\delta}$, for n sufficiently large $\mathbb{I}_{\{n<N_2\}} \leq \mathbb{I}_{\{n<N_2'\log(4Kn^{3/2})\}} = 0$ and thus $P(n,\delta) = \delta$.

Therefore, for n sufficiently large, using $1/(1-x) \leq 1+2x$ for $0 \leq x \leq 1/2$, we get,

$$L_{kn} \leq \frac{\sigma_k^2}{n\lambda_k}\left(1+2\frac{G'(n,n^{-3/2})}{n\lambda_k}\right)^2 + \frac{\sigma_k^2 G''(n)}{(n\lambda_k - G'(n,n^{-3/2}))^2} + R^2\, n^{-3/2},$$

which gives

$$L_{kn} \leq \frac{\sigma_k^2}{n\lambda_k} + \tilde{O}(n^{-3/2}) = \frac{\Sigma^2}{n} + \tilde{O}(n^{-3/2}) = L_n^* + \tilde{O}(n^{-3/2}).$$

Taking the maximum with respect to k yields the desired result. ∎

Bibliography

N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 1–9, 1998.

A. Antos, V. Grover, and Cs. Szepesvári. Active learning in multi-armed bandits. In *Algorithmic Learning Theory*, 2008.

J. Y. Audibert, R. Munos, and Cs. Szepesvári. Tuning bandit algorithms in stochastic environments. In *Algorithmic Learning Theory*, pages 150–165, 2007.

P. Auer, P. Fischer, and N. Cesa-Bianchi. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47:235–256, 2002. URL `http://www2.imm.dtu.dk/pubdb/p.php?2088`.

M. Balcan, S. Hanneke, and J. Wortman. The true sample complexity of active learning. In *Conference on Learning Theory (COLT) 2008*, 2008.

Statistics Canada. Your guide to the consumer price index, 1996. URL `http://www.statcan.ca/english/freepub/62-557-XIB/62-557-XIB1996001.pdf`.

O. Chapelle. Active learning for Parzen window classifier. In *In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 49–56, 2005.

D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

S. Dasgupta. Coarse sample complexity bounds for active learning. In *Neural Information Processing Systems (NIPS) 2005*, 2005.

S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Neural Information Processing Systems (NIPS) 2007*, 2007.

A. Emery and A. Nenarokomov. Optimal experiment design. *Measurement Science and Technology*, 9:846–876, 1998.

P. Etore and B. Jourdain. Adaptive optimal allocation in stratified sampling methods. In *Methodology and Computing in Applied Probability*, 2007. URL `http://www.citebase.org/abstract?id=oai:arXiv.org:0711.4514`.

G.S. Fishman. *Monte Carlo Concepts, Algorithms and Applications*. Springer-Verlag, 1999.

Y. Freund, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, pages 133–168, 1997.

P. Glasserman, P. Heidelberger, and P. Shahabuddin. Asymptotically optimal importance sampling and stratification for pricing path-dependent options. *Mathematical Finance*, 9: 117–152, 1999.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

J. Kiefer and J. Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12:363–366, 1960.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.

A. McCallum and K. Nigam. Employing EM and pool-based active learning for text classification. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 350–358, 1998.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, 2001.

A. I. Schein. *Active learning for logistic regression*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 2005. Supervisor-Lyle H. Ungar.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *COLT '92: Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294, 1992.